Interaction field : a new intuitive method to sketch collective behaviors

Adèle Colas

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N°601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

Adèle COLAS

Interaction Field: A New Intuitive Method to Sketch Collective Behaviors

Thèse présentée et soutenue à Rennes, le 18 novembre 2022
Unité de recherche : Inria, Centre Inria Rennes-Bretagne Atlantique (Inria-Rennes)

Rapporteurs avant soutenance :

Damien Rohmer	Professeur à École Polytechnique
Christopher Peters	Enseignant Chercheur à KTH Royal Institute of Technology

Composition du Jury :

Président :	Anatole Lécuyer	Directeur de Recherche à Inria
Examinatrices :	Nuria Pelechano	Enseignante Chercheuse à Universitat Politecnica de Catalunya
	Carol O'Sullivan	Professeure à Trinity College Dublin
Dir. de thèse :	Julien Pettré	Directeur de Recherche à Inria
	Anne-Hélène Olivier	Maîtresse de Conférences à Université Rennes 2
Encandrants de thèse :	Ludovic Hoyet	Chargé de Recherche à Inria
	Claudio Pacchierotti	Chargé de Recherche au CNRS

RÉSUMÉ EN FRANÇAIS

La simulation en temps réel du comportement d'une foule a de nombreuses applications, notamment dans les jeux vidéos, les films d'animation et la Réalité Virtuelle. De nombreux logiciels de simulation existent permettant de restituer des flux moyens d'individus appelés "agents", chaque agent se dirige vers son objectif, tout en étant soumis à des contraintes (utiliser un passage piéton pour traverser une rue par exemple). De nombreux algorithmes existent également pour traiter des tâches de pilotage communes à tous les agents comme l'évitement des collisions entre agents ou avec un obstacle, ou des comportements généraux de groupe, comme un regroupement ou une dispersion. Ces algorithmes sont difficilement adaptables à de nouveaux comportements plus singuliers, comme tourner autour d'un agent ou fuir un agent tout en suivant le flux moyen. La prise en compte de ces interactions locales singulières est nécessaire à la réalisation de scénarios spécifiques plus variés pour une simulation plus réaliste d'environnements virtuels peuplés.

L'objectif de cette thèse est de proposer une méthode générique intervenant en perturbation locale du flux moyen, permettant de simuler des comportements locaux singuliers au sein de l'évolution globale d'une foule. De plus, cette méthode doit être d'un usage simple et rapide pour un utilisateur de logiciels de modélisation des mouvements de foule.

Dans cette thèse, un *état de l'art* est d'abord fait. La méthode du *champ d'interaction* est ensuite présentée, ainsi que l'*interface graphique*, *éditeur*, qui permet de générer ces champs. Puis, on s'intéresse à son *implémentation*, *couplage avec le logiciel de simulation 2D* de la foule. Le couplage avec un logiciel d'*animation 3D*, bien que n'étant pas dans les objectifs premiers de la thèse, est évoqué. Il permet de rendre les simulations réalisées plus réalistes. Des *résultats* sur des exemples significatifs, validant le codage et le couplage sont présentés. L'outil ainsi construit a été soumis à une campagne approfondie de *tests utilisateurs* pour en vérifier l'efficacité et la pertinence. Enfin, les résultats d'une interface prototype en *Réalité Virtuelle* sont également montrés.

La Figure 1, ci-dessous, illustre les différentes étapes de la mise en œuvre du modèle proposé des champs locaux d'interaction.

État de l'art

Des méthodes existent permettant de modifier artistiquement un comportement localement dans la foule. Elles passent par le contrôle direct des paramètres de la simulation ou l'édition directe de trajectoires spécifiques, elles nécessitent souvent l'écriture de scripts complexes et sont fastidieuses. La difficulté de créer de nouveaux types de comportement interactif a, nous semble-t-il, limité la variété des

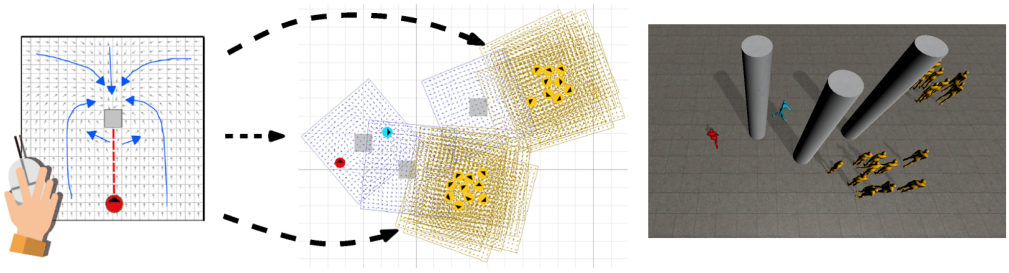


Figure 1 – Conception par dessin d'un croquis d'un champ d'interaction local entre agents utilisé dans une simulation de foule. A gauche : un utilisateur dessine les lignes de champ (en bleu), qui sont converties en une grille définissant le champ d'interaction. L'objectif de ce champ d'interaction spécifique est de conduire les agents qui y sont sensibles à se placer derrière un objet pour se cacher d'un autre agent. Au milieu : vue de la simulation 2D. Les obstacles en gris et les agents en orange ne sont pas sensibles au champ et l'ignorent. L'agent en bleu est sensible au champ, ce qui l'amène à se cacher de l'agent en rouge derrière l'obstacle source du champ. A droite : vue 3D de ce même scénario. L'animation 3D du corps de chaque agent est combinée avec la simulation de foule 2D.

scénarios simulés jusqu'à maintenant.

Le concept proposé du champ d'interaction semble, à première vue, proche du champ de navigation de Patil et al. [Patil et al., 2011] : une grille donne une direction de marche optimale en tout point du plan, éventuellement à partir de croquis. Cependant, alors que les champs de navigation spécifient des chemins globaux, les champs d'interaction spécifient comment chaque agent se comporte localement, comme une perturbation de la simulation globale.

On peut remarquer aussi que la liste des interactions locales prédéfinies proposée par Reynolds [Reynolds, 1999] n'a jamais été substantiellement étendue, malgré de nombreux développements en terme de modèles de simulation. Le concept d' agent peut aussi être rendu plus théorique pour la modélisation [Schuerman et al., 2010; Yeh et al., 2008], mais cela ne rend pas nécessairement de nouveaux comportements faciles à produire par des non-experts.

Champ d'interaction

Un champ d'interaction (CI) est un champ au sens classique de l'analyse vectorielle (comme le champ électrique, par exemple). Il traduit une propriété de l'espace définie en tout point du domaine de définition du champ, il peut être stationnaire ou dépendre du temps. Les champs se superposent, la valeur en un point est la somme des valeurs de chaque champ.

Comme couramment dans la recherche sur la simulation de foule, les scénarios sont modélisés dans le plan 2D, dans lequel on cherche à modéliser la trajectoire des différents agents.

Dans le modèle développé, le champ d'interaction donne en chaque point du plan, ou dans un domaine réduit, suivant sa portée (par exemple une mauvaise odeur) un vecteur vitesse, ou uniquement une direction par l'intermédiaire d'un vecteur unitaire. Le champ est généré par une source qui est un

agent, un obstacle ou une singularité de l'environnement, il est attaché à cette source, si elle se déplace, le champ se déplace avec elle. Par commodité, nous associons donc à chaque source un repère cartésien local dont l'origine se confond avec la source de coordonnées $(0,0)$ et dont les axes sont tels que le sens de l'agent défini de l'arrière vers l'avant est le sens des y négatifs.

Quel que soit un point $M(x,y)$ de vecteur position $\mathbf{p} = \mathbf{OM}$ appartenant à D, $D \subset \mathbb{R}^2$ représentant la portée du champ, le champ d'interaction créé par la source s, est la fonction qui associe le vecteur $\mathbf{v}$ à $M(x,y)$:

$$\begin{array}{rccc} \text{CI}: & D & \longrightarrow & \mathbb{R}^2 \\ & M(x,y) & \longmapsto & \mathbf{v} \end{array}$$

Si un agent est défini comme sensible à ce champ, il est "receveur" de champ, sa vitesse de déplacement est $\mathbf{v}$ (s'il n'est pas soumis à d'autres interactions).

La valeur du champ et sa portée peuvent être fonction de divers paramètres qui interviennent comme commandes de contrôle. On parlera alors d'un champ d'interaction paramétrique : pour un paramètre q, $M(x,y) \longmapsto \mathbf{v}(q)$ dans $D^*(q)$. Pour calculer les valeurs du champ pour chaque paramètre, une méthode d'interpolation économique et rapide a été retenue. Pour un paramètre donné, les autres étant constants, on calcule les valeurs du champ pour un nombre restreint de valeurs clefs q_i, $i \in \{1,\cdots,l\}$, de ce paramètre fournies par l'utilisateur. Les valeurs du champ pour une valeur quelconque du paramètre seront calculées par interpolation linéaire suivant le besoin.

En illustration, nous montrons deux exemples de champ paramétrique. Le premier traite le cas où le paramètre est l'angle α entre la direction définie par la source s et un agent o et l'axe des x. La Figure 2 montre le résultat de l'application d'un champ de vitesse paramétré par cet angle α. Ce champ de source s impose la vitesse de déplacement à des agents voisins de s de façon à toujours être caché par un objet situé en s, d'un agent situé en o. L'effet du paramètre α est simplement une rotation du repère local à s sans changer les valeurs du champ dans ce repère, ces valeurs sont alors projetées sur le repère fixe du scénario. La source s reste fixe dans le repère du scénario.

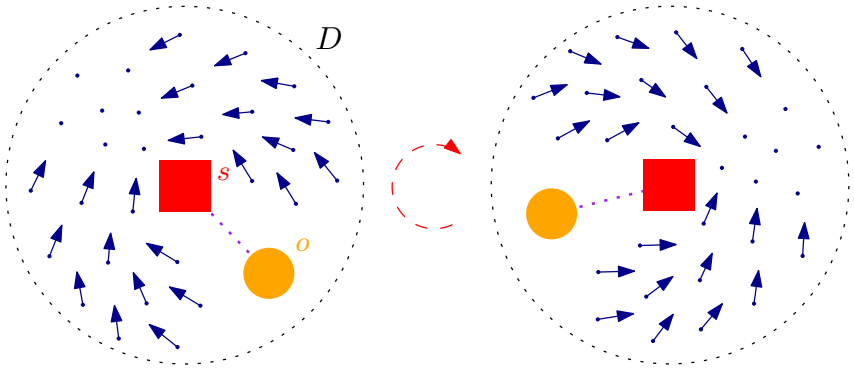


Figure 2 – Champ d'interaction paramétrique basé sur une relation angulaire. Le champ d'interaction dépend de la relation angulaire entre la source s et un autre objet o (scénario "cache-cache").

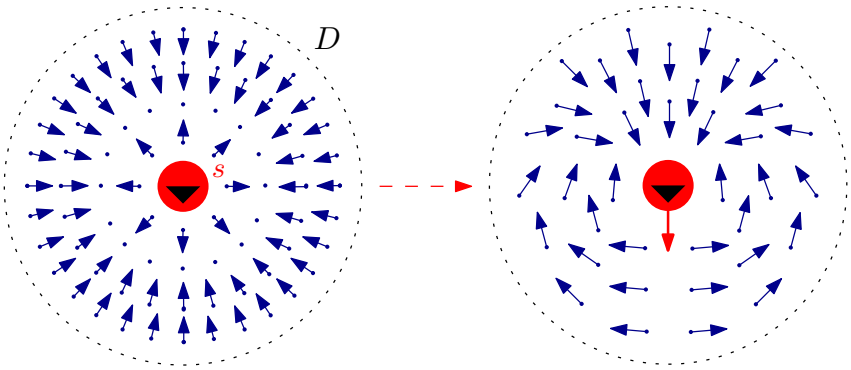


Figure 3 – Champ d'interaction paramétrique basé sur la vitesse de déplacement de la source *s*. A gauche : la source (en rouge) est immobile. A droite : la source (en rouge) se déplace (Scénario "VIP").

Le deuxième exemple traite de l'action d'un champ d'interaction (interaction-répulsion) où le paramètre est la vitesse de déplacement de la source du champ (Figure 3).

Éditeur graphique

Dans cette partie, nous nous intéressons à l'interface homme-machine (IHM) construite pour permettre une génération simple et intuitive des champs locaux d'interaction, deuxième étape de la Figure 1. Cet éditeur a été programmé en C++.

Le processus est le suivant. L'utilisateur esquisse le comportement souhaité. A cet effet, l'éditeur lui donne accès à plusieurs outils graphiques. L'utilisateur dessine alors des lignes de champ indicée i, auxquelles il associe le module v_i du champ. La direction du champ est, par définition, en tout point tangente à la courbe. Son sens est le sens d'esquisse de la ligne. L'utilisateur peut ensuite indiquer des zones nulles ("zéros"), dont le complément est le domaine D d'application du champ, le scénario ne s'étendant notamment jamais au plan infini. Ces zones sont traitées dans l'éditeur, on annule directement les valeurs du champ. L'utilisateur fournit également la liste des agents "receveurs" pour chaque champ, et, en cas de paramétrage, une liste de valeurs clefs du paramètre.

La numérisation des lignes de champ est réalisée par échantillonnage tous les 0.1 m (échelle liée à l'homme). Le calcul du champ en tout point de D, est réalisé par interpolation bilinéaire entre toutes les valeurs numérisées. Pour chaque champ, toutes ces données issues de l'éditeur graphique sont fournies au logiciel hôte, *UMANS*, sous forme de fichiers au format xml and txt. Le couplage avec le simulateur de foule permet à l'utilisateur de visualiser directement le comportement généré par son esquisse et de corriger son esquisse si nécessaire. Il peut ainsi, par itération successive sur son dessin, converger vers le comportement souhaité. Grâce à l'éditeur, les champs d'interaction sont donc spécifiés de manière visuelle, permettant aux utilisateurs de tout niveau de créer de nouveaux comportements avec une relative facilité.

Simulation 2D, implémentation

Grâce à l'éditeur, nous avons construit à partir des esquisses utilisateur, des champs locaux d'interaction dans leurs repères cartésiens liés à la source. Il faut maintenant les intégrer dans un logiciel de simulation 2D (Figure 1 étape du milieu).

Dans la simulation du déplacement de la foule au cours du temps, la source s est potentiellement mobile, si c'est un agent. Le repère relatif à la source dans lequel le champ est défini est donc en mouvement par rapport au repère fixe du scénario. A chaque temps du calcul, il faut donc projeter le champ sur le repère fixe, pour avoir les valeurs correctes du champ dans ce repère, à appliquer aux "receveurs". D'après la théorie des champs, le vecteur champ en un point est la résultante des champs qui agissent en ce point. La vitesse de déplacement d'un agent est le vecteur résultant de la vitesse de déplacement donnée par la simulation et de tous les champs auxquels il est sensible. C'est ce vecteur vitesse qui donnera la trajectoire "perturbée" de l'agent et sa position au temps suivant.

L'éditeur de champ a été écrit en C++ et un simulateur de foule enrichi des champs d'interaction a été implémenté en C++ en dehors de la plateforme d'origine *UMANS*. Il est obtenu en étendant un cadre existant de simulation de foule en temps réel [van Toll et al., 2020] reposant sur des agents, pour accepter les champs d'interaction. Les champs sont projetés sur une grille régulière de pas modifiables, la valeur du vecteur champ en un point est obtenue par interpolation bilinéaire entre les mailles de la grille voisine du point.

Animation 3D

L'animation de personnage en 3D (Figure 1 étape de droite) n'entrant pas dans le cadre de cette thèse, nous nous sommes concentrés sur l'intégration de techniques déjà existantes à notre logiciel. Nous avons choisi Motion Matching, disponible dans Unity Store. Motion Machine appelle périodiquement une base de données d'animation pour trouver l'image correspond le mieux à un ensemble de propriétés, comme la position des pieds du personnage en début d'itération ou le déplacement prévu sur le pas de temps. Une fois l'image la plus adaptée trouvée, la lecture de l'animation se poursuit à partir de ce point, un lissage est réalisé pour éliminer la discontinuité à la reprise. Claassen [Animation Uprising, 2020] a proposé une version de la correspondance de mouvement implémentée dans le moteur de jeu Unity appelée "MxM", disponible directement dans l'Unity Store. Néanmoins pour avoir des résultats vraiment satisfaisants, nous avons dû réaliser nous-même des sessions captures de mouvement. L'"animateur", qui utilise des enregistrements de mouvements réels, permet de filtrer de possibles trajectoires irréalistes, issues de comportements non réalisables par un humain produits par les champs d'interaction dessinés par l'utilisateur.

Les résultats obtenus sont convaincants, mais Motion Matching nécessite beaucoup de données d'animation pouvant varier de scénarios en scénarios. la pertinence et la quantité des données enregistrées ont un impact direct sur la qualité des résultats. Le résultat final est un compromis entre la qualité

de l'animation et la fidélité à la trajectoire 2D d'entrée.

Résultats

Cette section montre l'intérêt de l'usage du modèle du champ d'interaction au travers d'un certain nombre d'exemples de scénarios. Ils illustrent non seulement l'intérêt spécifique du champ d'interaction, mais aussi la facilité de son intégration dans des scénarios de plus en plus complexes.

Pour chaque scénario, nous montrons les champs locaux d'interaction d'entrée créés dans l'éditeur, ainsi que des captures d'écran de la simulation résultante. Toutes les captures d'écran de simulation comprennent une grille avec des mailles de 1×1 m, pour donner à l'échelle de l'environnement. Comme illustrations, nous montrons également quelques projections du champ d'interaction sur l'environnement. Le lecteur est invité à regarder les vidéos supplémentaires Table 6.1 qui montrent notamment le processus de conception des champs locaux d'interaction, plusieurs scénarios résultats en mouvement, et des couplages avec l'animation de personnages en 3D.

Scénario 1 : Cache-cache

Le premier scénario utilise un champ d'interaction paramétrique dépendant de l'angle α pour permettre à un agent de se cacher derrière un objet.

Ce champ d'interaction, représenté dans la Figure 4(a), a été dessiné à l'aide de 7 lignes de champ et d'un lien de rotation.

Dans la version la plus simple du scénario (Figure 4(b)), un obstacle O immobile émet ce champ d'interaction, avec un agent A_0 contrôlé par l'utilisateur comme objet lié. Un agent A_1 répond au champ d'interaction. Lorsque l'utilisateur déplace A_0, A_1 se cache automatiquement derrière O en fonction de l'endroit où se trouve A_0.

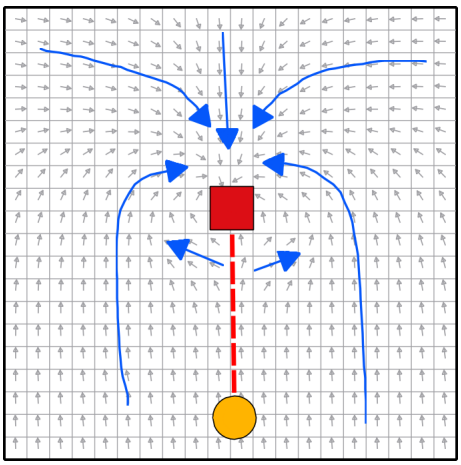
Dans le scénario étendu présenté dans la Figure 4(c), on a ajouté plusieurs obstacles et agents qui émettent tous le même champ d'interaction. Par conséquent, l'agent A_1 se cache derrière l'objet le plus proche, en traitant les obstacles et les agents de la même manière. Les agents supplémentaires ne répondent pas aux champs d'interaction, mais utilisent la prévention des collisions pour laisser la place à l'utilisateur si nécessaire.

Dans le scénario étendu, on montre qu'un champ d'interaction est un composant de simulation facilement réutilisable applicable à d'autres agents ou obstacles.

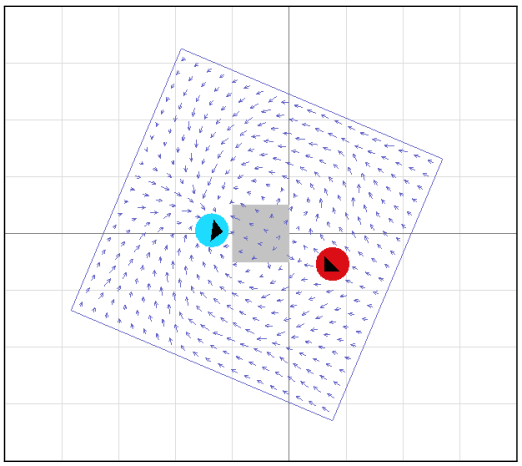
Scénario 2 : VIP dans une foule

Dans cet exemple, une foule s'écarte au passage d'un personnage illustre "VIP" contrôlé par l'utilisateur et le regarde.

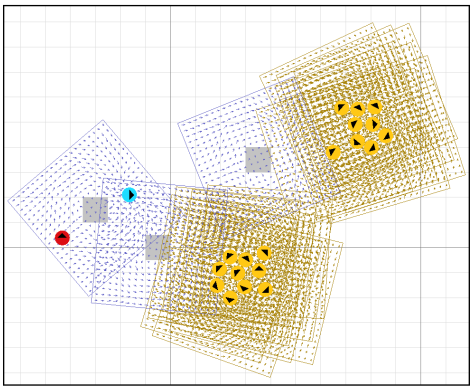
Pour modéliser ce comportement, nous faisons en sorte que l'agent VIP émette deux champs locaux d'interaction :



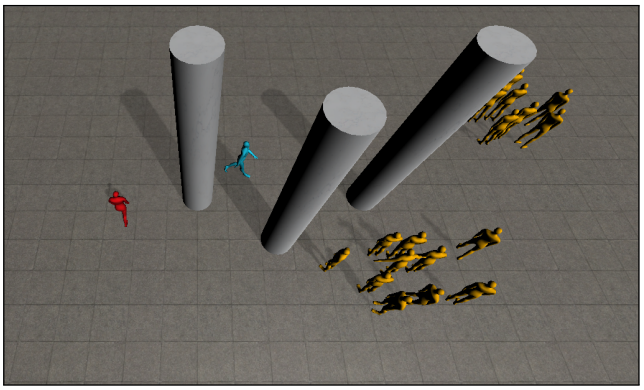
(a) Champ d'interaction en vitesse (5×5 **m)**



(b) Simulation (simple)



(c) Simulation (étendue)



(d) Visualisation 3D

Figure 4 – Résultats du scénario *"Cache-cache"*. (a) Champ d'interaction en vitesse avec un lien de rotation (segment pointillé rouge) entre la source (rouge) et un second objet (orange). Les lignes de champ sont indiquées en bleu. (b) Simulation où l'agent bleu utilise ce champ pour se cacher de l'agent rouge contrôlé par l'utilisateur. (c) Simulation où l'agent bleu peut se cacher derrière tous les obstacles et tous les agents orange, chacun émettant le même CI. (d) Impression 3D avec les deux agents principaux à gauche.

— un champ d'interaction paramétrique de vitesse qui dépend de la vitesse de la source (VIP) (Figure 5(a)),

— et un champ d'interaction d'orientation qui conduit les agents qui y sont sensibles, à regarder la source (VIP) (Figure 5(b)). Pour le champ d'interaction de vitesse, sa portée augmente quand la vitesse de la source (VIP) augmente, l'effet de poussée sur la foule devient plus important.

La simulation présente une petite foule. Le but de chaque agent est fixé à sa position de départ, de sorte que les agents reviennent à leur ancienne position après le passage du VIP. Les Figures 5(d) et 5(e) montrent la différence de réaction de la foule selon la vitesse de déplacement du VIP.

Enfin, cinq agents "garde du corps" sont inclus dans le scénario. Les gardes du corps sont soumis à un champ d'interaction en vitesse émis par le VIP auquel ils sont les seuls sensibles.

Ce champ d'interaction (représenté dans la Figure 5(c)) est à nouveau paramétrique, avec la vitesse de déplacement du VIP comme paramètre : il permet aux gardes du corps de s'aligner sur le VIP lorsqu'il se déplace, et de se regrouper autour de lui lorsqu'il est immobile. Cette dernière image-clé du champ d'interaction utilise des zones nulles ("zéros") pour permettre aux gardes du corps de s'arrêter en cercle autour du VIP.

Les gardes du corps eux-mêmes émettent le même champ d'interaction répulsif que le VIP. La Figure 5(f) montre un exemple de simulation avec des gardes du corps.

Ce scénario montre comment un champ d'interaction paramétrique avec des images clés peut être facilement conçu pour créer un effet très spécifique dans la foule. Le même champ d'interaction peut (encore) être réutilisé pour plusieurs autres sources, et il est possible, dans la simulation, de spécifier la sensibilité de différents agents à différents champs d'interaction.

Scénario 3 : Musée

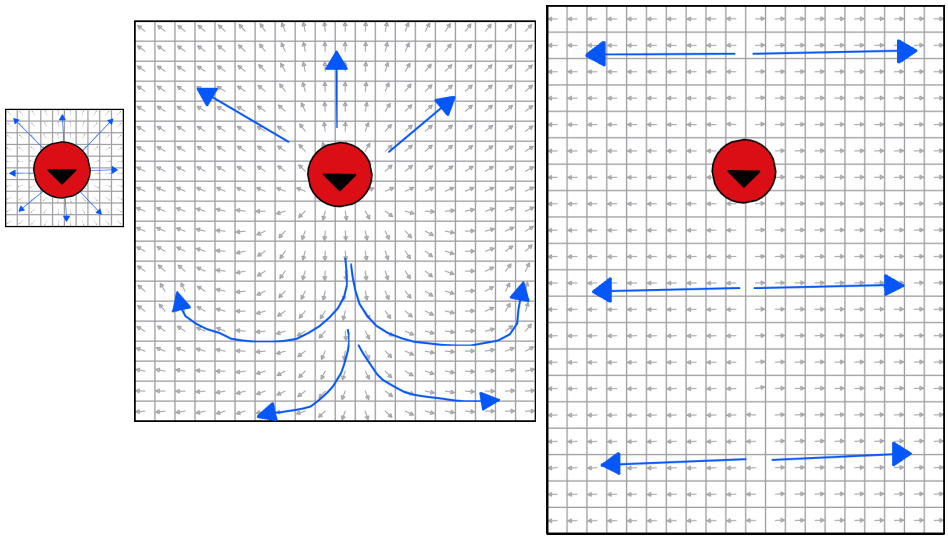
Le dernier exemple est le scénario du musée où huit visiteurs se déplacent dans une galerie en observant les tableaux exposés.

Le pilier central émet deux champs d'interaction distincts, un génère un déplacement autour de lui dans le sens des aiguilles d'une montre, un autre dans le sens inverse. Chaque agent est déclaré sensible à l'un ou l'autre de ces deux champs.

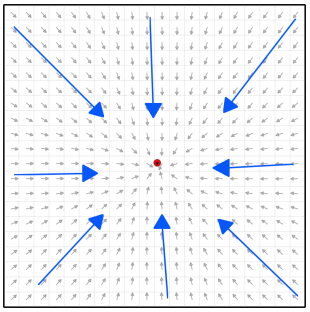
La Figure 6(a) montre le champ d'interaction en vitesse dans le sens indirect.

Chaque tableau émet un champ d'interaction en vitesse avec une zone zéro qui permet aux agents de rester immobiles à une certaine distance de ce tableau pour le contempler ; ces champs d'interaction sont présentés sur la Figure 6(b).

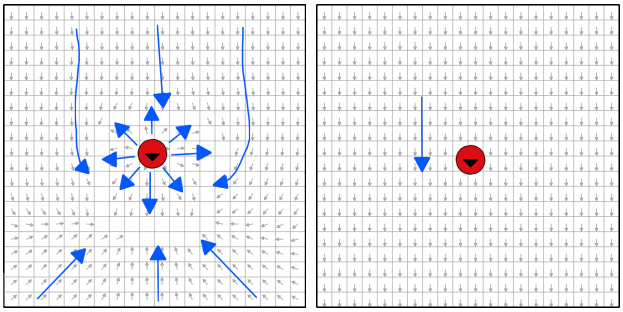
Chaque tableau émet également un champ d'interaction d'orientation qui conduit les agents sensibles à ce champ à faire face au tableau, les agents peuvent avoir des habitudes de visite différents et passé plus ou moins de temps devant les peintures. Ces champs ne sont pas représentés sur les figures pour des raisons de lisibilité.



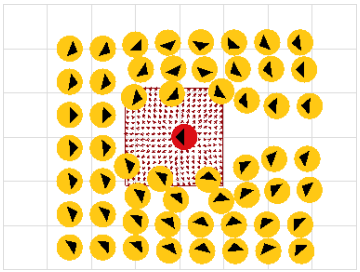
(a) Champ de vitesse perçu par la foule, pour $v = 0$ m/s (grille 1×1 m), 1 m/s (grille 3×3 m), et 1.8 m/s (grille 3×4 m)



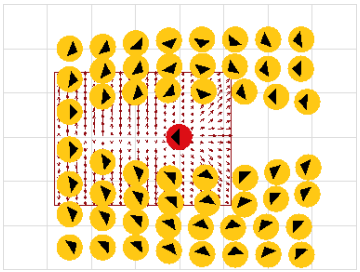
(b) Champ d'orientation perçu par la foule (20×20 m)



(c) Champ de vitesse perçu par les gardes du corps, pour $v = 0$ m/s et $v = 1$ m/s (5×5 m)



(d) Simulation (vitesse VIP : $0,6$ m/s)



(e) Simulation (vitesse VIP : $1,8$ m/s)

(f) Simulation avec gardes du corps

Figure 5 – Résultats du scénario *VIP dans une foule* (). (a) Images clés du champ de vitesse perçu par la foule. (b) Champ d'orientation perçu par la foule. (c) Images clés du champ de vitesse perçu par les gardes du corps. (d–e) Exemples de simulations avec différentes vitesses de déplacement du VIP (en rouge). Le champ d'interaction interpolé est également représentée. (f) Exemple de simulation avec des gardes du corps (en bleu foncé). Les champs d'interaction ne sont pas représentés pour une meilleure lisibilité.

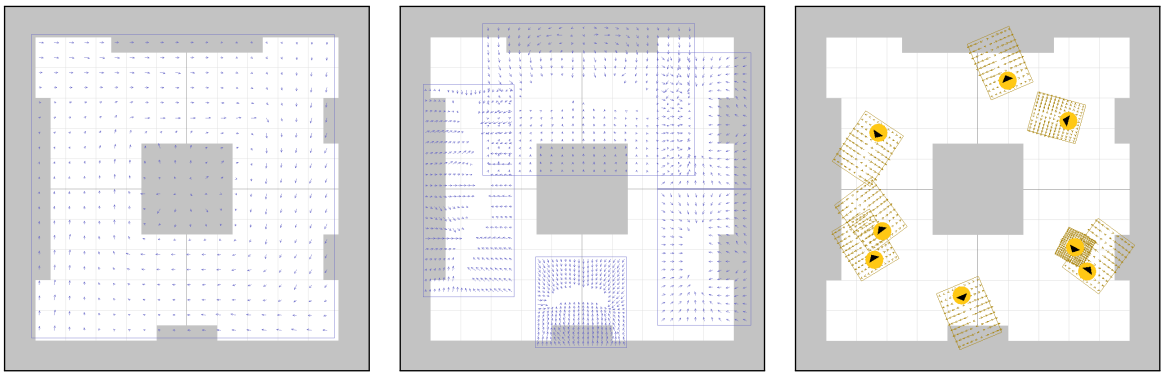
De plus, chaque agent A_i émet un champ d'interaction de vitesse paramétrique qui interdit aux autres agents sensibles de se placer dans sa ligne de vue lorsqu'il est immobile. Les autres visiteurs évitent poliment de gêner A_i lorsqu'il regarde un tableau. La Figure 6(b) montre une capture d'écran de la simulation et les champs d'interaction des agents.

D'autres résultats sont disponibles 6.1 dans les vidéos supplémentaires.

Pour permettre aux agents de passer de la marche à la contemplation d'un tableau, nous avons ajouté la possibilité d'activer ou de désactiver les champs à l'aide de chronomètres. Lorsqu'un agent entre pour la première fois dans le domaine d'un champ d'interaction de vitesse émis par un tableau, l'agent ignore le champ d'interaction de la galerie pendant un certain nombre de secondes. Lorsque ce temps est écoulé, l'agent ignore le champ d'interaction du tableau et est à nouveau sensible au champ d'interaction de la galerie, ce qui lui permet de continuer la visite. En revanche, les champs d'interaction d'orientation restent activés en permanence, de sorte que les agents font toujours face aux tableaux qui sont à leur portée.

Le système de minuterie ne fait pas partie de la technique du champ d'interaction elle-même, il a nécessité un effort de modélisation/programmation supplémentaire spécifiquement pour ce scénario. C'est le seul exemple avec un tel système de minuterie supplémentaire.

L'utilisation de champs locaux d'interaction a montré leur capacité à réaliser des scénarios complexes qui n'auraient pu être réalisés que difficilement avec d'autres techniques.



(a) Champ de vitesse (couloir) **(b) Champ de vitesse d'interaction avec les (tableaux)** **(c) Simulation**

Figure 6 – Résultats pour le scénario *Musée*. (a) Un des champs d'interaction en vitesse pour la marche autour du pilier central. (b) Champs d'interaction de vitesse pour les cinq tableaux. (c) Capture d'écran de la simulation, montrant également les champs d'interaction paramétriques autour des agents debout et en mouvement.

Étude utilisateur

Nous pouvons maintenant nous demander si ces scénarios sont effectivement aussi facilement reproductibles que nous le prétendons. Pour répondre à cette question, une étude a été réalisée pour tester la convivialité de l'interface, la facilité d'utilisation et l'intuitivité de la méthode. Vingt deux utilisateurs familiarisés avec l'animation par ordinateur, mais ignorant la méthode des champs d'interaction, ont participé à cette étude. Les scénarios à mettre en œuvre sont en majorité différents de ceux présentés dans la section précédente et constituent donc de nouveaux exemples d'utilisation du champ d'interaction. Ils vont d'un simple champ d'interaction de vitesse au scénario de "cache-cache".

Les participants ont réalisé l'étude en laboratoire, en présence de l'expérimentateur, en utilisant deux écrans de 24 pouces avec une fenêtre éditeur pour dessiner les champs et une fenêtre de simulation pour voir le comportement résultant, ce qui permettait d'améliorer leur croquis de champ d'interaction de manière interactive en affichant sur le deuxième écran le résultat de la simulation.

Tous les participants ont commencé par une courte session de formation guidée par vidéo, au cours de laquelle ils ont pu explorer librement l'outil de création des champs d'interaction et interagir avec l'expérimentateur. A l'issue de cette formation initiale, les participants ont été invités à esquisser des champs d'interaction pour sept scénarios de complexité croissante. Des formations spécifiques sous forme de tutoriel vidéo et de scénarii exemples, sur le contrôle de la vitesse ou la création d'un champ paramétrique, par exemple, étaient délivrées avant la réalisation de tâches pour lesquelles ces compétences étaient requises. Les tâches avaient été conçues de manière à ne nécessiter qu'un petit nombre de champs d'interaction chacune, et ordonnées de façon à présenter progressivement aux utilisateurs toutes les fonctionnalités des champs d'interaction (par exemple, les champs d'interaction paramétriques ont été traités en dernier).

Après chaque tâche d'évaluation, les participants ont fait part de leur satisfaction quant au résultat obtenu sur une échelle de Likert en 7 points à l'aide d'un formulaire en ligne. En fin de formation, ils ont également rempli un questionnaire d'utilisabilité basé sur SUS [Brooke, 1996]. Le temps nécessaire à l'étude complète variait selon les participants, sans jamais dépasser deux heures. Pour les détails expérimentaux et les résultats complets, le lecteur se référera au matériel supplémentaire Table 6.1.

La Figure 7 montre que les participants ont trouvé l'outil facile à utiliser et qu'ils étaient très satisfaits des comportements qu'ils ont conçus. Le temps d'exécution moyen par tâche était compris entre 2 minutes 24 (pour la tâche la plus rapide) et 5 minutes 43 (pour la tâche la plus lente). Le questionnaire final d'utilisabilité montre un score moyen élevé de 80,6 percentile, ce qui donne à notre éditeur de champ d'interaction une note A- sur l'échelle de notation de Sauro-Lewis [Lewis and Sauro, 2018].

Globalement, l'étude montre que des utilisateurs novices peuvent facilement utiliser l'éditeur de champ d'interaction pour esquisser des interactions entre agents. Sachant que l'éditeur de champ d'interaction est une interface graphique simple qui n'est pas encore conçue pour un usage commercial, cette note montre une performance d'utilisabilité très élevée.

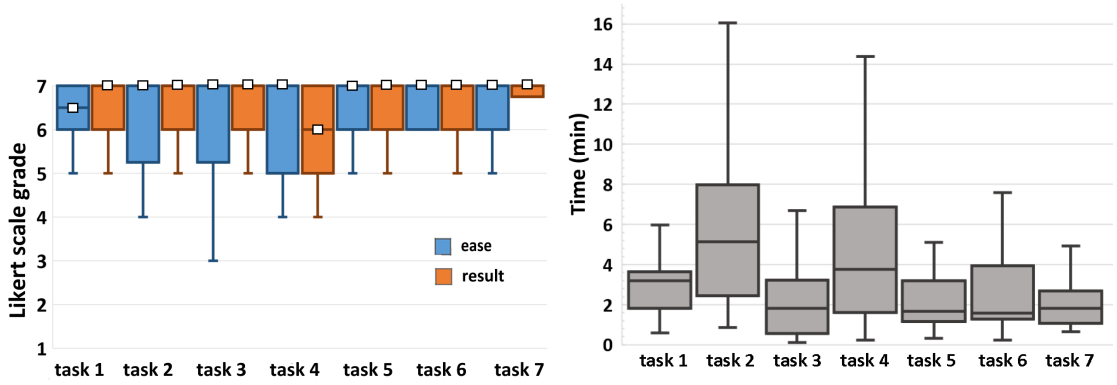


Figure 7 – Résultats de l'étude utilisateur, par tâche. Les diagrammes en boîte montrent les médianes, les intervalles inter-quartiles et les valeurs maximales/minimales (à l'exclusion des valeurs aberrantes). A gauche : notes des utilisateurs pour la facilité de conception (en bleu) et la satisfaction du résultat (en orange). A droite : temps de réalisation de chaque tâche (en minutes).

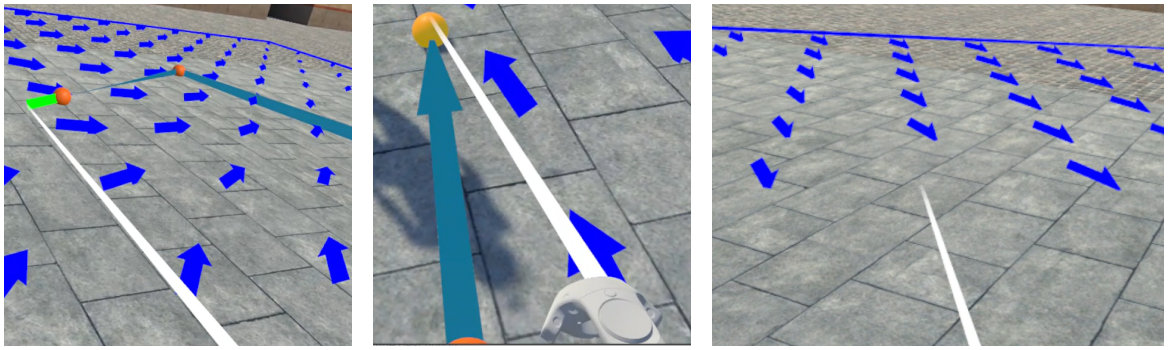
Réalité Virtuelle

La Réalité Virtuelle est souvent utilisée en simulation de foule pour valider les trajectoires générées par les modèles de la simulation et pour réaliser des études de perception, sur l'espace personnel par exemple. Nous pensons que les interactions personnelles subtiles, obtenues avec des champs d'interaction, pourraient être mieux jugées du point de vue d'un participant à ces interactions. Nous avons donc intégré la simulation avec champs d'interaction dans un environnement de réalité virtuelle sur Unity, l'utilisateur lui-même intervient en tant qu'acteur de la scène et peut expérimenter les interactions. La méthode du champ d'interaction est donc utilisable dans un scénario immersif. Il s'agit d'une étape importante pour évaluer le réalisme des résultats.

Nous avons présenter un scénario, une variation du scénario "cache-cache", au salon de Laval Virtual Europe en 2022 [Laval Virtual, 2022]. Les visiteurs ont donné des retours très positifs sur le réalisme de la réactivité des humains virtuels.

Dans le but de pouvoir tester l'interaction et de modifier les champs directement, nous avons intégré la possibilité de dessiner et de modifier les lignes de champ en Réalité Virtuelle suivant deux modes. Dans le premier mode, la boucle de simulation agit, les positions des agents, des champs d'interaction et du joueur sont mises à jour entre toutes les plateformes et l'animateur. Le deuxième mode est celui du dessin. La boucle de simulation s'arrête, les agents et les champs d'interaction sont fixes. L'utilisateur immergé en Réalité Virtuelle peut éditer des champs (au préalable dessinés sur l'interface 2D) (Figure 8(b)), dessiner sur de nouveaux champs (Figure 8(a)) et créer des zones nulles ("zéros") (Figure 8(c)). Avec cette façon de travailler, les premières observations montrent qu'il semble plus difficile d'être précis, en particulier à grande distance. Cependant, cette application donne une conscience plus réaliste de l'échelle de l'interaction et, nous pensons, permet d'être plus précis à courte distance. Une autre possibilité pour créer des lignes de champ a été explorée, elle consiste à utiliser directement la trajectoire de l'utilisateur

immergé. L'utilisateur appuie sur un bouton en mode dessin et sa trajectoire est enregistrée en plaçant une poignée à sa position toutes les secondes. Nous pensons que cette option innovante permet de dessiner les champs d'interaction de manière plus intuitive et peut même augmenter le réalisme de l'interaction.



(a) Création de lignes de champ **(b) Manipulation des poignées pour corriger le champ d'interaction** **(c) Suppression de vecteurs**

Fonctionnalités pour dessiner un champ d'interaction en Réalité Virtuelle, les lignes de champs sont en bleu clair, le champ de vecteurs en bleu foncé, les poignées de manipulation des courbes guides sont en jaune lorsque sélectionnées, en orange sinon.

Contributions

Les principales contributions de cette thèse sont les suivants:

— la méthode du champ d'interaction. C'est un moyen simple, efficace et flexible de modéliser de nouveaux types de comportements locaux dans les foules.

— une méthode de calcul des champs directement à partir des lignes de champ esquissées par l'utilisateur (interface graphique). Elle permet à l'utilisateur de dessiner de nouveaux comportements d'agents simplement et en peu de temps.

— les résultats d'une étude approfondie menée auprès des utilisateurs, qui confirme l'efficacité de l'éditeur CI pour l'esquisse rapide de comportements.

— extension de l' interface d'esquisse des champs d'interactions à la RV pour en explorer des avantages et des inconvénients. Cette interface permet notamment d'esquisser les champs en utilisant le propre mouvement de l'utilisateur.

Conclusion

Dans les simulations de foules basées sur les individus, le comportement de chaque agent est généralement décrit par des règles et des expressions mathématiques, modélisant un nombre restreint d'interactions.

Dans cette étude, par l'intermédiaire d'esquisses d'un comportement souhaité, nous proposons une méthode permettant à un utilisateur de générer de façon simple et intuitive une variété infinie de nouvelles interactions. Le couplage de cette méthode du champ d'interaction aux méthodes préexistantes de simulation de foule, se fait aisément et sans en augmenter sensiblement le coût. Cette méthode agit comme une perturbation locale d'une simulation de foule classique.

Un champ d'interaction spécifie les vitesses ou les orientations auxquelles sont soumis les agents sensibles à ce champ au voisinage d'un objet donné, source de ce champ. La valeur de ce champ ainsi que sa portée peuvent être fonction de paramètres, telle que la vitesse de la source, par exemple.

Grâce à une interface graphique, l'éditeur, qui permet de calculer les champs d'interaction directement à partir de croquis dessinés par l'utilisateur, on obtient un système efficace et intuitif pour générer de nouveaux comportements d'agents. Un champ d'interaction peut être réutilisé dans de multiple scénarios, et un comportement recherché peut être réalisé par la superposition de quelques champs d'interaction, ce qui fait du champ d'interaction une méthode générique. De plus, cette méthode se couple facilement aux modèles classiques connus pour leur efficacité, sans en affecter les performances en temps réel.

Cette nouvelle méthode de simulation a été intégrée dans un environnement de Réalité Virtuelle ou l'utilisateur peut s'immerger pour vérifier directement le réalisme et le rendu des comportements qu'il a créé dans la foule. Ceci doit permettre aussi d'améliorer le réalisme des interactions entre la foule et les divers acteurs virtuels ou réels.

Nous avons montré, dans quelques exemples, que des scénarios complexes sont simulés de façon très satisfaisante avec cette méthode à partir de quelques croquis simples. Une étude approfondie auprès d'utilisateurs non experts, révèle une prise en main facile de l'interface et une grande satisfaction quant à la qualité des résultats obtenus.

Plusieurs directions d'enrichissement peuvent être envisagées pour des travaux futurs.

Tous les champs d'interaction de cette étude ont été dessinés à la main dans l'éditeur. Une direction possible d'amélioration, serait de générer automatiquement les champs.

Cela semble possible pour certains types de comportements bien caractérisés (par exemple, se déplacer vers un point donné ou suivre un agent).

Un champ d'interaction pourrait également être généré automatiquement à partir d'enregistrements d'une foule réelle, en utilisant des trajectoires numérisées comme lignes de champ. Une autre direction intéressante pour les travaux futurs, serait de permettre la description de champ d'interaction à partir de fichiers textes, avec un lexique et une grammaire, qui pourraient être générés par l'interface à partir des lignes de champ, directement à la main ou par modification de fichiers existants.

Les champs d'interaction offrent un niveau sans précédent de contrôle créatif sur les comportements de conduite locale dans les foules. C'est une étape importante vers des simulations de foule totalement immersives où tous les agents se comportent de façon très humaine et où la foule répond de manière très réaliste aux actions de l'utilisateur.

TABLE OF CONTENTS

List of acronyms **21**

1 Introduction **23**
1 Macroscopic and Microscopic Crowd Simulation 24
2 Collective Movement and Expressiveness 25
3 Aims 26
4 Contributions 27
5 Thesis Structure 28

2 State of the Art **31**
1 Agent-Based Crowd Simulation 31
1.1 Collision Avoidance 33
Force-based models 33
Velocity-based models 33
Vision-based models 34
Data-driven models 35
1.2 Grouping 37
Small group 37
Large group 39
1.3 Following 41
1.4 Other Interactions 42
1.5 Cost function 43
2 Crowd Authoring 45
2.1 Parameter Tuning 45
Narrative 47
Mapping to human traits 48
2.2 Discrete Set of Templates 49
Motion patches 50
Deformable meshes 50
Crowd patches 51
2.3 Field Propelling 52
Crowd flow 52

Sketching field . . . 53
3 Sketch-Based Interface . . . 55
3.1 Crowd Sketching . . . 55
Free-hand global path sketching . . . 55
Group formation sketching . . . 56
3.2 Sketching Techniques for 3D Character Animation . . . 57
Sketching keyframe . . . 58
Motion path sketching . . . 59
3.3 User Evaluation . . . 61
4 Virtual Reality (VR) . . . 63
4.1 VR Sketching . . . 63
4.2 VR Crowd . . . 65
5 Summary and Objectives . . . 68

3 Interaction Fields: Overview and General Definitions 71
1 System Overview . . . 71
2 General Definitions . . . 72
Velocity interaction fields . . . 73
Orientation interaction fields . . . 73
Parametric interaction fields . . . 73
Keyframes and interpolation . . . 74
Relations between objects . . . 76

4 Sketching Interaction Fields 77
1 Main Elements of the IF Editor . . . 77
2 Converting a Sketch to an IF . . . 79
Interpolating between guide curves . . . 79
Computing the final IF . . . 81
3 Sketching Parametric IFs . . . 82
4 Discussion . . . 83

5 Implementation and Animation 85
1 Implementation . . . 85
1.1 Applying IFs During the Simulation . . . 85
1.2 Combining IFs With Other Simulation Components . . . 86
1.3 Crowd Simulation Framework and Settings . . . 87
2 Character Animation . . . 88
2.1 Coupling With Character Animation . . . 88

3D Integration . . . 88
2.2 Animating Characters Using Motion Matching . . . 90
2.3 *MxM* Plugin for Unity . . . 91
2.4 Library of Motion Capture . . . 94

6 Results and Evaluation **97**
1 Demonstration of Results . . . 97
1.1 Scenario 1: Hide and Seek . . . 97
Hide and seek . . . 98
Several hiders . . . 99
Scary giant . . . 99
1.2 Scenario 2: VIP in a Crowd . . . 101
1.3 Scenario 3: Crossroad . . . 102
1.4 Scenario 4: Museum . . . 103
1.5 Scenario 5: Mooses . . . 104
2 User Study . . . 106
2.1 Pilot Study . . . 106
2.2 Protocol . . . 107
2.3 Results . . . 108
2.4 Discussion . . . 111
3 Conclusion . . . 116

7 Sketching Interaction Fields in Virtual Reality: a Proof of Concept **117**
1 VR Implementation of IF . . . 118
Simulation mode . . . 119
Sketch mode . . . 119
2 Sketching in VR . . . 121
2.1 VR Interface . . . 121
2.2 Sketched Guide Curve . . . 123
2.3 Acted Guide Curve . . . 124
2.4 Zero Area in VR . . . 124
3 Observations . . . 125
4 User Study Proposal . . . 126
4.1 Non-Experts User Study . . . 127
Protocol . . . 127
Data and analysis . . . 128
4.2 Experts User Study . . . 129
Protocol . . . 129

Data and Analysis 130
5 Conclusion 131

8 General Conclusions and Perspectives **132**
1 Conclusion 132
2 Future Work 134

Bibliography **141**

List of figures **162**

List of tables **163**

LIST OF ACRONYMS

IF	Interaction Field
KNN	K-Nearest Neighbors
CGI	Computer Generated Imagery
PIF	Parametric Interaction Field
MoCap	Motion Capture
VR	Virtual Reality
AR	Augmented Reality
RVO	Reciprocal Velocity Obstacle
ORCA	Optimal Reciprocal Collision Avoidance
DL	Deep Learning
RNN	Recurrent Neural Networks
ttca	Time To Closest Approach
ttc	Time To Collision
dca	Distance To Closest Approach
ISO	International Organization for Standardization
SUS	System Usability Scale
GOMS	Goals, Operators, Methods and Selection rules model
FVF	Force Vector Field
CMMs	Crowd Motion Models
WIMP	Windows, Icons, Menus, Pointing devices
HMD	Head Mounted Display
HCI	Human Computer Interaction

CHAPTER 1

INTRODUCTION

Many applications of computer graphics, such as cinema, video games, or Virtual Reality require the design of situations in which multiple virtual humans are involved. In these cases, the realism of the nonverbal behaviors of these virtual people is key to a compelling product. In movies, more and more people or creatures are computer generated. In video games, the player must interact with these virtual humans. In both applications, characters respond appropriately to their environment by exhibiting various behaviors. Behaviors can be defined as the way one acts or conducts oneself, especially toward others. Realistic behaviors are expressive and reactive, when a user is immersed in the simulation (e.g., video games, VR), and enable the transmission of information through verbal communication, i.e., the exchange of information through speech or spoken words, and nonverbal communication, i.e., expression through body language (e.g., gestures, facial expressions, personal space). In this thesis, we focus exclusively on reproducing the nonverbal behavior of virtual characters as close to reality as possible. In sociology, Robert E. Park [1967] defined collective behavior as "the behavior of individuals under the influence of an impulse that is common and collective, an impulse, in other words, that is the result of social interaction". In a simulation, the goal is to model convincing social interactions between agents and describe group-level behavior. The realism of these behaviors is what convincingly populates a virtual world, because a realistic world (except perhaps a post-apocalyptic one) consists of many living beings. Therefore, it is necessary to populate such virtual environments by simulating the behaviors of large numbers of virtual humans. However, simulating such behaviors requires taking into account interactions between characters, which may be of different natures. They can be diverse and complex, depending on the scenario. For example, a street can be the scene of many different types of interactions: pedestrians gathering around a street show, people window shopping, a street seller, a mailman... In a football game, the players first enter the field then go to their position and play accordingly, the viewers will watch the game, the referee will follow the ball from afar... In a playground, children can play many games: green light, red light, tag, hide and seek, teachers have to watch over them and ultimately the children get in order to go back to class. Depending on these scenarios, people's global behaviors differ. Moreover, the social context or the local situation of a smaller number of people also influences their behavior: a group, children, family members, players of a game... These numerous factors make simulating a variety of behaviors, complex, as we need many different models of interactions. The aim of this work is to offer possibilities to simulate these numerous complex local interactions.

As previously said, this work focuses on collective nonverbal behavior. We do not deal with the

appearance of those characters but with the motion and the animation. Nonverbal behavior is composed of the trajectory, the body animation and the facial expression of the virtual character. This thesis first studies the trajectories of characters and then couples them with already existing animation systems. Crowd simulation gives ways of simulating the trajectories of a large number of virtual humans in a virtual world. Therefore different ways of simulating crowds have been proposed over the years, and can be divided into two main categories, namely macroscopic and microscopic.

1 Macroscopic and Microscopic Crowd Simulation

A crowd is a confluence of a large number of people in one place. In crowd simulation, the goal is to reproduce, through computation, the behavior of the crowd, which is characterized by the movement of people through space over time. *Macroscopic* and *Microscopic* models are the two ways in which a crowd can be modeled. In these models, the virtual humans whose movement is being simulated are called "agents". A crowd can be modeled as one object that moves, and the model is then called *macroscopic* [Degond et al., 2010; Maury et al., 2010; Treuille et al., 2006]. In this case, the crowd can be considered as a whole material such as a fluid that has its own flow, as shown in Figure 1.1(a). The flow can be unidirectional if the fluid flows in one direction, or can be multidirectional if there are several flows with opposite directions, e.g., people not following the same path. But the fluidity of this crowd is in fact made up of different particles, the humans, each of them having its own motion, speed, acceleration... Looking at these particles one by one and working on the properties of each virtual human is the definition of *microscopic* or *agent-based* approaches.

(a) Macroscopic Crowd **(b) Microscopic Crowd**

Figure 1.1 – Crowd crossing Shibuya Street, photo by Sei F – Wikimedia. (a) Macroscopic model: The crowd is seen as a stream in different directions (blue and orange arrows). (b) Microscopic model: same image zoomed in, each individual is simulated individually with local interactions such as collision avoidance in green and group formation in blue.

In agent-based approaches, the trajectory of a virtual agent can be defined as a combination of several processes. Path planning, which defines the global trajectory a virtual human should take to reach its destination, and local interactions, which describe how the agent reacts when it encounters elements of the environment (other virtual agents or obstacles) in its path. These local interactions can be many and varied: collision avoidance, grouping and following are three examples commonly found in the literature (see Figure 1.1(b)).

As previously mentioned, this thesis focuses on simulating the behaviors of virtual agents. To better simulate the virtual world and make it more realistic, it is interesting to model subtle and diverse interactions between agents. As mentioned above, in agent-based simulation, local interactions define how an agent reacts to an object or another agent, and can be directly assimilated to simulated behaviors. Those interactions can be of collective nature and depend on other agents of the simulation, as it is the case when group motions are modeled in crowds (see Section 1), which explains why we focus on microscopic crowd simulation. A lot of information about the social context can also be conveyed by collective behaviors in real life, which is why we believe that it is interesting to focus on collective behaviors' expressivity.

2 Collective Movement and Expressiveness

In physics, the principle of locality states that an object is directly and only influenced by its immediate surroundings. For Ozgur [2010], local interactions refer to social and economic phenomena where individuals' choices are influenced by the choices of others who are "close" to them socially or geographically. This definition fits the context of microscopic crowd simulation as it is defined as a change in the trajectory of an agent at the local level relative to other objects (e.g. agents, environment, obstacles). It can be summarized as the mutual influence of the agents via their trajectories.

However, psychological and social researchers have shown that these local interactions can actually give context to a scene. The way people move in a group, one in relation to the other, brings a lot of nonverbal information about the social context. According to Wilder [1986], "persons organize their social environment by dividing themselves and others into groups". Three categories are then described:

1. there is no relationship between the perceiver and the group,
2. the perceiver is a member of the group (in the group), and
3. the perceiver is not a member of the group and compares with his own group (in group/out group).

This categorization implies notions of similarity, homogeneity and differences and would enable individuals to simplify their social environment and predict future social behavior. Furthermore, previous work on conversational groups [Kendon, 1990] has shown that the relative position of the members of the group, which can be considered as an in-group situation, is such that each member of the group has a similar shared space with direct and exclusive access. Kendon refers to the F-formation system that describes this "spatial-orientational behavior", which can be dynamically adjusted to include another person in the

group. Recently, Cafaro et al. [2016] used closely related concepts to design believable virtual agents in small conversational groups (static condition) that exhibit nonverbal behavior. The authors manipulated both the relative position of each individual (group formation) and interpersonal attitude (friendly vs. unfriendly). In applications where a user is immersed in the virtual environment, the (collective) behavior of these virtual humans must be realistic to improve the user's immersion; i.e., increasing its perception of being physically present in a nonphysical world. As part of realism, expressive behavior appears to be a crucial aspect. For example, Slater et al. [1999] showed that the expressive behavior of an audience in Virtual Reality had a direct impact on the speaker's performances and perception of themselves. More recently Bönsch et al. [2018] conducted experiments to study the personal space in Virtual Reality, a protective area around oneself leading to discomfort when invaded. They showed that the displayed emotion of the virtual character directly impacted the size of the space, participants keeping bigger personal space when encountering angry virtual characters. The category of 1-to-n nonverbal communication scenario considers the interaction between *1* user and several *n* virtual humans. In terms of reactions a large palette of body motions may convey to the user the fact that its presence among virtual humans has triggered events, such as making eye contact, turning bodies toward the user, moving toward the user, moving away from the user, leaving his room to join, etc. The intensity of such a reaction is also adaptable, to vary the information conveyed to the viewer, while synchrony and propagation of reactions will convey the reaction collectiveness. Moreover, we note that the previous studies have all been designed for immersive scenarios, since it enables users to easily interact with the virtual characters and analyze the responses of the users to the interactions. Therefore, it is of interest for this work to deal with applications in VR.

In line with these studies, we would like to extend the design of interactive and expressive virtual humans to dynamic situations. In particular, we would like to modulate the nonverbal expressiveness that virtual humans convey through their collective movements, which occur when individuals sharing an environment interact with each other. In this context, the simulation of an expressive and collective movement consists in defining the respective position of each virtual human in time in order to convey a certain social context. The modulation of the expressiveness of the group is achieved through its movement and its final configuration in relation to that of the user.

3 Aims

Simulating crowds is a primary component of creating realistic and lively virtual worlds. The virtual members of the crowds populating such worlds undergo collective behaviors that are important to simulate to describe scene situations. This thesis aims to investigate ways of intuitively designing and simulating such behaviors.

Most crowd simulation techniques are *agent-based* in that they simulate each character as an individual intelligent agent. To steer each agent through the environment in interaction with other agents, many algorithms (detailed Section 1) have been developed for specific purposes such as path planning, colli-

sion avoidance, and grouping. Although the resulting models are highly successful, it is difficult to adapt them so that the agents display new kinds of behaviors for which the algorithms were not designed, such as hiding behind an agent or blocking its path. Even though a designer can influence agents' overall paths and tune simulation parameters to change the properties of a specific algorithm, they cannot easily let agents interact in entirely different ways, such as making an agent circle around another agent or hide behind an obstacle. Furthermore, parameter tuning and scenario-specific scripting can be time-consuming. In both cases, animation designers are in charge of controlling the motion of those many characters. To this end, they have to manipulate crowd simulators and their many parameters to generate the desired behaviors, depending on the scenario they have in mind.

In response to these problems, this research is aimed at simplifying the design of local steering behaviors in crowds, by letting users sketch how agents should move in relation to other agents, obstacles, or the environment, enabling us to create a variety of behaviors. Our central concept is to define and use an *Interaction Field* (IF) that defines the velocities or orientations that agents should use around a particular source, such as an obstacle or another agent. An IF can also be made *parametric* to change dynamically according to simulation parameters, such as the current speed of an agent. We also present an editor in which users can sketch IFs, enabling them to quickly and intuitively create a wide variety of new types of agent behavior. Furthermore, IFs can be combined with other crowd simulation techniques so that users can focus on sketching only those behaviors for which traditional algorithms do not suffice. This editor was also implemented in VR to enable users to sketch IFs while being immersed in the scene, easing the trial and error process.

Previous research has led to several other methods for artistically modifying the behavior of a crowd, [Kwon et al., 2008; Patil et al., 2011; Ulicny et al., 2004] but these methods focus on other aspects, such as controlling simulation parameters or editing global trajectories. To the best of our knowledge, we present the first method that enables users to intuitively sketch *local interactions*, i.e., to sketch how agents should move relatively to other (moving) obstacles or agents.

By generating IFs from sketches, users can quickly design new behaviors that would otherwise require laborious programming and parameter tuning.

4 Contributions

Our main contribution is the concept of Interaction Field, which is a simple technique to simulate local interactions between crowd characters. Apart from the theoretical framework, we also proposed an implementation to enable users to sketch Interaction Fields easily and intuitively. We also demonstrate that our animation method thus constituted enables us to significantly extend the variety of scenarios that can be simulated, and evaluated through a user study, to drastically speed up the design time of new types of scenarios. Finally, the framework was translated onto a prototype VR interface to investigate the advantages of sketching while experiencing the interactions.

In short, the main *contributions* of this research are the following:

— We present *Interaction Fields* (IFs) as a simple yet effective way to model new kinds of steering behaviors in crowds.

— We present a novel way to compute IFs based on *user sketches*. This results in an IF editor that allows designers to draw new agent behaviors in a small amount of time.

— We show the capabilities of IFs in various *scenarios* that would be difficult to simulate using traditional models alone.

— We present the results of a thorough *user study* that confirms the efficacy of the IF editor for fast behavior sketching.

— We extend our original sketching interface in VR to explore its advantages and disadvantages. This interface especially enables to sketch the fields using the user's own motion.

This work led to a poster [1] published at ACM Motion, Interaction and Games 2021 that received the third price poster and to a full paper [2] presented at Eurographics 2022. At this occasion, it received the Günter Enderle Honorable Howard. This work was also part of the scenarios presented by *Inria* at Laval Virtual 2022 and part of a VR experience 'Delirious Departures' exposed at the immersive pavilion of Siggraph 2022.

5 Thesis Structure

This thesis is organized as follows:

— Chapter 2 describes the state of the art related to the main concepts of this thesis. We will first focus on the local interactions already simulated in the microscopic crowd simulation literature, and show that only few of them are considered. We will then explore how we could add variety to those local interactions through authoring crowd methods. We noticed that those methods never focus on local interactions, nevertheless we see the potential of approaches using sketching. We further investigate sketching interfaces in animation to better analyze if sketching could be an interesting approach for local interactions. As our work has applications in VR, we will also study the sketching interfaces existing in VR and the uses of VR in crowd simulation. Finally we can take out of the related work, that sketching local interactions is of interest and was never tackled before.

1. A. Colas, W. van Toll, K. Zibrek, L. Hoyet, A.-H. Olivier, and J. Pettré, "Interaction Fields: Intuitive Sketch-based Steering Behaviors for Crowd Simulation," Computer Graphics Forum, pp. 1–14, Apr. 2022. [Online]. Available: https://hal.inria.fr/hal-03642462

2. A. Colas, W. van Toll, L. Hoyet, C. Pacchierotti, M. Christie, K. Zibrek, A.-H. Olivier, and J. Pettré, "Interaction Fields: Sketching Collective Behaviours," MIG 2020: Motion, Interaction, and Games, Oct. 2020, poster. [Online]. Available: https://hal.inria.fr/hal-02969013

— We present in Chapter 3 an overview of our approach and define the building block of our method, *Interaction Fields*. IFs enable to easily and intuitively sketch interactions. The main mechanisms which build upon this building block will be then defined to design more complex interactions.

— In Chapter 4, we present how IFs can be sketched using a dedicated Graphical Interface. This interface as well as the tools to manipulate the diverse IFs functions will be described in this section.

— In Chapter 5, we explain how, once IFs have been sketched, they can be used and combined to drive the motion of agents in a simulation. With this, we obtain trajectories that can be imported into a 3D scene and animated. The technique to animate those trajectories, using full-body character animation, will be explained in the second part of this chapter.

— In Chapter 6 we present and evaluate the results obtained using our IF framework. First, we highlight the possibilities of IFs by presenting a number of complex simulated scenarios in 2D and 3D, which illustrate the various mechanisms making use of IFs. Second, we present a user study which we carried out to evaluate the usability of IFs.

— Chapter 7 investigates the potential of sketching IFs using Virtual Reality. To this end, we describe a new VR interface as well as the new functions available to IF in this platform. The second part of this chapter discusses the advantages and perspectives of VR and presents the protocols that should be performed to evaluate the interface.

— In Chapter 8, we conclude and summarize the work of this thesis and then discuss the possible future work directions that stem from this work.

CHAPTER 2

STATE OF THE ART

This chapter is dedicated to the analysis of the previous work related to local interaction design for virtual scenarios. This review of the state of the art will be divided in four sections.

— First, we analyze the literature to list the local interactions that can already be simulated in virtual crowds, with the objective of verifying the need or not for other types of local interactions.

— The list of interaction being restricted, we wonder how we could create new local interactions. We will analyse the state of the art regarding crowd authoring. In crowd authoring, we will focus on works that already attempt to give control to the users, letting them adjust crowd simulation's parameters to their need.

— Among those works, sketching-based techniques seem of interest, as they are intuitive and enable quick design compared to other techniques. We will dedicate a section of the state of the art to sketching methods for crowd simulation authoring and for 3D body animation, to see if some work already focuses on sketching collective human behaviors.

— Since we also want to design behaviors for immersive scenarios, we finally analyze the literature about Virtual Reality, in particular in relation with crowd simulation and how sketch-based techniques can be translated into VR.

1 Agent-Based Crowd Simulation

This research aims at providing intuitive ways of shaping interactions between a collective of individuals (the crowd) by non-expert users. To do this, we need to consider the movement of each individual separately to simulate detailed interactions. This is why we will start the state of the art by considering agent-based approaches, with a focus on the type of local interactions typically considered.

Microscopic or agent-based approaches simulate each human as an intelligent agent that navigates in an environment. This process is layered in several levels, shown in Figure 2.1. The main task of each agent is to reach a goal, by first planning a path that provides solutions to achieve its goal and then following that path: this is *global path planning*. During navigation, the agent may encounter and interact with other agents or obstacles. In this case, the velocity of each agent must be updated at a specific frequency so that each agent reaches its destination while reacting appropriately to neighboring

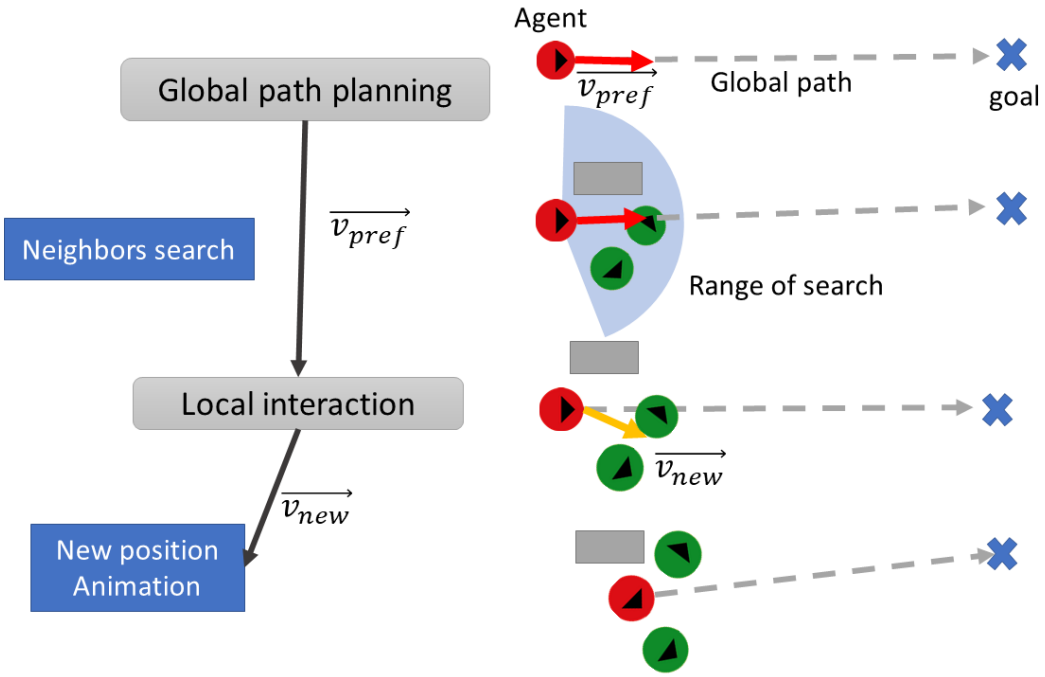


Figure 2.1 – Microscopic model processes by levels: High level is global path planning and low level is local interaction. For each frame, the global path gives the preferred velocity $\mathbf{v}_{pref}$. After searching for neighbors, local interactions are defined and applied in the form of unbreakable rules. $\mathbf{v}_{new}$ is the closest match to $\mathbf{v}_{pref}$ that complies with these rules. Applying $\mathbf{v}_{new}$ to each agent yields the new position.

agents and obstacles. Compliance with these local rules is the other level of this process, called *local interaction*.

An agent-based simulation consists of several time steps called 'frames'. The computational process in each frame revolves around different steps for each agent in the simulation, illustrated in Figure 2.1. The first step is to compute a preferred velocity $\mathbf{v}_{pref}$ that sends the agent towards its target position or progresses along its global path, then neighboring obstacles and agents within a defined range are searched. Based on the neighbors information (i.e. velocity, acceleration), a new velocity $\mathbf{v}_{new}$ is computed, that is close to the preferred velocity but conforms to the local rules based on the neighbors information. Finally, the agent's position is updated according to this new velocity. There are many frameworks that use this principle to simulate large crowds in real time [Curtis et al., 2016; Kielar et al., 2016; Pelechano et al., 2007; van Toll et al., 2015]. The first need of microscopic models is to ensure that agents do not collide with each other. For this reason, collision avoidance is the most common local interaction found in the literature. Depending on the situation, agents can have many other local interactions besides collision avoidance: being part of a group or following another agent are examples that are also commonly modeled in the literature.

This section focuses on the various simulation models for local interactions that can be found in the

literature [1].

1.1 Collision Avoidance

The *collision avoidance* task has received much attention from authors interested in modeling local interactions within a crowd. In this subsection we will present the main approaches developed.

Force-based models

Helbing and Molnar [1995] proposed the best known model of collision avoidance in crowds of their time. In their model, each agent answers to forces, that can be attractive, steering it towards its goal position, or repulsive, from the obstacles and other agents. In this case, the repulsive force enables collision avoidance. Later, Karamouzas et al. [2009] also modeled collision avoidance using forces, but added the concept of time and thus prediction of collisions. The repulsive force exerted on an agent is a simple function of the predicted time to collision (Figure 2.2(a)) with another agent or obstacle. In parallel with this model, Zanlungo et al. [2011] also considered time. However, their model used the overall time to collision to scale to all neighbors, while Karamouzas et al. treated each neighbor's time to collision independently and ignored neighbors where no collision was expected. Meanwhile, Kang and Kim [2014] introduced the "universal power law". Here, a different principle applies, which states that agents should try to minimize their energy spent on interactions, knowing than the farther away the object is, the lower the chance that the agents will collide with it. This method has received a lot of attention because, at the time of publication, it was lighter in computation and simpler to design than the velocity-based methods (described in the following section), and gave better results. Nevertheless, velocity-based approaches should be more robust and support more scenarios than force-based ones.

Velocity-based models

After force-based models, velocity-based models emerged to simulate collision avoidance. They enable agents to analyze all possible velocities rather than just the effects of a single velocity. The agent examines each of its possible velocity before deciding on the best outcome velocity by predicting the consequences of each one. While the computation cost is higher for force-based techniques, it can handle more difficult scenarios and better match human behavior. All velocities in a velocity space are evaluated according to certain criteria (e.g. avoiding collision). Then, an agent has to choose in this space the optimal velocity. Velocity-based techniques lie in the definition of a cost function that assigns a scalar cost to each velocity in the space: a lower cost indicates a velocity is a 'more attractive' option to choose. Paris et al. [2007] were the first to present a collision-free velocity-based algorithm, by first excluding the velocity inducing collision and then using a cost function to select among the remaining options. One

1. For an overview of local interaction models proposed by the crowd simulation community over the past 12 years, see the review by van Toll and Pettré [2021].

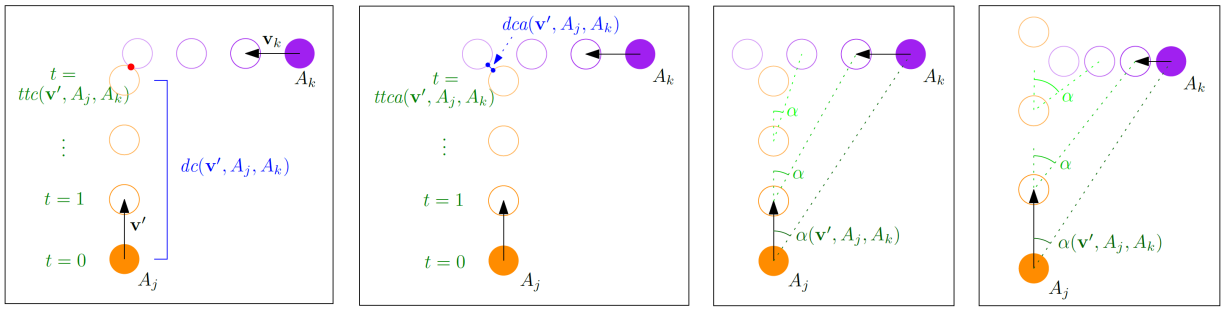
year later van Den Berg et al. [2008] published the method RVO (Reciprocal Velocity Obstacle), which assumes that all agents will spend an equal amount of energy to avoid each other. They then improved the method by proposing ORCA (Optimal Reciprocal Collision Avoidance) [van den Berg et al., 2011] that transforms the mathematical definition of the collision avoidance problem, so that an agent can compute its optimal velocity without sampling (which was not the case for RVO) making ORCA computationally more efficient and robust. PLEdestrians by Guy et al. [2010] combines the ORCA concept with a cost function based on the concept of energy minimization. While RVO and ORCA were first applied to robots, PLEdestrian focuses on minimizing the effort of a pedestrian, making the resulting trajectory more human-like. Karamouzas and Overmars [2010] introduced a novel approach that fully exploits cost functions in velocity space. In this method, agents test several speeds and directions via regular sampling, and a cost is attributed to each sample velocity according to its difference with the current velocity, the deviation from the prefered velocity and the time to collision (Figure 2.2(a)). To define the specific equations and parameters of the cost function, the authors used experimental results of real human collision avoidance. Moussaid et al. [2011] suggested a simplified version of this method one year after, which is less computationally demanding and has fewer parameters to tune but do not use real trajectory data.

Vision-based models

Vision-based models use the human-like metaphor that collision avoidance is foremost handled by vision. First, visually-driven steering algorithms define behaviors based on variables close to what people visually perceive without knowing the entire environment geometry. The idea is to replicate directly how real human vision works and to remove all global coordinate information. The techniques aim at replicating more realistic human trajectories by making agents react to what they perceive (e.g. an object moving through their field of view). This type of method, which we refer to as visually driven steering, was developed by Huang et al. [2006], who combined modeling with real-world experiments. In Park et al.'s work [2013], to avoid colliding with a neighbor, an agent will use the evolution over time of the 'bearing angle', which is the angle between its heading direction and the line connecting the agent and the possibly colliding neighbor as shown Figure 2.2(c) and Figure 2.2(d).

Another category of vision-based approaches goes as far as simulating a human retina, by projecting visual information on an image, called virtual retina, for each agent. The retina-based algorithm are hence computationally more expensive than the previous methods but they represent human perception more accurately. Ondřej et al. [2010] projected every visible objects of the environment onto the virtual retina and predicted collision for non empty pixel. It determined the pixel that has the highest risk of collision to make the agent avoid the threats, using the time to closest approach (the time at which the distance between two agents is the smallest) described in Figure 2.2(b) as well as the bearing angle (Figure 2.2(c) and Figure 2.2(d)). Ondřej et al.'s method is reactive and agents only change their velocity when a future collision is predicted. According to the future crossing distance, agents will react or not based on experimental data. Dutra et al. [2017] applied the idea of cost function but based on the information given

by the virtual retina. In those two last methods, the non empty pixel is then tracked back to the object, agent or obstacle, from which the simulation's properties, such as the velocity or time to collision (see Figure 2.2(a)), are directly used. In their work, Lopez et al. [2019] not only analyze the environment by the information of the virtual retina but also create a dense optical flow generated by the succession of the perceived virtual retina. The apparent movement of pixels in an image is known as optical flow, and the term "dense" refers to the estimation of this apparent motion for each individual pixel. The difference between the images enables to estimate the apparent motion of each pixel to then deduce the velocity of the objects. The agents ultimately navigate, similarly to Dutra et al.'s approach [2017], using a cost function based solely on perceived data. Vision-based and more precisely retina-based algorithms aim at simulating how human vision work more realistically, shifting the focus of research from using reasoning about objects in world coordinates to using visual information. Vision-based models use gradient-based steering, which appears to be more constrained than velocity-based models, which investigated the entire space of potential velocities. To close the gap between velocity-based and vision-based approaches, several parts of retina-based algorithms are also applicable to the "traditional" domain of 2D simulation.



(a) Time To collision (ttc)

(b) Time to Closest Approach (ttca) and Distance to Closest Approach (dca)

(c) Bearing angle α (decreasing)

(d) Bearing angle α (increasing)

Figure 2.2 – Overview of collision-prediction concepts between two agents A_j (in orange) and A_k (in purple). $\mathbf{v}'$ is a hypothetical velocity for A_j, and $\mathbf{v}_k$ is the current observed velocity of A_k. (a) The time and distance to collision. In this example, the agents collide. (b) The time and distance to closest approach. In this example, the agents do not collide. (c) α, bearing angle that decreases over time. (d) α, bearing angle that increases over time. Based on the current speed and respective position, time to closest approach and distance of closest approach can be computed to identify whether there will be a collision (a) or not (b). Image taken from van Toll and Pettré's survey [2021].

Data-driven models

The last type of models are data-driven methods. Those models, instead of applying mathematical rules to define the behaviors of the agents, aim at replicating some input data (e.g. trajectories contracted from videos), which can be of any kind of local interaction.without explicitly defining the behavioral

rules themselves. Hence, data-driven methods should be theoretically able to replicate more subtle and specific behaviors that are difficult to define with rules (force, cost function...). Several works concentrate on data-driven models for local navigation that use databases and deep or reinforcement learning. In early works, Lerner et al. [2007] use a database of examples of pedestrian interactions, and at run time, a search is made to find the most similar data set and copy the data behavior onto the simulation. These solutions efficiently rely a lot on the database. If the database does not correspond to a particular scenario, suitable behaviors will not be found, and if the data is too large, searching will be too time-consuming. To address these concerns, Charalambous et al. [2014] clustered database entries, reducing search time. Zhao et al. [2013] used trained artificial neural networks (ANN) to determine which cluster best matches a situation. Another particularity of this work is that it used simulated data (here using ORCA [van den Berg et al., 2011]). Boatright et al. [2015] used machine learning to learn a behavioral policy from data generated by a crowd-simulation algorithm that replicated the initial algorithm. Thus, at runtime, they only searched a behavioral model that was previously learnt from all entries rather than a database of entries. Ren et al. [2021] presented a method that uses records containing velocity as a database. For each frame, the velocity that optimizes the safest criteria modeled by an energy function is determined. This method is similar to a velocity-based algorithm that uses a cost function, but the input data define the possible velocities. A common problem with approaches that use a database of crowd actions is that the selected input data set is used without much modification, requiring appropriate data sets for all conceivable circumstances. A recent trend to address this problem is to use the generalization capabilities of Deep Learning (DL) to create a more abstract model of agent behavior that can theoretically be applied to new scenarios. RNN (recurrent neural network) is a data-driven DL technique that can be used in our case to estimate an agent's next position(s) based on its neighbors and its own past motion. Most RNN-based models come from the field of computer vision, where the goal is to track or predict human motion rather than simulate crowds. However, these models can also be used for our purpose. Once trained, RNNs can immediately serve as an agent navigation model in a crowd simulation. The navigation itself is computationally intensive and can be used for many agents in real time. For a complete overview of these methods, see the survey by Rudenko et al. [2020]. The Social-LSTM approach of Alahi et al. [2016] used LSTMs (Long Short-Term Memory), a type of RNN that can learn both long-term and short-term patterns in data. Using several popular real-world datasets, the authors show that their LSTM-based method is good at predicting the future course of people's input trajectories, and is (understandably) more accurate than force-based methods that do not rely on that input data. One drawback of LSTMs is that they only ever make a single prediction, which is roughly equivalent to the average behavior observed in the input samples. Gupta et al. [2018] and Amirian et al. [2019] have recently presented GAN -based methods for trajectory prediction, and there are small differences between them that are too subtle to discuss here. Compared to a pure LSTM approach, these methods can produce a wider variety of trajectories with the same input. On the other hand, the training process of GANs is time-consuming and difficult to control. Finally, some authors have focused on using Deep Neural Network (DNN), where

the state description can be raw data (e.g., the relative positions and velocities of neighbors) instead of a customized summary. Lee et al. [2018] implemented such a system for crowd simulation by training their DNN using trajectories generated by ORCA. After training, the DNN can compute the time to goal for any candidate velocity in a given state. Haworth et al. [2020] investigated a different learning mechanism and applied it to footstep-based navigation. Although their state descriptions and reward functions are slightly different, the common conclusion is that deep RL with a simple reward function can outperform traditional methods.

DNN systems also provide more convincing results than the classical crowd algorithms in unpredictable scenarios. However, a repeatedly highlighted drawback of DL is that the trained model is a black box that no longer has intuitive meaning in the original domain. Moreover, it is not intuitive why such a model behaves the way it does, and it is impossible to make any adjustments to improve the behavior in individual cases. In addition, combining DL models to achieve combined effects may be challenging, unlike, for example, combining collision avoidance with group behavior in the traditional way. In order to improve the DL-based crowd simulation, researchers need to find ways to interpret the generated models. On the other hand, DL can be a powerful tool to simulate behaviors that cannot be captured by rules alone.

1.2 Grouping

The different approaches described in the last section: force-based, velocity-based, vision-based, and data-driven, can be used to model other kinds of interactions. This section focuses on the various methods that model groups of agents using the previous approaches.

The formation and motion of small groups, including social relations between the agents of groups, is standard behavior in a crowd. A group typically shares a common global path, and the entirety of the group behavior is the local interaction between the members. Reynolds [1987] was among the first to propose and implement a steering algorithm to simulate flocks of agents, using forces and velocity matching to make the members stick together.

Small group

A group can be defined as two or more people who interact for a shared goal. In psychology, a group of 3 to around 12 individuals is considered a small group (because two people would be a pair or dyad). Group cohesion is the extent to which group members are attracted to the group and its goals. To simulate group cohesion, force-based models are a good option because an additional force can be defined to attract the agents towards the center of the group [Pedica and Vilhjálmsson, 2008], towards another agent [Braun et al., 2003] or both [Jan and Traum, 2007]. In the survey of Nicolas and Hafinaz [2021], these models are divided into grid-based, where each agent's place is defined by a grid cell and continuous models. In both cases, group cohesion can be modeled by adding radial cohesive forces [Li

et al., 2017; Liu et al., 2018]. This attracting term can either be replaced or supplemented with a term that promotes the alignment of the individual velocities of the group members [Chen et al., 2020; Qiu and Hu, 2010]. Moussaïd et al. [2010] collected empirical data with video recordings of public areas and noticed that a crowd mainly comprises small groups, highlighting the need to represent them in crowd simulation. They also find that groups at low density (up to 3 members) tend to walk side by side and form a line perpendicular to the walking direction, the so-called line abreast pattern. As the number of members increases, the formation turns into a V-like (or U-like) pattern, with the middle individual lagging a bit. When the group density is high, the members form a river-like pattern and lane formation, with a leader now in front of the other group members. Group formation results from a tradeoff between walking faster and facilitating social exchange. Federici et al. [2012] also used video footage to confirm the same hypothesis and the group patterns visible in Figure 2.3. In addition, Xi et al. [2014] studied video footage and conducted laboratory experiments under low and moderate density conditions to identify group shape patterns of five members or more. Their studies showed that, for groups size between 5 and 9, members tend to redivide in multiple smaller groups adopting the previously mentioned pattern. According to Costa [2010], when navigating through a crowd, a group often forms three dynamic group formation types that facilitate communication and cohesion among members. They account for differences between groups based on the gender of the members. For example, male groups of two (dyads) or three (triads) tend to walk more abreast (see Figure 2.3) than female groups.

Knowledge of these models is a crucial factor in designing dynamic group models, which are mainly concerned with ensuring group cohesion (or, if necessary, separation) in the crowd.

To better simulate the group pattern, some works focused more on the previously described group shape. In their extension of the social forces model, Moussaïd et al. [2010] added a 'visual' force to minimize the angle between the velocity and the direction of gaze toward the center of mass of all other group members. This addition successfully captured the reduction in velocity with increasing small group size and the experimentally observed deviations from walking side by side in small groups, making groups adopt the observed patterns. Lavergne et al. [2019] also integrated vision-based information into a complex particle-based model, driving the agent toward the visible group members. Huang et al. [2018] proposed a social group force model derived from Helbing's social force model [1995] to simulate group behavior, focusing on group avoidance and subgroups simulation. In this work, subgroups try to maintain a distance between themselves. When the distance between subgroups is significant, the force becomes attractive, and the rear subgroups speed up to catch up with the front subgroups. In the opposite case, the force becomes repulsive to prevent collision between the subgroups.

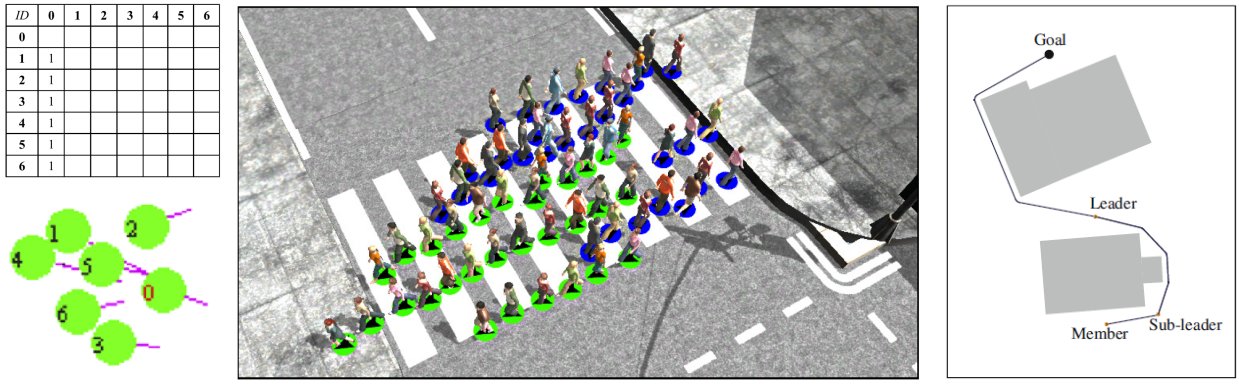
Other authors choose to work on the group space. Yang and Peters [2019] defined a social dynamic group space that matches the three dynamic formations or the transition between them. The space can model static and dynamic group formations in a socially acceptable manner. The social-aware space is modeled in order to produce a speed map used by a fast marching method to find the fastest path to the goal position. Work from Rojas and Yang [2013; 2014a], also define space but as discrete slots hinge

Figure 2.3 – Typical patterns of walking groups (from the left to the right: line-abreast, V-like, river-like pattern) [2012].

connected that agents must follow. To evaluate the realism of the simulated group behavior, they used VR to assess presence and degree of realism using a questionnaire about the realism of the group formation, and the feeling of inclusiveness. Karamouzas and Overmars [2012] let groups switch formations dynamically to better adapt the environment (avoid collision to obstacles or others agents) and then compute a prefered velocity for each agent so that this optimal formation is maintained.

Large group

For larger groups, Musse and Thalman [2001] and Qiu and Hiu [2010] describe reactions and behaviors at both the agent level and the group level. Musse and Thalman [2001] defined a crowd model represented through a hierarchical architecture where the minor entity to be treated consists of groups. The structure can be defined by scripting, and agents can dynamically belong to a group according to their relationships with groups, their intentions, belief, or domination. For example, when one agent's domination is high, it means it is the leader of the group and is the only one to know the past trajectory of the group. The flocking motion of the group is then ensured by rules over the agents' properties: all group members have the same velocity and list of goals. In their work, Qiu and Hiu [2010] modeled the relationships within (intra) and between groups (inter) with weight matrices. The intra-group matrices define, for a value $x_{i,j}$, the dependence between agents i and j or, in the case of the inter-group matrix, between group i and group j. An example of an intra-group matrix is visible in Figure 2.4(a). Finally, Ren et al. [2017] designed a cost function in velocity space for group formation. They used the concept of velocity obstacle, a set of prohibited velocities that would lead to a collision with an obstacle, and introduced the counter concept of velocity connection: a set of encouraged velocities that ensure close grouping. By combining velocities obstacle and velocities connection in the velocity space, the method's cost function accounts for collision avoidance and group behavior.



(a) Intra-group matrix **(b) Group formation crossing** **(c) Sub-groups**

Figure 2.4 – (a) Example of an intra-group matrix and the resulting simulated formation in [Qiu and Hu, 2010]. (b) Example scenario of groups crossing a street in [Ren et al., 2017]. Groups can split and merge back following link connection between subgroups. (c) In [Kremyzas et al., 2016], groups separate into subgroups, in this case a subleader is defined to wait for other sub-group(s).

Other works focus on automatic group formation. For example, Lermercier and Auberlet [2015] and He et al. [2016] gave higher-level reasoning to agents and let them adapt their local interaction (group formation, collision avoidance...) to the situation. Some other works use data-driven methods to model group behavior. Lee et al. [2007a] used video data to feed a learned model, which is used to simulate group behavior. Casadiego and Pelechano [2015] showed preliminary results of a Reinforcement Learning (RL) method for agent navigation. RL systems aim at achieving an objective via trial and error. The two most important components are the state description, which encodes the current situation of an agent, and a reward function, which rewards or punishes specific actions (leading to a change of state). Their state description encodes the relationships between an agent, its goal, and its neighbors in a discretized way. Their reward function rewards approaching the target and penalizes being too close to obstacles. They used the same model to create group formations by rewarding agents when they are close to group members and penalizing them if they are not. The results are promising, but the authors highlight the difficulty of finding the “best” problem design. Other authors worked on role attribution inside groups. For example, some papers identified the role of leader and follower. While the signification of a leader is not always the same in the literature, it is usually a member that has a unique role in the group by either yielding more information [Musse and Thalmann, 2001] or guiding the entire group, which is the most common case.

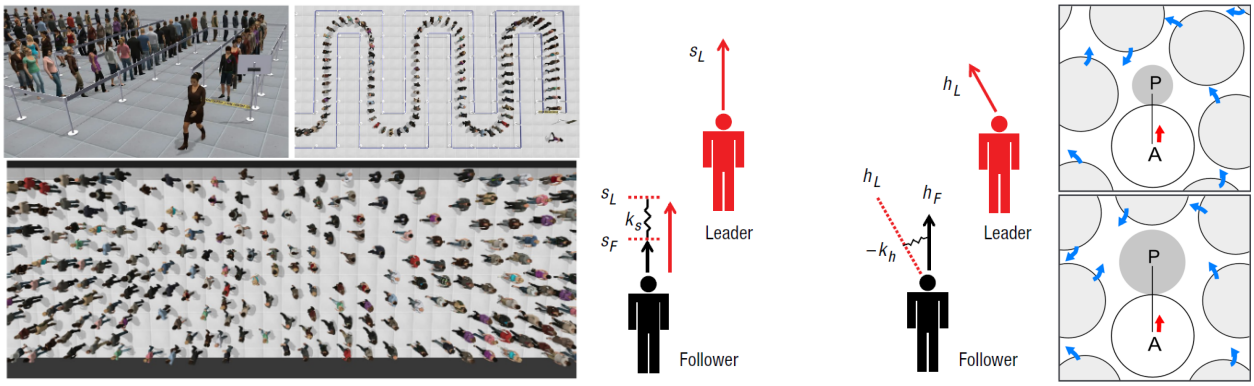
The leader is however not necessarily fixed, as in the work of Kremyzas et al. [2016], where the leader role always switches to the agent whi is the further ahead in the group. A group can be coherent if all agents can see the leader and go in the same direction. If cohesion is broken and subgroups emerge, the leader may wait for followers to plan a global path to the leader, thus determining the global path of agents. In this case, a group formation affects multiple levels of crowd navigation, as it serves to

determine the global path of followers and their local interactions. This is not the only work that switches between navigation levels, as following a leader can generally be described as setting the global path with the leader's position as the destination. This becomes a following scenario, which brings us to our next section.

1.3 Following

Following happens when one's motion is constrained by another pedestrian moving in front of one, without the possibility of overtaking. In this case, followers adjust their motion to move behind the obstacle without causing collisions. This interaction is often studied in crowd simulation literature, e.g., how people follow each other in a queue or a corridor (see Figure 2.5(a)). In those cases, the control of the agent's speed is specific because it requires the constant adjustment to the motion of the agent in front. The acceleration and deceleration of the agent need to be triggered on and off , which propagates through the crowd at a higher level. This speed adaptation depends on various parameters and is not easily implemented. Lemercier et al. [2012] used real-life data of circular queuing to define a rule stipulating that a person's acceleration depends on the crowd density, the difference in speed with the person in front, and a delay time. After adding this rule to the RVO model [van den Berg et al., 2008], they showed that the resulting simulation was closer to real-world measurement. Lemercier and Auberlet [2015] simulated group behavior, similarly to Kremyzas et al. [2016], by coupling grouping with following local interaction. Each agent analyzes its environment to check which conditions are satisfied to activate a behavior (group avoidance or following). In the latter case, they compute a tangential acceleration which can be considered an anticipation in speed to match the one of the leader.

Rio et al.'s model [2014] is also centered on the leader, that can move around freely while the followers have maintain personal distance. They could test their model in VR where a real person was the follower, and showed that their results match the results for one line following. The previously presented retina work [Dutra et al., 2017; López et al., 2019] can also model following behaviors by using scenario-specific cost functions. Instead of focusing on matching the speed of the follower and the leader as in previous work [Lemercier et al., 2012; Rio et al., 2014], Bruneau et al. [2014] explored the distance at which followers try to match the leaders'speed. They defined a dynamic ideal following distance that the follower should reach according to the leader's predicted position. The model is evaluated with real data and showed that it can generate realistic trajectories and reproduce patterns observed in real life. Finally, Warren [2018] presented a model in which the follower linearly accelerates to match the leader's speed and angularly accelerates to match the leader's heading direction. He was also able to model the global and local behavior of an agent following a group. In this case, the agent's motion is affected by the movement of all agents in the group within a certain area ahead. This model could be tested and compared with VR during several scenarios, demonstrating the interest of the method. This method was applied to crowds where a conglomerate pattern emerged. Each neighbor follows the ones in front of it, and as the pattern spreads through the crowd, each agent becomes both a follower and a leader.



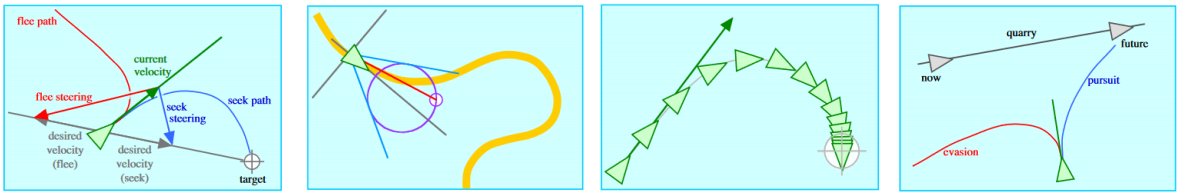
(a) Example of queuing and following **(b) Follower model** **(c) Proxy example**

Figure 2.5 – (a) Example of queuing along a winding path (top) and corridor traffic that combines avoidance and following behaviors (bottom) from Lemercier et al. [2012] (b) In Warren's work [2018], a follower is attracted to either a leader's speed by a spring with stiffness k_s (left) or a leader's heading direction by spring with stiffness $-k_h$ (right). (c) Example of an aggression proxy from [Yeh et al., 2008], as A's urgency increases, its aggression proxy, P, grows and the other agents move to avoid it, leaving a space for A to move into.

1.4 Other Interactions

While collision avoidance, grouping and following are the three local interactions that are the most frequently studied in crowd simulation literature, it is however important to highlight that many other behaviors are displayed in real life crowds. The question of variety in local interactions therefore appears to be a key point to model to increase realism. To add a new type of behavior to a crowd simulation, it is common to add a new algorithm, force, or cost function for that specific purpose. Each new behavior therefore requires programming effort, knowledge of simulation details, and parameter tuning. In his work, Reynolds [1999] brings a variety of local behaviors by adding for example wandering (Figure 2.6(b)), seeking and fleeing (Figure 2.6(a)), arrival (Figure 2.6(c)) or pursuit and evasion (Figure 2.6(d)). To model those behaviors, Reynolds had to define new specific velocity rules for each precise behavior. This design process is difficult, as specific rules can be hard to combine and complex to conceive. Recently, Saeed et al. [2022] re-implemented three of Reynolds' flocking behavior rules to simulate group steering in case of cohesion, separation and alignment.

Several other methods also try to enrich local interactions by increasing the range of possibilities by stretching the concept of an agent. For example, Yeh et al. [2008] modeled special interactions by adding invisible "proxy agents" to the simulation. This can model, for example, an agent that makes more or less room in a crowd depending on its walking speed, or an aggression proxy that appears when an agent is in an emergency situation (Figure 2.5(c)). Kapadia et al. [2009] define the concept of affordances to address space-time planning and account for complex interactions of agents with their immediate surroundings (e.g. agents, obstacles). Each agent perceives the environment through a set of vector and scalar fields that



(a) Seek and flee **(b) Wander** **(c) Arrival** **(d) Pursuit and evasion**

Figure 2.6 – Reynolds [1999] models example of local interactions

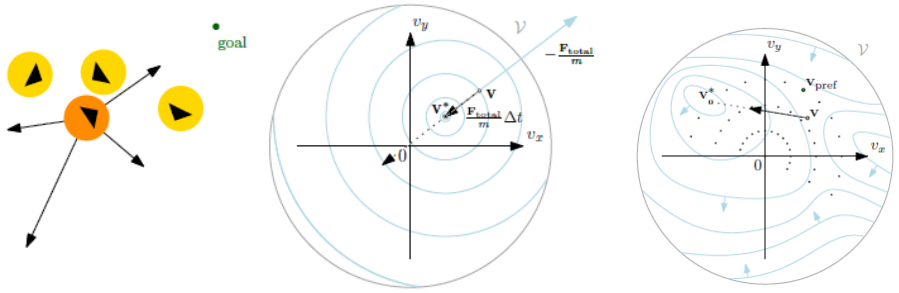
are represented in the agent's local space. Affordance fields quantify a certain measure called "fitness" for each possible action. The actions having the optimal fitness are selected as output to control the velocity of the agents. Comparably, the "situation agents" by Schuerman et al. [2010] are abstract agent-like entities designed to solve specific problems, such as deadlocks at narrow passages. The orientation of an agent is usually not explicitly controlled, although some exceptions to this rule exist [Hughes et al., 2015] where orientation is controlled to model holonomic behaviors like sidestepping, walking backwards. Although such techniques can indeed model additional behaviors, designing a new type of behavior still requires substantial effort and expert knowledge.

1.5 Cost function

This section highlighted three main local interactions that can be modeled by defining how agents move according to a certain principle, such as forces [Helbing and Molnár, 1995; Karamouzas et al., 2009; 2014; Zanlungo et al., 2011], velocity cost functions [Guy et al., 2010; Karamouzas and Overmars, 2010; Moussaïd et al., 2011; Paris et al., 2007; van den Berg et al., 2008; van den Berg et al., 2011], or vision [Dutra et al., 2017; López et al., 2019; Ondřej et al., 2010].

It is interesting to point out that those modeled can actually all be translated into cost function. In their paper, van Toll et al. [2020] demonstrated that all those principle can be translated as method that optimize a cost function in a velocity space. The challenges becomes then to define the right cost function for each algorithm and the optimization method to apply to this function to compute an acceleration or velocity vector for the agent.

In this end, van Toll et al. defined for each agent, a preferred velocity $\mathbf{v}_{pref}$ that is described as the optimal velocity an agent A could take to reach its goal. $\mathbf{v}_{pref}$ could be for example the velocity propelling towards the goal position of A. The goal is then to select from a velocity space (Figure 2.7(b)), the function with the lower cost to get the closer possible velocity to $\mathbf{v}_{pref}$ (Figure 2.7). van Toll et al. defined different cost functions, according to the model of collision avoidance of the previously cited crowd algorithm. The cost of a velocity for an agent can be based on various kind of information (position, goal, velocity...) that induces several metrics used to define cost functions. Those metrics are the same than the one highlighted in the literature and can be, according to the authors, any mix of the following: the distance between two agents, time to collision between two agents, distance to the collision, the distance



(a) Force Model **(b) Velocity Space**

Figure 2.7 – Figure courtesy of van Toll et al. [2020]. (a) Translating a force-based navigation method to the domain. An agent experiences forces from other agents and from the goal (left). The cost $C(\mathbf{v}')$ depends on the distance between $\mathbf{v}'$ and the velocity $\mathbf{v}^*$ suggested by the forces (right). (b) Translating a typical sampling-based navigation method to the domain. Values and gradient of the cost function are visualized in light blue.

and the time to closest approach (Figure 2.2(b)). For each algorithm, they provided fitting equation for those metrics defining the final cost function. Then, using the cost function, there are several ways to choose the agent's next action through an optimization method:

— Gradient step: giving the current velocity of an agent, move it to the opposite direction of the gradient of the cost function. The result is an acceleration vector. This makes this method particularly fitting for force-based algorithms, translating this acceleration to a force (see Figure 2.7(a)).

— Global optimization: find a velocity with minimal cost and apply it to the agent or convert the result to an acceleration as well. This method does not always have an analytical solution, some implementations therefore approximate the optimal velocity by sampling multiple candidate velocities and choosing the one with the lowest cost (see Figure 2.7(b)).

Overall this functioning principle is a powerful tool to control agents motion and design local interactions. However we can still notice that the number of local interactions defined by cost function is still limited.

Each category of this state of the art highlights one of the three common local interactions studied in crowd simulation. Apart from the few approaches cited above, models for other types of local interactions are very rare. Nonetheless, collision avoidance, group formation, and following are just examples of local interactions. On a street, children may play tag or other games. When two people walk hand in hand, they maintain a close distance and always stay side by side. If someone is giving away flyers, they would try to block the path of a potential customer or follow them to a certain distance. A child might push or pull another person to make it go faster. Some people might stop or watch a street show. And there are many other examples that could be found in other more mundane settings (a playground, a museum, a football game...). While this variety should be considered, bringing more variety in crowd simulation is

challenging as there are difficulties in the conception of new behaviors for the sake of a specific scenario.

When talking about cost functions, it is actually a hard process to find the adequate cost function and optimization for the desired local interaction. The definition of those behaviors are not accessible to novices because it requires mathematical and computational skills. It is important to point out that the previously cited works focus on very precise local interactions that are not intuitive to design and that, up of today, there is no solution to diversify and extend these models. One way to enrich these behaviors is to allow the user to customize them depending on the specifics of the scenario. Since an important goal of this work is to enable the easy design and editing of local interactions, the next section of this state of the art presents the existing techniques of interest, describing in more details how a crowd simulation could be authored.

2 Crowd Authoring

Authoring crowd is an interesting process as it aims to enable naive users to design a specific scenario according to their need. In their review, Lemonari et al. [2022] describe different levels at which a user can author a crowd (Figure 2.8). At a high level, one can author a scenario before the simulation is generated (defining goal, path planning...) and interactively change the environment at run-time, or at a low level, one can polish the animation of the characters (appearance, character animation...). Providing tools that enable interactive modification at all stages of the crowd simulation process is a prerequisite for users to achieve their desire outcome scenarios. The following subsections will review various approaches to crowd authoring proposed in the literature.

2.1 Parameter Tuning

First, most of the collision avoidance models described in Section 1 can be tuned by parameter modifications. Karamouzas and Overmars [2010] cost function has three main factors (time to collision, desired velocity and difference between the desired and the current velocity) that can be tweaked to produce a variety of avoidance behaviors. ORCA [Guy et al., 2010; van den Berg et al., 2011] also provides low-level control by scripting. van Toll and Pettré [2019] used a topology driven method that detects possible collisions and triggers the re-planning of the global path. By adjusting the re-planning time parameter, authors control the avoidance strategy. In vision-based approaches [Dutra et al., 2017; Ondřej et al., 2010]), users can control certain properties such as anticipation time and security distance from obstacles. López et al. [2019] also let users tweak parameters to vary adaptation time and freedom as to deviate trajectories. Data driven methods allow for some level of authoring in addition to changing the input data. For instance, in Charalambous et al.'s work [2014], the neighbors taken into account and the temporal representation can be modified. More recently van Toll et al. [2020] redefined classical crowd algorithms as cost functions with specific parameters, for each function, that can be tweaked by scripting.

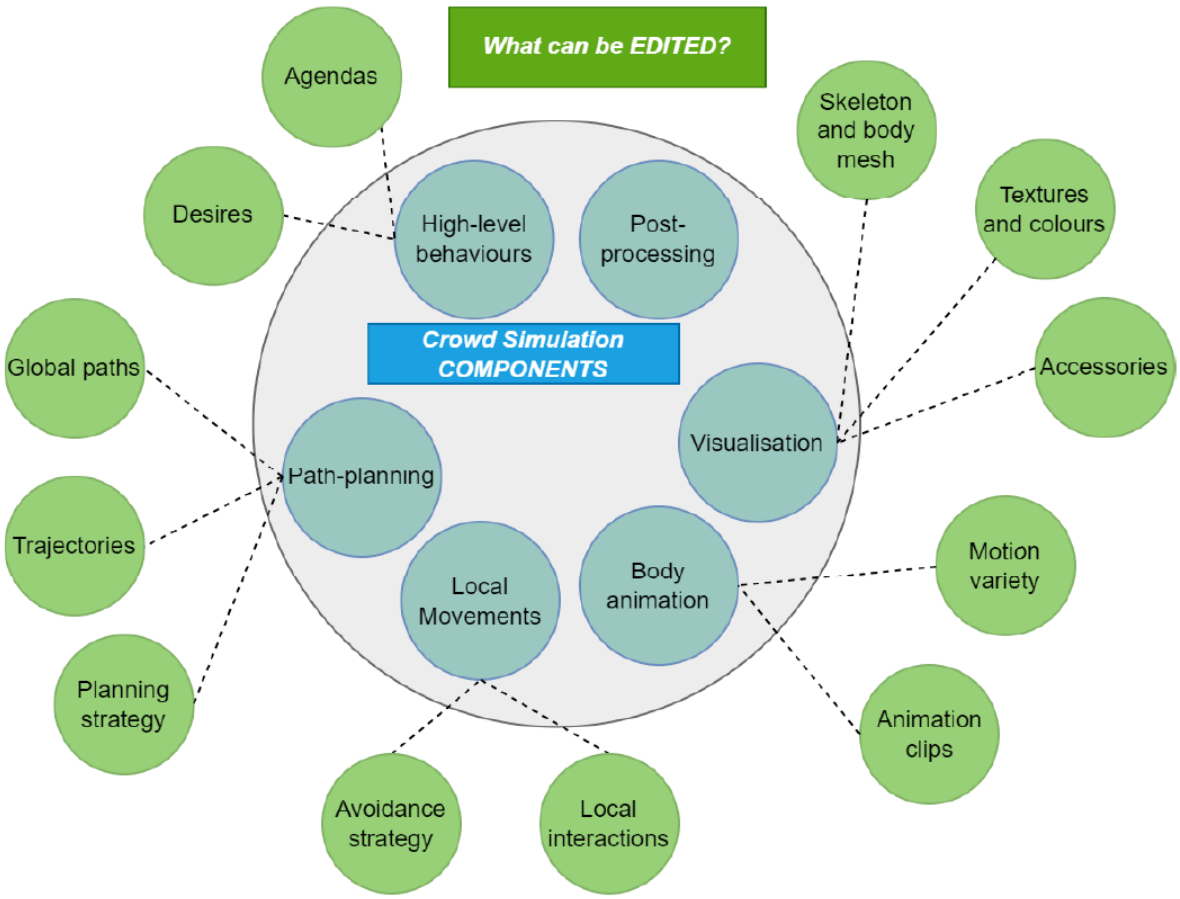


Figure 2.8 – Overview of crowd simulation components and each component's authorable aspects. Image courtesy of Lemornari et al. [2022].

Parameter customization of those methods solely allows to modify collision avoidance behaviors by adjusting the local interaction, such as the distance of avoidance. Other techniques enable user scripting input to tweak group formation, such as editing intra and inter-group matrices [Qiu and Hu, 2010] to decide the influence between agents and groups. Ren et al.'s method [2017] also enables the velocity connection set to be authored via relation matrices, that can be edited dynamically at run time. This refinement of parameters can, for example, force groups to keep cohesion and not break, while other methods enable to change group properties such as size or the shape [Krontiris et al., 2016], sometimes dynamically like [He et al., 2016]. The latter approach enables for environment semantic and crowd demographics manipulation, through the definition of environmental attractors (that can be agents too), ultimately creating influence maps that propel agents. When it comes to following interactions, user intervention is also limited to parameter manipulations, such as agents' reaction times, density and acceleration amplitude [Lemercier et al., 2012].

To resume, those methods enable low level authoring control that is only available through scripting,

using very abstract parameters which makes those method unattainable to the many. Scripting and abstract parameter tweaking is difficult to tackle for novices because it require either an expert understanding of the algorithm or an adaptation process with various trials and errors to understand the parameters control. We can point out that interactive techniques, allowing dynamic change of parameters at run-time have a non-negligible advantage, reducing the trial and error process time drastically as users can directly derive the results. However, these parameters would always be difficult to understand and their effect is limited.

Studies have hence been made to improve the accessibility of such authoring tools, either by improving the interface to get rid off the scripting or by mapping parameters to less abstract concepts. These latter techniques include narrating the scenario through text or scripting when the users can basically describe the scene, most of the time including temporal control of the scenario as well.

Narrative

Numerous studies focus on making crowd authoring more operational, either by simplifying the setting of parameters through graphical user interfaces (GUI) or by mapping the abstract parameters to more representative information such as personality traits or desires. The first option is very simple and easier to implement, but it is not always sufficient to use sliders to tweak around parameters. If the parameters are too numerous or too abstract to be clearly understood, the outcome of the simulation is not easily predictable and it is still difficult to create an accurate scenario. Normoyle et al.'s approach [2014] computes randomization parameters for crowd simulation which supports direct control of the crowd (activities, duration of the activities, groups) using a GUI with sliders. The method of Allain et al. [2014] suggests optimal parameters to the user. Depending on the environment, the initial state, and the constraints given by the users, these optimal parameters should describe a scene that matches the users' expectations.

Most often, authors focus on other types of interfaces or on a more descriptive method. For example, they can use natural language with a framework that takes English sentences describing a scenario as input [Badler et al., 1998; Chen et al., 2020; Liu et al., 2020]. These methods provide control over a limited number of attributes: time scheduling, grouping, start and end position, animation, triggering events, where the goal is to build a narrative. Other approaches work on the narrative by creating schedules, such as the CAROSA system [Allbeck, 2010], where users write and assign responsibilities and attributes to agents via a simple Microsoft Office tool. Roggla et al. [2021] integrate procedural crowd generation and enable specification of agendas using a rule-based grammar. The environment can also be manipulated by copying and pasting objects in different locations or by adding landmarks. Lee et al. [2007b] used video data to learn behavioral models that enable the authors to assign a behavior to each agent, e.g., chatting. Similarly, Li and Allbeck [2011] assigned roles to characters that define their activities: shopper, spouse, mailman... The method then relies on a role switching that can be triggered by events (meeting between two friends), schedule, or location (workplace).

Mapping to human traits

Complementary to providing ways of creating crowd scenarios, some authors have attempted to model a more realistic approach by adding real human characteristics to the framework of the simulated agents so that users could select not only behaviors but also psychological human traits. The challenge is then to map these traits to human trajectories and motion. This brings us closer to the psychological domain where perceptual studies may be necessary to validate the results.

Funge et al. [1999] gave agents the concept of knowledge and learning that enables them to autonomously select a sequence of actions that satisfy the specification. Shao and Terzopoulos [2007] incorporate a memory component into their system, representing the mental state of agents with individual parameters such as fatigue and curiosity, as well as other parameters such as courage [Yu and Terzopoulos, 2007]. The cost function described in van den Berg et al. work [2008] uses a parameter that determines the agents' reflected aggressiveness or inertia. The HiDAC [Pelechano et al., 2007] framework focuses on local interactions during evacuation processes by modeling panicked agents that pass on their anxiousness. Guy et al. [2011] mapped collision avoidance parameters as in Table 2.1 to Eysenck's 3-factor personality traits model [Eysenck, 1985]. Using video recordings, they mapped various crowd simulator traits to the three PEN personality traits: Psychoticism, Extraversion, and Neuroticism. The resulting trajectories were validated by a perception study. Sinclair et al. [2015] also attempt to integrate emotion and personality into their framework, which affects both local movement and pathfinding. Table 2.1 shows the assignment of agent parameters for each personality used in their implementation. Based on perceived personality data, collected from a user study with users watching videos of different crowd simulation scenarios, different personality traits are mapped to different parameters values. Those values are ORCA [van den Berg et al., 2011] default agent parameters such as neighbor distance, maximum number of neighbors, planning horizon, radius and preferred speed and are then directly set in an ORCA library simulation. They actually adapt the parameters of Guy et al. [2011] to larger crowds by reducing the radius trait by half, using the same parameters. Durupinar et al. [2011] extend the HiDAC model by integrating the mapping of low-level parameters into the OCEAN (Openness, Conscientiousness, Extroversion, Agreeableness, and Neuroticism) personality traits of the five-factor model [Wiggins, 1996]. Untrained authors can then more naturally select the behaviors they want an agent to exhibit, which also creates motion variety.

Traits	Max. Neighbors Distance	Max. Number of Neighbors	Planning Horizon	Obstacle Planning Horizon	Agent Radius	Max. Speed
Aggressive	15	20	31	31	0.6	1.55
Implulsive	30	2	90	90	0.4	1.55
Shy	15	7	30	30	1.1	1.25
No Personality	15	10	10	10	1.0	2.0

Table 2.1 – Personality trait mapping to low level crowd simulation parameters [2015].

Durupinar et al. [2015] further worked with the OCEAN model to include crowds with different personalities and emotions. In this later work, they give the possibility of creating scenarios with different emotion levels, for example, representing expressive and acquisitive crowds at a protest or in a sale. Finally, group representation could be modeled using personality, as in Villamil et al. [2003], where agents are represented using social traits: Sociability, Communication, Comfort, Perception, and Memory. The resulting group is characterized by the Cohesion parameter, which symbolizes the homogeneity of the group members' mindsets.

The behavior produced by a local algorithm usually depends on a number of parameters. For any system that operates solely on simulation parameters, the results are naturally limited to what parameter variations can produce. By tuning these parameters, designers have some control over agents behavior, but this tuning is not always intuitive and requires expert knowledge of the simulation model. Furthermore, parameters alone do not allow designers to develop entirely new behaviors. In this section, we have described the different ways in which local interaction models can be tuned, and we note that the main focus has been on making different types of limited local interactions (collision avoidance and grouping) easier to describe by, for example, changing the inter-personal distance between agents. However, collision avoidance, grouping and following are only three examples of possible local interactions, and a user might rely on other types of local behaviors. Therefore, it remains difficult to establish the relationships between a designer's desired crowd animation and simulation models and their many parameters. This depends on the experience of the animators, who must make all the successive adjustments to the simulation settings in order to successfully manage the crowd simulation process to produce the expected animation.

However, if we take a look at other levels of crowd authoring (Figure 2.8) and related literature, we find that authors have tried to facilitate the crowd creation and editing process for global planning, for example. We see much more of inventive and user-friendly authoring processes there, and sense interest in such techniques that could be applied to local interactions. We believe that some techniques used for path planning authoring could be an inspiration to design local interactions as well. Exploring the different types of authoring path planning is not totally out of the scope since it can sometimes really well include the local interactions between agents. There are many ways of authoring path planning: two of the most popular are the use of patches, and the second the use of flows to describe the trajectories of the agents. The first methods revolve around the pre-computation of a set of behaviors, often called patches, or trajectories from which the user can choose to build a scenario. The next section looks at these various behavior templates. Those template can be piece pre-computed crowd simulation (patches) or simply local interaction modeled by fixed parameters.

2.2 Discrete Set of Templates

Letting users choose from a set of behaviors to apply onto the scenario enables to keep a control on the realism of the results while letting users make the final decisions. Those templates directly give a

structure of what is possible while assuming that templates can be combined or edited, providing great variability of scenarios. Since the templates are often pre-computed, it gives the possibility to preview the template which enables, compared to the last section, users to know what to expect from the final simulation. Special care must be given to the templates selection or editor interface to ensure that the preview of the templates and their application will be understood easily by the non-trained users. For example, Kraayenbrink et al. [2014] specified re-usable crowd templates to author scenes based on agent desires and ambitions selected via an interactive editor. Peters et al. [2009] defined subgroups using cohesion matrices that specify the space between sub-group members and enable users to select from group formation templates, such as in Figure 2.3, to apply to the groups. Groups can still break into smaller subgroups to avoid objects or to transition from one formation to another.

Motion patches

Another set of solutions is more oriented towards editing existing motion clips to adapt to new situations.

Those approaches are data-driven and grant authoring control to manipulate the trajectories and even local interactions over time and space. Several patches can be predefined by authoring, and the challenge of those methods is then to be able to edit patches and most importantly to stitch them together into a larger scale environment. The first work using this technique is Motion Patches [Lee et al., 2006] which builds blocks of animated motion from registered data, thus ensuring realistic pieces of local behaviors. Even though users can tweak some parameters affecting data-clustering, the main authoring interest here is the possibility to interactively design the animated trajectories of agents by stitching the motion patches. Shum et al. [2007; 2008] worked on applying those patches to dense crowds, incorporating local interactions of agents inside patches. Close interactions between characters are precomputed offline by expanding a game tree, and these are stored as data structures called interaction patches to be called at runtime. The user can select, from a set of high level commands, an action for a character. From this command, an interaction patch will be selected based on the requirements. In addition, Kim et al. [2009; 2012] worked on the free editing of motion data of synchronized animated characters. They originally designed a free-hand authoring tool that enables direct motion manipulation through handles that specify a spatial location, direction, temporal location, and timing of interaction [2009]. They later on extended their work [2012] by improving the tiling of motion patches, achieving refined local interactions between autonomous agents that can now meet inside patches and interact through captured animation of contacts (such as hand shaking, hugging, and carrying a heavy object collaboratively), illustrated in Figure 2.9(a).

Deformable meshes

There is also a line of work that models a crowd as a deformable mesh, to steer it efficiently along (user-specified) paths in the presence of obstacles [Henry et al., 2012; 2014; Kwon et al., 2008; Zhang et al., 2020]. Kwon et al. [2008] let users edit group motion by manipulating the trajectories represented

(a) Motion patches deformation **(b) Crowd patches stitching** **(c) Sculpting crowd made of crowd patches**

Figure 2.9 – Example of patches and their combination respectively. (a) [Kim et al., 2012], (b) [Yersin et al., 2009] and (c) [Jordao et al., 2015].

as graphs. Those graphs constructed from motion clips, can be stretched, cut and deformed through Laplacian deformation. This inspired the method introduced in Kim et al. [2014] that grants straightforwards interactive crowd control, still allowing time and space path manipulation of larger crowds. To obtain this, they modeled blocks of trajectory lines that are grouped under a 2D cage-based geometry. Those cages can then be interactively deformed through the usual handles. This work opens the door to the manipulation of crowd path through enclosed meshes. Henry et al. [2012] represented crowds with a mesh that can self deform to react to the environment but does not allow user's control on this mesh. Recently, Zhang and al. [2020] took inspiration from Kim et al. [2014] and similarly enclosed animated characters into a mesh that can be bent by the user, with the difference that the vertices of the mesh are actually defined by the positions of the agents. The mesh is able to deform automatically, to pass through a narrow entry for example, while retaining the formation of the crowd and the relative positions between characters.

Importantly, Kim et al. [2014] and Zhang et al. [2020] showed very intuitive and user-friendly techniques to manipulate rigidly and robustly dense crowds. The mesh representation transforms the crowd into an elastic object, making it easy to bend the global path by vertex manipulation while keeping local interactions constraints. However as it can only be used for modeling lines of walking forwards characters, other approaches are required for more complex situations.

Crowd patches

The most used patch-based technique for crowd simulation authoring is Crowd Patches by Yersin et al. [2009]. It basically stores a data set of pieces of local crowd simulations, defining diverse patch of trajectories that can be extended or cut. Those pieces of crowd can also be stitched and controlled by fitting trajectories entry and exit (see Figure 2.9(b)) over time and space to populate realistically a large environment. Following up on the concept of 'crowd patches' that can be stitched together [2009], several methods let users intuitively deform crowd motion clips to create crowds of different shapes [Jordao et al., 2014; Kim and Lee, 2016; Kim et al., 2014] or densities [Jordao et al., 2015]. This concept has been utilised in further research by Jordao et al. [2014] who introduce a powerful authoring tool that integrates

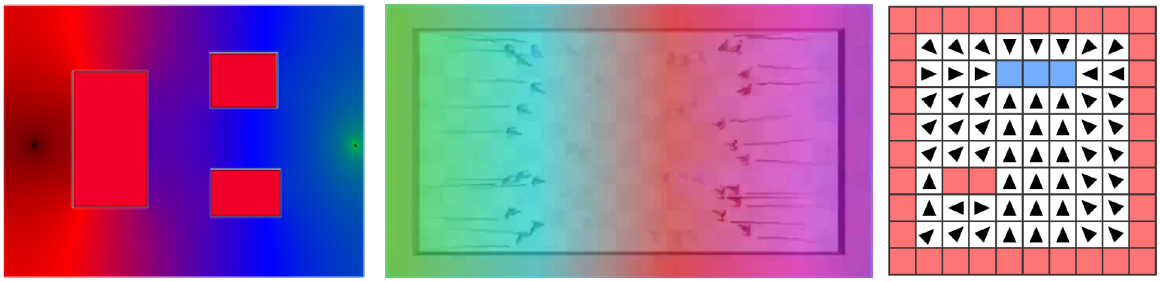
space-time alterable and stretchable crowds by automatically inserting or removing unit crowd patches as can be seen in Figure 2.9(c). Those pieces can be intuitively bent, stretched, cut and blended together at wish at run-time, always ensuring continuity of the characters' animation and of the population size. The authoring also enables the selection of crowd patch schedules to create large, endless, moving crowds. In Jordao et al.'s work [2015], crowd patches are used to guide crowd flows through the environment via a painting interface, enabling local control on both crowd density and motion flows, achieving artistic results. Crowd Patches are therefore a good solution to design quickly and realistically large scale crowd environment, as in Jordao et al. [2014] work, but it does not enable to refine precisely the interactions between agents. The more often, collective behaviors are restricted to co-navigation, i.e. managing a large number of agents' customizable locomotion in a scalable environment, but do not consider other local interactions than collision avoidance. As in any template oriented techniques, we are limited by the given discrete set of data, here the animated motion of the unit patches, which therefore prevent a user to easily create new behaviors.

2.3 Field Propelling

As mentioned before, vector fields are also often used to propel agent. Fields give clear and straighforward visual information about the the trajectories of the agents. They can be easily defined and sketched which gives much more flexibility to the authoring. This topic will be described in this section.

Crowd flow

Research has been made over the years to provide crowd flow specification. Flow control relies on the possibility to manipulate the fields propelling the agents. The first that took an interest in flows is Chenney [2004], that described flow tiles. Similarly to crowd patches [Yersin et al., 2009], flow tiles are small fields that can be combined to produce large flows, with border adaption to ensure continuity. An interface to specify constraints and velocity across tiles is provided and the tiles are divergence-free, which ensures that the crowd is guided safely. Barnett et al. [2014], used an harmonic field as a representation of the space to extract the topology of the scene (Figure 2.10(a)). The max-flow is then calculated across the graph to reflect the global path of the agents. A number of guidelines are also calculated by following the gradient of this field, which are followed by flocks of agents at runtime. Treuille et al. [2006] exploited the advantages of potential fields to guide a crowd as a whole. Several grids are first defined according to user inputs (positions, entrance, exit, walls...) and then combined into a dynamic potential field deciding global navigation as illustrated in Figure 2.10(b). This method simulates the motion of large crowds without explicitly defining collision avoidance but offers poor authoring possibilities. Park [2010] also made use of potential fields, determining such fields from the trajectories of the control particles, while reinventing their role for guiding crowd flow in the process. The trajectories of the control points are defined through a dedicated interface where agents can define their trajectory of control particles via key-framing their positions.



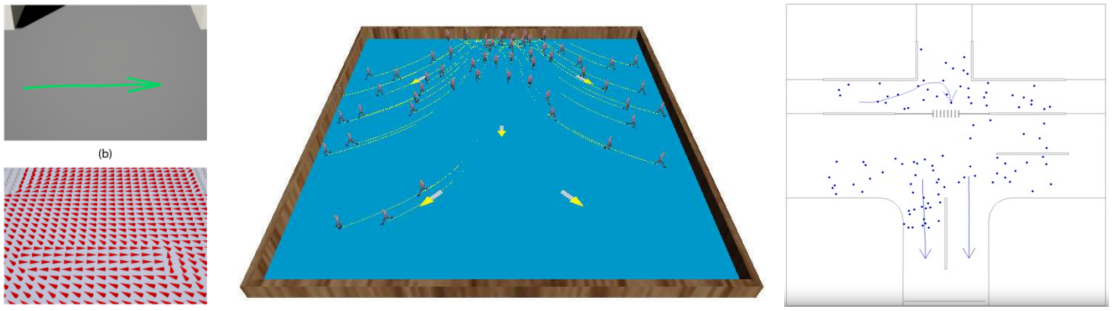
(a) Harmonic Flow (b) Potential Field (c) Force Vector Field FVF

Figure 2.10 – Example of flow fields. (a) [Barnett, 2014], (b) [Treuille et al., 2006] and (c) [McIlveen et al., 2016].

Data driven methods such as Ju et al. [2010] can extract information from data and learn both trajectories and formation models, the interaction between the two enabling to synthesize crowds of any size and length. Those various crowd styles can be blended by interpolation. Bulbul and Dahyot [2017] also used a data-driven approach to produce flows of pedestrians, using real life city data. Paravisi et al. [2008] extracted trajectories from video footage from which they generate velocity fields. They also used example trajectories given by users to estimate comfort maps and define areas where humans will tend to go. By exploiting geo-location through social media, they can estimate the spatial and temporal distribution of people. Karmakharm et al. [2010] focused on large crowds and, as McIlveen et al. [2016], make use of force vector fields (FVF) combination to describe navigation. For example, a Collision FVF encodes repulsion forces whereas the specification of entrance and exit define Navigation FVFs. The PED system [2016] extends this work by enabling more authoring through a complete user interface. Users can first define the environment through an editor and then attribute layers of behaviors to objects. Designers can sketch FVF through the interface that are then combined and converted to obtain a final FVF that guides pedestrians to the exit (see Figure 2.10(c)). Similarly, Gonzalez and Maddock [2017] exploit a sketch-based interface to define navigation maps (Figure 2.11(a)). By sketching arrows in the environment users can define and change the trajectories of agent.

Sketching field

Sketching fields through guiding lines is actually a pretty popular authoring technique when it comes to using fields. For example, Jin et al. [2008] enabled users to specify vector fields by combining RBFs (Radial Basic Function) as in Figure 2.11(b). Users can sketch arrows to control the crowd motion by assigning velocities to anchor points. The resulting velocity field is an interpolation of the effects of the anchor points. The method provides immediate feedback through interactive manipulation of these points and proves useful for dealing with large crowds at interactive rates. Kim and Lee [2016; 2014] generated flow fields through a painting interface. Users can paint strokes which radius indicate the impact and the direction of the line. Finally, with navigation fields, Patil et al. [2011] enabled authoring in two ways:



(a) Sketch Navigation maps **(b) Sketch by RBFs** **(c) Sketch guidance field**

Figure 2.11 – Example of sketched fields. (a) [Gonzalez and Maddock, 2017], (b) [Jin et al., 2008] and (c) [Patil et al., 2011].

through a sketch-based interface and by inputting flow fields corresponding to real-life crowds motion. Users can sketch lines, containing stroke width and decay ratio information, to design "guidance fields". Those fields are attributed to subgroups to guide the motion of agents on top of their global path (see Figure 2.11(c)). The method blends the procedural and user input to ensure the agents to reach their goal while roughly being guided by the sketch. Compared to Jin et al. [2008], Patil et al. ensure that no singularities are present in the navigation vector field, by reprocessing their guidance field to remove local minima.

To summarize, some work explored ways to determine simulation parameters from more intuitive input, such as trajectories and flow fields [Allain et al., 2014; Kang, Kim, et al., 2014; Mathew et al., 2020] the crowd should follow. Such concepts were later adapted to deal with large-scale environments [Gonzalez and Maddock, 2017; Montana Gonzalez and Maddock, 2019].
Through our analyze of authoring methods, we highlighted approaches that rely on editing and manipulating existing data: models modification through parameter tuning or manipulation of behavior through selection of patches. Those techniques are implicitly limited to what the framework allows. It seems to us that local interactions cannot be included into a limited set, hence the interest to be able to create them from scratch.

As we saw in this last section, sketching is a viable way of authoring because it relies on very basic skills the majority of people are familiar with and gives a lot of flexibility. For instance, sketching field through the environment [Jin et al., 2008; Kim and Lee, 2016; Patil et al., 2011] lets user completely decide of the trajectory of the agents, giving the tool to build the wanted scenario more easily. We point out here that curves are sketched through the global environment, as one can see in Figure 2.11, to impact the global path of agents and not a lower the level of simulation. Still, it seems interesting to us to explore other sketching techniques in order to refine more precise behaviors. The next section will focus on sketching techniques and the necessary interface to exploit the freedom given to designers.

3 Sketch-Based Interface

Most related work on interactive crowd motion design is based on parameter editing or adaptation of existing crowd motions. To enable designers to create different behaviors at wish for the specifics of a scenario, a more powerful editing tool is necessary. From our point of view, sketching is a way to give complete freedom to the users in a intuitive way. The concept of traditional sketching, free-hand drawing on paper, is familiar to the many and does not require training. Sketching is very user-friendly and enables outcomes of types "what you see is what you get" which is why we want to explore into deeper details the potential paradigms of sketching in this section. Computer-aided sketching refers to drawing 2D free-form strokes on a surface using a mouse or any input device. These strokes must be converted to polylines before using the information. This process is called sampling. To keep this intuitive aspect, authors must be cautious of their interface so that novice users can still test their tool.

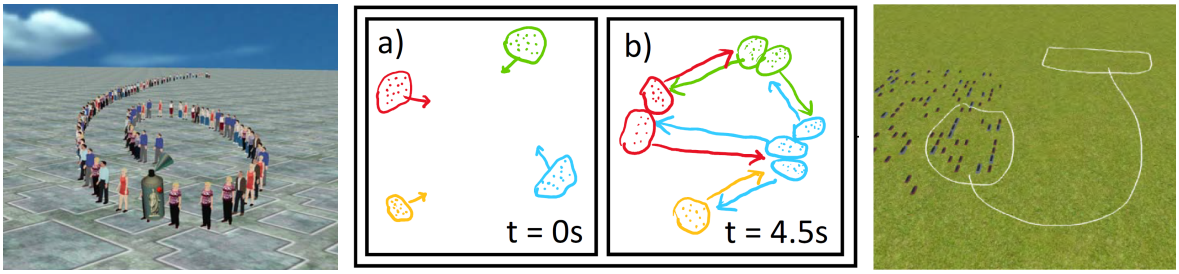
3.1 Crowd Sketching

Various methods exist that try to give users more intuitive control over a crowd via sketch-based techniques. Most of these provide control over *global paths*, by letting users draw curves for agents to follow [Bönsch et al., 2020; Gonzalez and Maddock, 2017; Kapadia et al., 2009; Oshita and Ogiwara, 2009a; Ulicny et al., 2004] or directional hints that are converted to a flow field (see Section 2) [Kang and Kim, 2014; Paravisi et al., 2008; Patil et al., 2011]. Some of these methods also propose sketch-based control of other simulation parameters, such as the walking speed and the smoothness of paths [Oshita and Ogiwara, 2009b; Ulicny et al., 2004]. Other work focus more on changing the geometry of the environment [McIlveen et al., 2016; Montana Gonzalez and Maddock, 2019; Savenije et al., 2020], on fitting a group into a formation [Gu and Deng, 2013; Henry et al., 2014], or on giving users high-level control over where agents go and which actions they perform [Kapadia et al., 2011; Krontiris et al., 2016; Mathew et al., 2020; Millán and Rudomin, 2005]. This section presents two main families of approaches for sketching crowds: one revolves around sketching the path of the agents and drawing directly the shape the group should align with.

Free-hand global path sketching

Most sketching techniques driving crowd simulation focus on directly drawing the global path of the agent or populating the environment. Various techniques like "CrowdBrush" [Ulicny et al., 2004] and [Oshita and Ogiwara, 2009b], propose a painting-based system to control simulation parameters. In "CrowdBrush", designers can use a virtual brush and spray to place the agents (see Figure 2.12(a)), to trigger different animations or actions or even to style the characters. They can also spray the trajectories virtual humans should take. Similarly, Oshita et al.'s work [2009a] let users sketch example path that will affect agents with the closest initial positions. Mathew et al. [2020] created spatio-temporal crowd motion models (CMMs) from real world trajectory or sketch input. They provided a sketching GUI

to easily set up virtual trajectories and sketch new CMMs. Designers can re-target those attributes by sketching through this interactive GUI. Those CMMs describe the initial and final position as well as the spatio-temporal clustering and motion of the agents (see Figure 2.12(b).) Gonzalez and Maddock [2019] make use of the Unity's NavMesh [Unity, 2021] (the NavMesh represents the area where the center of the agent can move) and provide a sketching tool to update a NavMesh in real time. With this tool, users can design a scenario: add agents' spawn and goal locations, sketching obstacles to alter the crowd movement, create flow lines to guide the motion of the agents, draw areas to create way-points, and define journeys via storyboards.



(a) Spray agents' positions **(b) Sketch CMMs** **(c) Transition formation**

Figure 2.12 – Sketch-based interface in [Ulicny et al., 2004] (a), that enables to free-handly define the positions of the agents as well as there global path. Interface in [Mathew et al., 2020] (b) and [Allen et al., 2015] (c) that enables to sketch groups motion and formation of agents.

Group formation sketching

Various work exploited sketching interfaces to design agents' group formation from sketch inputs. The goal is to intuitively and quickly describe desired formations, and to let the system ensure smooth transition between formation or path following. Early work from Takashi et al. [2009] let users define key frame formations and interpolate between them using spectral analysis. The key-frames are extracted to the desired shape from sketch inputs or video data. Later on, Gu and Deng [2011] presented an intuitive interface to sketch the formation boundaries. The method uses the formation coordinates to preserve the social distance between agents when they switch position and change formation. Gu and Deng [2013] then used the same principles using the brush metaphor. Groups can take any free-from shape and transition following a path while respecting local constraints (collision avoidance). Users can also decide of the crowd density inside the shapes by trying out different sampling rates. Finally, Allen et al. [2015] divided crowds in subgroups with specified paths via a sketch-based interface. The system achieves flow and formation control for heterogeneous crowds. A user selects subgroups by circling around the characters and then sketches a line for their trajectory before drawing the border of the next formation at the end of this line as seen Figure 2.12(c). Conversely Xu et al. [2014] do not use an intuitive interface but offer users control over several stages: target formation, overall direction and agent movement control.

Subgroups are formed to maintain the cohesion of the group. The movement of the agents is determined using the principle of least effort and an enhanced social forces model.

Some original work [Millán and Rudomin, 2005; Sung et al., 2004] does not only focus on sketching path but also on sketching more local behaviors (e.g., action, relationship). For instance, in Sung et al. work [2004], situations can be spatial, and attributed to a place through drawing directly on the environment, or non-spatial, and directly affect the crowd. By sampling a probability distribution, agents are more likely to realize an action according to the combined situations agents are under. Millán and Rudomin [2005] presented a 2D interface to paint image maps (every pixel holds information): geometric maps, isomorphic to the virtual environment, and agent-space maps linked to agent. Those maps modify the behavior as well as specify the attributes and state of these agents.

Although these latter works attempt to gain control over local behaviors, they do not enable users to easily define new behaviors. Overall, this section shows that sketching applications in crowd simulation improve global planning or group formation and give more power to the user. However, they still do not focus on the local level, which is the core of our work, as ways to design new local interactions are still lacking. Sketching is nevertheless a very interesting way to increase user control. In the next section, we will then extend the research area to get more insights in relation with sketching techniques in character animation.

3.2 Sketching Techniques for 3D Character Animation

A lot of research has been made to exploit sketches for 3D content creation, the main challenge being to translate the 2D input sketch onto 3D structure control. The survey of Bhattecharjee and Chaudhuri [2020] overviews the different sketch-based techniques for 3D modeling and animating, and describes a generic framework displayed in Figure 2.13. In this section we will only focus on the animation of characters through sketching, 3D object modeling being out of scope of this thesis.

Various methods focused on 2D animation [Gupta and Chaudhuri, 2018; Patel et al., 2016] where 2D human sketches are directly animated on the interface, however these approaches are not adapted to 3D characters simulation, as resulting 2D characters could not be included in a crowd simulation framework. Apart from that, a lot of work studied the creation of 3D animation from 2D sketches. Compared to 3D, 2D sketches are still easier made using a 2D interface like a computer screen and are more familiar to novice users. However, the difficulty resides then in translating a 2D sketch into a 3D model. Davis et al.'s method [2003] parses a set of 2D sketches to create articulated 3D animations, an example is shown in Figure 2.14(a). In Chaudhuri et al. [2004], they presented a view-dependent character animation to mirror the relationship between the character pose and the camera viewpoint as indicated by a sketch sequence. Jain et al. [2009] exploited the skills of trained 2D-sketch animators to create 3D animations of a human in motion. Professional animators sketch 2D animations that are generated in 3D using captured motions. Later on, Jain et al. [2012] added the possibility of including 3D proxy (e.g. clothes) to be influenced by the 2D animation as well. Gupta et al. [2018] proposed a way of animating input sketches but in 3D. The

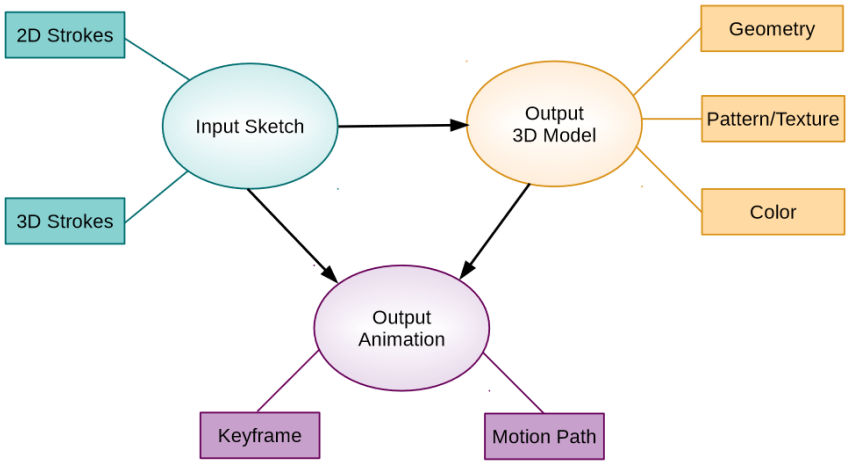


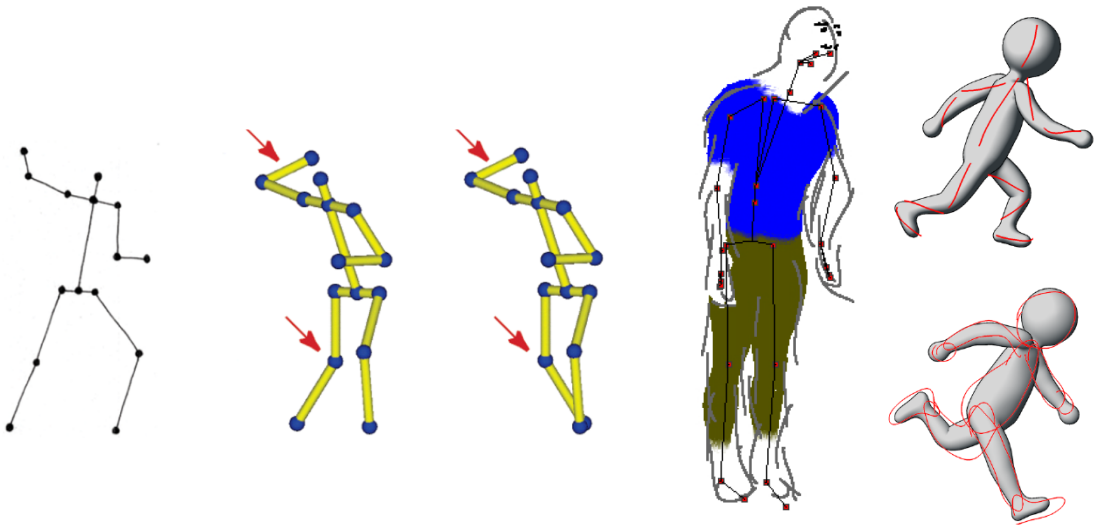
Figure 2.13 – Diagram of a sketch-based content creation framework. Image courtesy of Bhattecharjee and Chaudhuri [2020].

user provides two sketches: one from the front and the second from the side view of the character. The user needs to drag and drop a desired skeleton onto a deformed template mesh to fit the sketches (see Figure 2.14(b)). The pose of the model is then matched to the sketch by using a standard 2D-3D joint matching technique, where the ambiguity related to scale is resolved by using the two input sketches. The sketches are then animated and new sketches are generated by making the strokes match to silhouettes of the posed mesh, making possible to rotate the 2D animation in 3D. The applied animations come from a preregistered dataset and cannot be edited. Those previous cited works focus more on other model's parameters (e.g. style) and on 2D sketches to 3D model translation than body animation authoring, which is the principal focus of this section.

Animating a character relies on deciding on a motion path and key frames, i.e deformations model will under-go through time. In sketch-based animation, as suggested in Figure 2.13, one can either sketch the model pose for different discrete times (keyframes) or sketch strokes to apply to the model. The later can be used to modify different properties from the motion path to the shape deformation. In the context of sketch-based animation, there are two variants for deriving the path and deformation —deriving from multiple sketches of the model or deriving from one sketch stroke and applying it to the model.

Sketching keyframe

Techniques to support keyframe sketching are the ones that rassemble the most the traditional animation process. To animate 3D models, users can sketch the pose of the model at different keyframes, which are then animated by interpolation. The following techniques focus then on facilitating the process of 3D pose modeling using sketching. Bassett et al. [2013] proposed to let users sketch key-frames that are then integrated to the traditional animation pipeline. The rigged 3D model is posed from the key frames



(a) 3D reconstruction from 2D sketch **(b) 3D skeleton targeting on 2D sketch.** **(c) 3D pose with 2 different abstractions.**

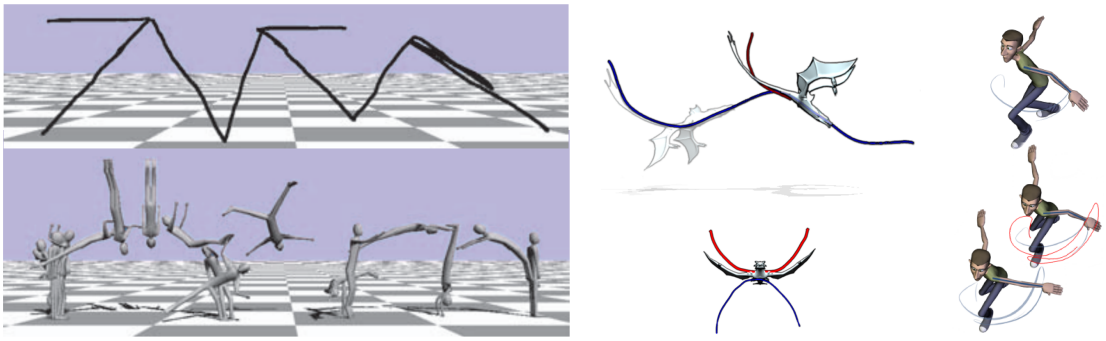
Figure 2.14 – Example of 3D pose reconstruction from 2D strokes. (a) [Davis et al., 2003], (b) [Gupta and Chaudhuri, 2018] and (c) [Hahn et al., 2015].

after sketch strokes processing (using convex combination of matrix to map the vertex transformation). Hahn et al. [2015] suggested a system to sketch the poses of 3D rigged characters. The sketch is then translated onto a sketch abstraction (two abstractions are visible in Figure 2.14(c)) composed of rigged curves forming a 2D representation of the model from particular viewpoints. The system minimizes a non linear interactive closest point energy to find the rigging parameters that match the best the sketch abstraction. Sketch abstraction can be drawn on the fly by projecting a drawn curve onto the character's mesh. They show application results where the user can sketch animations by drawing the different curves in a prescribed order.

Motion path sketching

Another way of editing animations through sketching is to use input strokes to define the motion of the model through time, i.e. the animation. The strokes help define the trajectory of the model's joints or the global (the trajectory of the overall body in the environment) and local (e.g. stretching, bending body parts) body motion of the model. Those techniques allow to sketch very rapidly and intuitively body animation. Thorne et al. [2004] enabled to animate models in 2D and 3D environment by supporting both 2D and 3D strokes inputs. The system "Motion doodles" enable users to define first a skeleton through particular types of strokes and then sketch curving strokes (doodle curves) to represent the motion path, using the vertical direction at corner points (Figure 2.15(a)). Eighteen types of doodle curves exist from

which they can decide the motion of the skeleton. The corresponding motion is applied to the 3D model while appropriately keyframing and interpolating the frames. Guay et al. [2013] presented an intuitive tool to sketch 3D virtual characters motion. The user draws a single line of action and the system fits the 3D model to the line shape. They define a body line: a linear subchain of the character's kinematic tree. The model is then posed so that the body line and the line of action match in orientation and position. The two lines are mapped to favor segments matching to rigid bones using a curvature-based energy. In later work, Guay et al. [2015] described temporal and spatial poses as well using "space-time curve" (see Figure 2.15(b)). With those curves, lines of actions can be computed using stroke outlines and velocity. Motion of certain part of the model are then computed to follow the pose. Several lines of action can be layered to over-sketch, edit a current path, change the shape, apply periodic motion and refine motion path. In this work, sketching enables users to define directly both the motion trajectory and the poses of the models through time (at specific key-frames).



(a) Motion Doodle **(b) Lines of action** **(c) Sketchimo**

Figure 2.15 – Motion path sketch interfaces. (a) Curves define the motion of the hand and upper body of the mode [Choi et al., 2016]. (b) Red line of action defines the local motion (flap its wings and tail) and the blue one, the global path of the model (fly up and down) [Guay et al., 2015]. (c) Animation adapts to the sketch to follow the trajectory [Thorne et al., 2004].

Similarly, Sketchimo [Choi et al., 2016] enables users to edit an animated motion by sketching strokes over the model. The interface displays the motion curve or path of the different joints (see Figure 2.15(c)) of the animated skeleton, providing the user with the possibility of either re-sketching the path of the join or re-sketching the position of the joint after at a certain keyframe. Other properties are available like editing the joint path and retiming the motion by changing the path width.

All of those techniques are successful but they focus on individual character motions and do not involve the interaction between several characters. Nevertheless it opens the question of sketching animations and building user-friendly interfaces. The previous sketching techniques we saw in the last two sections made use of simple interfaces only requiring the mouse and a screen to sketch. Other areas focus on finding new interfaces to better guide users onto the use of these tools. It is one of the question the HCI area works on. The work of Shen et al. [2018] used a multi-touch device to match a set of crowd

trajectories to a set of free-hand gestures. Walther-Franks et al. [2011] used such a device to animate 3D models.

To evaluate those diverse interfaces, the question of the evaluation of those tools becomes of interest as their primary goal is to be intuitive and quickly manageable. It is interesting to see how authors evaluate their interface.

3.3 User Evaluation

There appears to be limited work evaluating sketching in crowd simulation. This section presents those works, but also explores some formal approaches to evaluating user interfaces in general. When evaluating a new interface, authors attempt to quantify the usability of their method, the quality of the results as well as measuring more quantitative data like the task completion time. To do that they can for example compare their tool with existing techniques.

Sketch-based user interfaces for crowd simulation control have been evaluated using user studies. Oshita and Ogiwara [2009a] evaluated the effectiveness of a user interface for controlling the path of a crowd with an experiment. Four subjects were given a sample animation and asked to create it using the sketch-based system and a traditional interface. The processing time for each interface was measured and compared and the results showed that participants reproduced the animation more than ten times faster using the sketch-based approach. Similarly, Allen et al. [2015] had participants create scenarios involving crowd formations and movements using both a traditional control system and a sketch-based interface. The results showed that the sketch method was more accurate and easier to use, but it required more time to draw the shape of the formation. The effectiveness of any interactive interface, including sketch-based interfaces, with respect to its human-computing factors can be formally evaluated using several strategies developed specifically for this purpose. A detailed discussion of these strategies and their applicability in different situations is presented in the work of Ledo et al. [2018].

Sketch-based interfaces for various applications have been evaluated based on user experience. Xu et al. [2002] evaluated the usability of a user interface for conceptual/schematic design. The sketch-based interface was compared to a traditional button-based interface to design schematics-like sketches. Users found sketching to be more intuitive and faster compared to the button interface. Kara et al. [2007] evaluated a sketch-based 3D modeling system by conducting a study to determine three perceptual aspects: personal satisfaction, usefulness, and ease of use. Participants were asked to complete a short tutorial, design an object, and fill out a questionnaire. Users found the system intuitive and expressed positive opinions about the interface. However, some participants described the menus as cumbersome and no comparison with traditional modeling systems was made in this study. Tsiros and Leplâtre [2016] conducted a user study to evaluate the effectiveness and usability of a sketch interface for controlling a sound synthesiser. The study consisted in designing two soundscapes and answering a questionnaire. Overall, participants were satisfied with the interface, but also pointed out usability issues, such as the lack of options that exist with traditional image processing systems.

An important aspect in determining the quality of user interfaces is usability. A standard from ISO defines quality of use as: "the degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of us" [ISO/IEC 25010, 2011]. Seffah et al. [2006] summarizes several factors that are considered when measuring the usability of a system: Efficiency, Effectiveness, Productivity, Satisfaction, Learnability, Safety, Trustfulness, Accessibility, Universality and Usefulness. To assert that, usability questionnaire [Lewis, 2018; Sauro and Lewis, 2016] are often used. They are designed for the assessment of perceived usability, typically with a specific set of questions presented in a specified order using a specified format with specific rules for producing scores based on the answers of respondents. The questionnaire SUS [Brooke, 1995] consists of ten questions that users answer using a 5-point Likert scale from 'strongly disagree' to 'strongly agree'. SUS has been incorporated into commercial assessment toolkits and is referred to as an 'industry standard' [2013]. An advantage of this questionnaire is that a single value is obtained that represents the user's perception [Bangor et al., 2008]. This value ranges from 0 to 100, with lower values representing poorer usability. However, it remains open to interpretation at what value the system is considered usable. Bangor et al. [2009] conducted a study in which they added an eleventh question to determine the overall user perspective on the usability of the system. The question contains seven options ranging from 'Worst imaginable' to 'Best imaginable'. The purpose of the study is to provide an interpretation of SUS 's score by matching it with users' opinions. The study found that the adjective given by the participants was closely related to the score of SUS. The results ranged from 12.5 (worst imaginable) to 90.0 (best imaginable). An approach to evaluate and design user interfaces is the Goals, Operators, Methods and Selection rules model (GOMS) proposed by Card et al. [1983]. This user interface design model describes the knowledge required by the user to perform a task. Goals are presented by an action-object pair to identify the tasks that users try to complete. Operators are actions to be performed by the user. Goals and operators are similar, but the difference is that the operator is executed and the goal is a task to be accomplished. Methods are a set of operators needed to complete a goal. Lastly, selection rules are used to choose a suitable method to achieve the goal. GOMS models focus more on the design of graphical interface to improve performance. The contributions do no reside in the efficiency of the graphical interface per se but in the use of a new method underneath to design behavior. This is why we will focus mostly on usability and perception of the users in evaluating the usability if our proposed approach.

This section gave an overview of the different sketched based techniques. Sketched based techniques are overall more intuitive than other classical interface as demonstrated by Xu et al. [Xu et al., 2002]. We first could see that the diverse sketching tools used in crowd simulation did no enable the definition of new local interaction but on the control of other crowd simulation parameters (e.g. positioning, global trajectories). The second subsection allowed to take a peek into other sketching application, in particular animation. We saw that some powerful interfaces, such as sketchimo [Choi et al., 2016], enable the authoring of 3D body motion with a few strokes. Those interfaces are very intuitive and give a lot of control

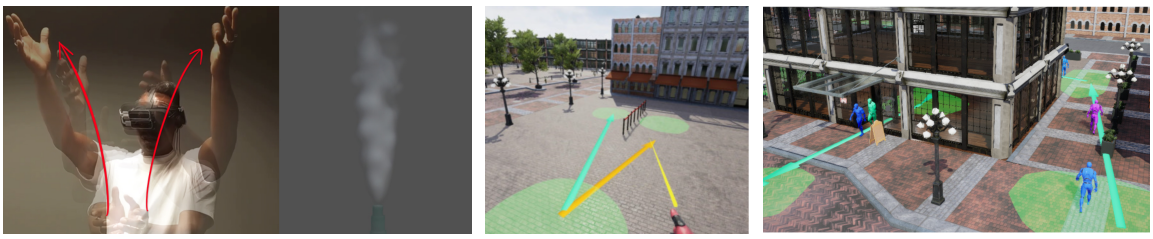
to the users. We think that similar techniques could by applied to intuitively design local interactions. By sketching a few strokes on a simple interface, users could simply sketch the local trajectories of agents when encountering. This is the key idea of this work. Moreover, as it is also important to evaluate how such approaches are appreciated by users we also plan to apply our approach to VR. With this in mind, the next section will review the detail further the interests of VR, for crowd simulation and for sketch-based techniques.

4 Virtual Reality (VR)

It is custom to model crowds simulation trajectories in 2D, using disk-shaped agents. however, more and more crowd model are now integrated in VR as it gives a way of evaluating the trajectories. VR is powerful tool to conduct experiments, whether they are perceptual studies where psychological parameters or diverse human behaviors are studied. Experiment can be easily reproduced and variables can be easily isolated. As reproducing humans behaviors in order to create compelling immersive scenarios is an important application of this work, it is interesting to analyze in more details the state of the art concerning VR. In relation to the previous section, we will first review the different sketch-based approaches for alternative interfaces such as VR. Then we will focus more on the different uses of VR in crowd simulations to better understand what could be the advantages of immersive applications of our work.

4.1 VR Sketching

In VR, Arora et al. [2019] studied sketching under different conditions to analyse the factors that affect the ability to sketch strokes in the air. The user study was divided into two experiments. The first task compared traditional and VR sketching. Participants were asked to draw a given shape on a solid surface, in the air in a VR environment, and on a physical surface while using the VR headset. The deviation between the target shape and the sampled sketch points were calculated. Traditional sketching showed the most accurate results. The second experiment investigated the use of visual guidance to guide aerial sketching in VR. A grid and a target line were used as aids. Participants achieved better results when the grid and target stroke were used together. More recently, Araro et al. [2019] also used immersive techniques to animate 3D objects. They used free hand gestures and their attribute in VR to trigger actions corresponding to animations (see Figure 2.16(a)). They defined a subset of hand-gestures, defined by the pose, the speed and the movement of the hand, that are mapped to those animations. A proof of concept was developed in which the key framing and interpolations are taken care of by the system internally. They presented a detailed study and analysis of the different interpretations of several hand gestures in the air used to represent different actions to animate an object. They seek to understand the preferences of different users for the various gestures they would use to specify a particular animation action. Their user study provides direction on how to make a VR platform-based animation interface



(a) Animation of a steam with gesture **(b) Crowd flow sketching** **(c) Crowd Flow application**

Figure 2.16 – (a) shows an example of mid-air gesture to animate a steam [Arora et al., 2019], (b) and (c) illustrate the interface of Bönsh et al. [Bönsch et al., 2018], users can sketch flows using a ray interactor (b) and virtual characters follow the flow (c).

more intuitive for each user.

Similarly, Dudley et al. [2018] also presented a detailed discussion of various techniques available for barehand sketching in the mid-air but focused more on Augmented Reality. This work provided a set of principles to follow when developing a mid-air hand-gesture based system and analyzed the utility of general mid-air interaction techniques to a user depending on the task given. They could for example demonstrate that bare-hand sketching by placing control points to create line segments was more accurate than freehand drawing and favored by participants. Bergig et al. [2009] also exploited Augmented Reality to animate mechanical systems from 2D sketches. To do so, they used a physical simulation system which processes the annotation given by the users, specifying parameters like force and friction.

Machado et al. [2009] evaluated and compared two interaction approaches, Sketch and WIMP (Windows, Icons, Menus, Pointing device), in 3D object modeling tasks. Results varied between the interfaces depending on the task. While the Sketch-based interface provided faster generation of 3D models and offered greater usability for the creation task, the WIMP interface showed greater usability for the editing tasks. Arora et al. [2017] compared traditional sketching on a physical surface with sketching in VR, with and without a physical surface on which the pen rests. Their user study shows that accurate sketching in VR is challenging and induces a lack of precision compared to traditional sketching. For example, they found that sketching on flat surfaces yields more accurate and fairer curves, and that horizontal plane alignment is the most accurate. Weise et al. [2010] conducted a user study in which participants repeatedly sketched primitive shapes (circles, squares, spheres, and cubes) in an immersive cave. Processing time, quality, and subjective mental workload of the tasks were measured. The results indicate that the quality of the sketches improved significantly over time, but the time required to complete a sketch did not change.

With regards to crowd simulation, Savenje et al. [2020] constructed an interactive table complemented with handheld AR via smartphones to interactively edit crowd simulation parameters. A model by Ju et al. [2010] converted example crowd motion to a continuous space, allowing for interpolations between types of motion. Shen et al. [2018] proposed a data-driven technique that chooses the appropri-

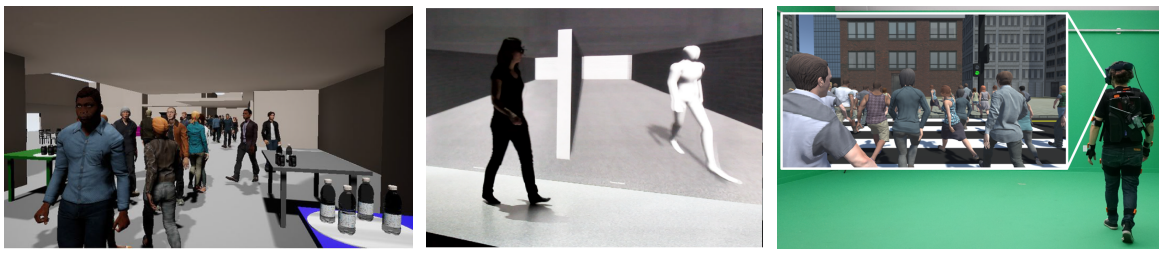
ate crowd motions based on multi-touch gestures. Bönsh et al. [2020] offered a proof of concept and the first prototype of a VR sketching tool for crowd simulation. In their survey, Bhattacharjee and Chaudhuri [2020] actually encourage future work on using immersive interfaces to exploit 3D sketching since it would drastically reduce the challenge of translating image space sketches onto 3D deformations. Nevertheless very little work on VR sketching were found to this day.

Even though crowd editing is not very present in VR, VR has been used to evaluate human behaviors in crowds. As we want to build up new local behaviors for VR scenarios, we need to take a look to the already existing use of VR in crowd simulation.

4.2 VR Crowd

We will now review briefly the work that combined Virtual Reality and crowd simulation. VR facilitates experiments in many ways (e.g. cost, preparation, and execution) and even allows the study of situations that would be impossible under real conditions (e.g. exact repeatability, mismatches). In this context, VR is often used to analyse and evaluate human behaviors in virtual crowds.

First, because of differences in how we interact in VR compared to real life (e.g. it is not always possible to naturally walk), it is important to evaluate potential effects of interacting in VR. For instance, Olivier et al. [2014] compared trajectories generated in a virtual environment and reference trajectories acquired in real-world situations. Their user study (Figure 2.17(b)) showed that regardless of virtual locomotion conditions, users attempted to generate trajectories that were similar to real-world trajectories. Another user study [2018] compared collision avoidance in VR and in real-world conditions. Users perceived the situation of interaction with the agent correctly, but the information about collision was shortly delayed compared to reality, and the position in the virtual environment was perceived with a slight offset (about 10 cm). The results also showed that all studied locomotion interfaces (e.g. joystick-based interfaces, whole-body locomotion metaphors) resulted in qualitatively realistic trajectories, with some quantitative differences in avoidance distances or strategies. Users however slightly over-adjusted their trajectories, possibly due to differences in distance perception in VR [Knapp and Loomis, 2003]. Later on, Meerhoff et al. [Meerhoff et al., 2018] compared virtual dyadic avoidance (1 immersed user vs. 1 agent) and virtual triadic avoidance (1 immersed user vs. 2 agents). Their observations revealed that triadic interactions depend strongly on how the reciprocal interactions between all walkers evolve, and even more so than dyadic interactions. In a similar context, Gerin Lajoie et al. [2008] compared experimentally obstacle avoidance in VR and real life and found that the personal space during locomotion had similar overall shape and side asymmetry, but is wider in VR than in real life. Moussaid et al. [2016] a replicated virtual emergency situations (with high and low stress) and found that participants maintained similar patterns and social conventions in shared 3D virtual environments and in real-life crowded situations. All of these works come to the same conclusion: VR can be used to qualitatively study the kinematics of human locomotion in goal-directed trajectories or collision avoidance tasks, although quantitative differences remain.



(a) Experiment with high density crowd. **(b) Experiment of collision avoidance in a CAVE** **(c) Cross Road experiment**

Figure 2.17 – Screen shot of experiment around crowd in VR. (a) Study of the impact of crowd density [Dickinson et al., 2019]. (b) Comparison between virtual and real world collision avoidance [Olivier et al., 2014]. (c) Cross road experiment comparing various settings [Koilias et al., 2020].

Due to the variety of VR setups available (e.g., CAVE vs. HMD), other authors studied the influence of VR setups on human behaviors such as their influence on gaze behavior during collision avoidance tasks between walkers [Berton et al., 2019]. They performed an experiment in both real-world and virtual environments using different virtual setups (e.g. cave, screen, HMD). Participants were asked to walk and avoid another walker (virtual or not), and their gazes were recorded using eye-tracking devices. Results show that collision avoidance behavior is similar regardless of the employed VR setup. However, the walking speed was lower when participants wore a HMD but had to move among physical obstacles. Regarding gaze allocations, participants looked at similar visual content in real and virtual conditions, but some differences remained, and HMD conditions' results were the closest to real-life.

VR was used by Rojas and Yang [2013] and Rojas et al.[2014a] to study small group formations, and Ahn et al. [2012] used a similar method to compare different collision avoidance algorithms in a street scene. In this case, the user remained stationary and observed the scene in a CAVE environment. Kim et al. [2016] used both 2D screens and HMDs to evaluate the similarity of simulations to videos of real scenes. The survey of Pelechano and Allbecky [2016] summarizes and discusses the topic of model validation using VR.

Heater [1992] identifies the sense of presence as the process of discerning and validating oneself existence in a virtual world, similarly than in real world. In this sense, presence can be used as a measure to validate virtual environments as a high feeling of presence asserts the convincingness of the environment. Pelechano et al. [2008b; 2008a] explored the field of presence for the evaluation of crowd simulation methods. In their work, presence was used as a measure to compare simulation methods (including social forces). It was evaluated using a tailored questionnaire and qualitative descriptions of video recordings. The work of Kyriakou et al. [2017] demonstrated that the representation of collision avoidance between the user and the virtual agents with some basic interactions between them can enhance the sense of presence and helped make the virtual environment appear more realistic and lifelike. In their studies, they also found that interaction with the virtual crowd can enhance the sense of presence. The types of interaction used in their work were limited to waving at the characters or verbally greeting the participant.

Another use of VR in crowd simulation is the study of participants' response to the physical proximity of virtual agents in VR environments. Proxemics has been studied with small groups of virtual figures walking toward the participant [Llobera et al., 2010]. In these experiments, physiological arousal was found to increase as the figures approached the participant and also as the number of virtual humans increased from 1 to 4. Through experiments, Sohre et al. [2017] investigated the role that collision avoidance between virtual agents and the VR user plays on overall comfort and perceptual experience in an immersive virtual environment. Participants were asked to walk through a dense stream of agents displaying or not collision avoidance. When collision avoidance was used, participants took more direct paths, with less jerking or backing away, and found the resulting simulated motion to be less intimidating, more realistic, and more comfortable.

Other works investigated whether human motion is affected by the motion parameters assigned to a virtual crowd in a virtual reality scenario. Dickinson et al. [2019] also reported the effects of crowd density on affective state and behaviour (Figure 2.17(a)). Results showed a significant increase in negative affect with density as measured by a self-report scale. Other studies worked on road crossing between virtual human and a participant [Koilias et al., 2020; Nelson et al., 2019]. They both created a crosswalk scenario with a virtual crowd and a participant crossing the road of a virtual city (see in Figure 2.17(c)). In this scenario, different densities, speeds, and directions of the crowd were tested. Nelson et al. [2019] showed that a large density and speed of the virtual crowd changed the participants' movement behavior to a significant extent. However, no change in human movement behavior was observed when the direction of the virtual crowd was examined. In the same scenario, Koilias et al. [2020] showed that the high-density, low-velocity, diagonal-direction situations associated with the virtual crowd had the greatest effects on participants' velocity, deviation, and trajectory length when walking in a virtual environment and surrounded by a moving virtual population.

Bönsch et al. Figure 2.16(b) and Figure 2.16(c)) of crowds and the environment obstacles (barriers) in VR. A brief user study took place and an average SUS-score indicated an adjective rating of "good", participants were all able to use the tool and sketch pedestrians flows. The promising results of this proof of concept and the previous work showed that there is an interest of having a sketching tool in VR to edit crowd local interactions. First it will enable the validation of the obtained sketched trajectories and then it will enable the design of immersive scenarios as well as quick editing and testing. Additionally, the previous studies highlight some interesting features of Virtual Reality. Immersive settings do not seem to interfere with the motion or perceptual behavior of a user interacting with a virtual human during locomotion. Even though there are some quantitative differences, the nature of the behaviors remains the same. This seems to be a fundamental point to consider new methods for sketching trajectories, where users could move in the environment and realize themselves the trajectories they want for the simulation.

5 Summary and Objectives

First, we would like to remind the reader that we are focusing on the concept of local interactions in the field of microscopic crowd simulation. As we mentioned earlier, local interactions subsume all local changes in the agents' trajectories by the environment (e.g., by other agents). In all scenarios involving multiple virtual humans (as well as the user), they are key components as they enable virtual humans to interact. They provide expressiveness, for example by providing information about relationships (e.g., ingroup, friendship), and interactivity, by enabling the user to interact dynamically with autonomous agents. Local interactions are part of a realistically populated world. To obtain a realistic simulation, they must be numerous and fit the desired scenario.

This review of the literature highlighted several missing aspects of the related work.

— The literature showed that the local interactions available in crowd simulation are very restrained and can be resumed to: collision avoidance, grouping and following. There is definitely a lack of other kind of interactions models and we acknowledge that defining rules for each local interaction would be extremely challenging.

— To address this issue, we explore the possible way of authoring a crowd. It turns out most of crowd authoring tool do not allow for refined control over the local interaction. They drive other parameters of the simulations hence focus on other scales. Nevertheless crowd authoring includes many models that enable users to easily design a scenario by, for example, letting them sketch the global path of the agents.

— In those techniques, sketching seems of main interest because it enables to interactively and intuitively sketch the behaviors of agents. In the diverse sketching interfaces that exist around animation, none of them addresses the sketching of agent interactions.

— To validate sketch-based interfaces, previous works highlighted that it was common to compare the evaluated technique with others that deliver the same types of outputs. In our case, the literature showed that no technique exists to design new local behaviors. Nevertheless, the review of the state of the art about usability evaluation brought new systems to light and suggests that we could adapt already existing SUS questionnaire to validate our approach. We also noticed that VR was frequently used to evaluate human behaviors, which would be of interest to our method as well.

— Studies have been made to study sketching in Virtual Reality, giving some guidelines (e.g. sketching on a horizontal 2D plane reduces imprecision). VR is widely used to validate models of crowd simulation or study the perceptual response of participant, however, very few tools exist to edit directly a crowd scenario in Virtual Reality.

Therefore, this thesis explores a novel way of designing local interactions between agents by proposing a new sketch-based approach. We were inspired by Patil et al. *navigation fields* approach [2011], which uses a grid that proposes an optimal walking direction at any point in the environment, possibly

based on sketches. We propose to use *Interaction Fields* (IFs), i.e., vector fields that describe the orientations or the velocities of agents, to design these local interactions, However, whereas navigation fields specify *global* paths through the environment, IFs specify how agents *locally* behave around other agents or obstacles. As such, IFs can move through the environment during the simulation, and they can change according to parameters.

We believe that the difficulty of creating new kinds of interactions has limited the variety of scenarios that can be simulated. As such, the list of local interactions proposed by Reynolds [1999] has never been substantially extended, despite the many developments in terms of simulation models. As discussed earlier, the agent concept can be applied in a more abstract way to model additional behaviors [Schuerman et al., 2010; Yeh et al., 2008], but this does not necessarily make new behaviors easy to design for non-experts. By contrast, IFs are specified in a purely visual way, enabling novice users to create new behaviors with relative ease. In addition, IFs take as input hand-drawn curves that are translated directly into a field. There are no restrictions on the shape or number of these curves, which means that any sketch can lead to a local interaction. This immensely expands the number of local interactions available to the user, which also increases the variety of possible scenarios. The remaining of this thesis will present the theoretical and practical applications of this new concept.

CHAPTER 3

INTERACTION FIELDS: OVERVIEW AND GENERAL DEFINITIONS

This thesis presents a novel sketch-based method for modeling and simulating many steering behaviors for agents in a crowd. Central to this is the concept of an interaction field (IF): a vector field that describes the velocities or orientations that agents should use around a given 'source' agent or obstacle. An IF can also change dynamically according to parameters, such as the walking speed of the source agent. This section will first give an overview of the IF framework to then focus on the primary definition of IFs.

1 System Overview

This section briefly describes the simulation details that are required for understanding the framework of IF, and gives an overview of how all components fit together.

Our crowd simulation takes place in a bounded 2D environment $\mathcal{E} \subset \mathbb{R}^2$ with $m \geq 0$ *obstacles* $\{O_i\}_{i=0}^{m-1}$ and $n \geq 0$ *agents* $\{A_i\}_{i=0}^{n-1}$. Each agent or obstacle can emit IFs to influence neighbor agents. At each frame of the simulation, agents detect the IFs they are influenced by and sum them to deduce a desired velocity. This will be detailed further in Section 2.

At any point in time, each agent A_i has a position $\mathbf{p}_i$, a velocity $\mathbf{v}_i$, and an acceleration $\mathbf{a}_i$, with the usual relations between them:

$$\frac{d\mathbf{p}_i}{dt} = \mathbf{v}_i, \quad \frac{d\mathbf{v}_i}{dt} = \mathbf{a}_i.$$

It is common, in the 2D space, to implement obstacles as simple polygons and to model each agent A_i as a disk with radius r_i. However, our method does not explicitly rely on these implementation choices. Each agent A_i therefore also has an *orientation* $\mathbf{o}_i \in \mathbb{S}^1$ (a 2D unit vector) that represents the direction that A_i is facing.

The simulation uses discrete time steps (*frames*). In each frame, every agent A_i computes a new value for its acceleration $\mathbf{a}_i$, which will induce a change in its velocity $\mathbf{v}_i$ and position $\mathbf{p}_i$. We refer to the process of updating $\mathbf{a}_i$ as *local navigation*. IFs can be used together with other navigation algorithms, as well as independently. As explained in Section 1, the process of computing $\mathbf{a}_i$ can be based on algorithms for, e.g., path following, collision avoidance, and group behavior. The crowd simulation models mentioned

in Chapter 2 all follow this overall structure, therefore which navigation algorithms are actually used will not impact the following of our discussion.

In most crowd simulations, the orientation $\mathbf{o}_i$ is simply a consequence of the agent's velocity. By contrast, we are interested in controlling $\mathbf{o}_i$ explicitly. We use IFs for this purpose as well. Thus, we will present IFs as a way to control the *velocities and orientations* of agents.

Figure 3.1 shows an overview of the proposed system that combines IF design, crowd simulation, and character animation. The details per component will be provided throughout this thesis. First, a user

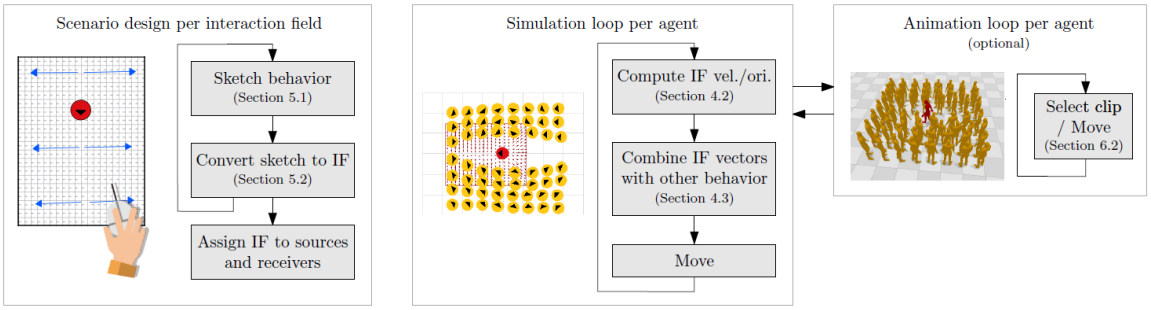


Figure 3.1 – Outline of a complete simulation system with IFs.

sketches an IF in an *editing tool*, which we will describe in Chapter 4. Users can draw elements onto a canvas, and this sketch is automatically converted to an IF. Users can then inspect the resulting agent behavior in the simulation (to be described below) and return to the sketching phase if they wish. They can iterate until they obtain the desired agent behavior. To set up a complete simulation scenario, users should also specify which objects *emit* the IF (as *sources*) and which agents *respond* to it (as *receivers*).

Next, the sketched IFs are applied to the *simulation* in the way presented in Section 1. For ease of comprehension, we will discuss the simulation first and the IF editor second. In each simulation frame, every agent A_i performs a sequence of tasks. First, A_i (if receiver) should respond to the IFs emitted by nearby sources, which results in an IF *velocity* and IF *orientation* proposed by these IFs. Next, A_i can combine this result with other behavior such as collision avoidance, resulting in a new velocity and orientation to use. Finally, A_i moves and rotates according to the computed vectors.

It is also possible to combine the 2D simulation output with animated 3D characters. Although this is not the focus of our work, it is an important and non-trivial component for many applications. We will discuss the options and our implementation in Section 1.

This section defines the concept of an *Interaction Field* (IF) which can be used in a crowd simulation to control the velocities and orientations of agents relative to other objects.

2 General Definitions

Overall, a single IF describes either the *velocities* or the *orientations* that agents should use in the vicinity of a particular *source*, which we denote by s. We also state that the source *emits* the IF. A source

can be an agent, an obstacle, or any other aspect of the environment that should induce a certain kind of behavior. Simply put an IF is a vector field: an assignment of a vector to each point in a subset of space. The domain D is a subset of the 2D Euclidean space $\mathbb{R}^2$, where the IF is active, it is defined relatively to the source.

Velocity interaction fields

Because an IF prescribes behavior around a source s, we define it in a Cartesian coordinate system relative to s, with s located at the origin $(0,0)$ and oriented towards the negative y-axis. Using this, a *velocity IF* with source s and domain $D \subset \mathbb{R}^2$ is a vector field

$$VIF_{s,D} : D \to \mathbb{R}^2$$

that maps any position $\mathbf{p} \in D$ to a 2D vector $VIF_{s,D}(\mathbf{p})$, indicating the velocity that any receiver agent should use at this position.

Orientation interaction fields

Likewise, an *orientation IF* is a vector field

$$OIF_{s,D} : D \to \mathbb{S}^1 \subset \mathbb{R}^2$$

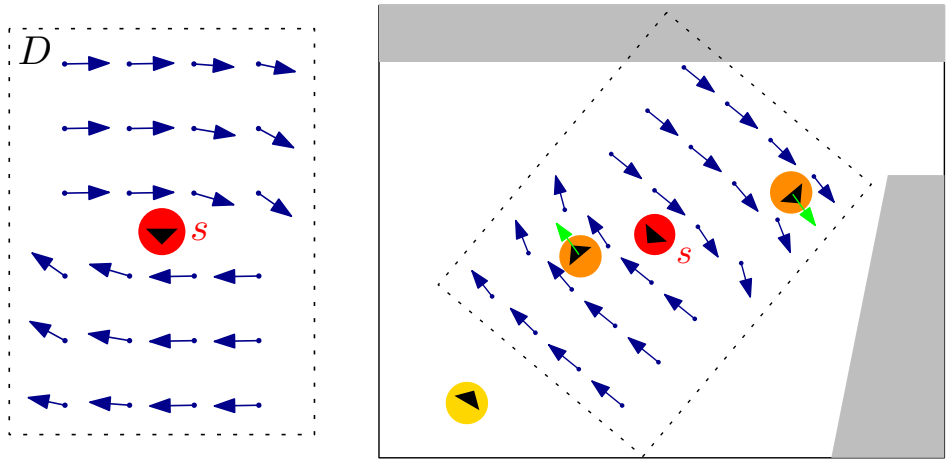
that maps any $\mathbf{p} \in D$ to a 2D unit vector $OIF_{s,D}(\mathbf{p})$ that agents should use as their orientation. Figure 3.2(a) shows an abstract example of an IF. Whenever it does not matter whether an IF concerns velocities or orientations, we will use the notation $IF_{s,D}$. Note that an IF prescribes a vector for *all* points in the domain D. Our figures will only show sample velocities for the sake of illustration.

Parametric interaction fields

We defined the basic IF, a vector field that can map orientation or velocity vectors relatively to a source. Those two types of fields are the building block of the IF framework. However, most of the time, the motion of a person can change according to the environment properties (e.g., other agents speed, density). It is also important to be able to create IFs that can display different behaviors depending on different simulation parameters. We therefore extend IFs so that they can change during the simulation according to parameters, which we call parametric IFs. These parameters may affect both the vectors and the domain of the IF. In other words, a parametric IF encapsulates different 'ordinary' IFs for different parameter values.

Formally, a *parametric IF* with $l \in \mathbb{N}$ scalar parameters can be described as a function

$$PIF_s : \mathbb{R}^l \to (D^* \to \mathbb{R}^2)$$



(a) IF definition **(b) Mapped onto the environment**

Figure 3.2 – (a) An IF is a vector field (shown here in blue) that prescribes velocity or orientation vectors in a domain D around a source object s (here: the red agent). (b) During the simulation, the IF is mapped onto the environment to match the current position and orientation of s. Other agents (in orange), if they are receivers, use this mapped IF to compute a velocity or orientation (in green), which they can apply directly or combine with other IF prescriptions or other navigation algorithms. Agents outside the domain D (in yellow) are not affected.

where the resulting velocity vectors and the domain D^* now also depend on the l parameter values. Theoretically, there is no limit on the number of parameters. In this work, though, we create IFs based on user sketches, and we will use at most one parameter to keep the design process intuitive. We will now discuss two specific types of parametric IFs that are supported by our sketching tool: keyframing and interpolation of parametric IFs, and parametric IFs description in relation with objects (any element that can serve as a source).

Keyframes and interpolation

One way to specify a parametric IF is to define IFs for a few specific values of a single parameter. These IFs then act as *keyframe IFs*, called *KIF*, at runtime, and the IF for any other parameter value is defined via linear interpolation between the two nearest keyframe IFs.

For example, Figure 3.3 shows a parametric velocity IF with two keyframes, where the parameter is the speed of the source agent s. When s is standing still, agents will gather around s in a circle. When s is moving at a certain predefined speed, agents will attempt to follow s from behind. There are infinitely many vector fields for the source speeds in-between. During the simulation, agents will use an interpolated field that matches the current speed of s.

Next to the speed of the source agent, other examples of parameters could be the width or height of a source obstacle (to apply the IF to obstacles of various sizes), the current simulation time (to model

behavior that changes over time), or the local crowd density around an agent (to model density-dependent behaviors). A parameter could also represent an agent's state of mind, such as its hastiness or the amount of panic it experiences.

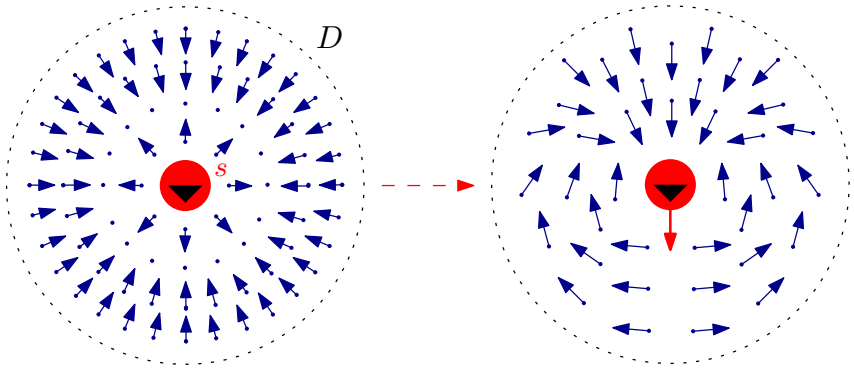


Figure 3.3 – Parametric interaction fields based on the source speed. Example of an IF that depends on the speed of its source agent s.

The simulation never needs to fully compute an interpolated IF. In any simulation frame, an agent only needs to compute a single output vector for each parametric IF in range. Formally, let there be k keyframe IFs associated to k parameter values: $\{\langle q_j, KIF_j\rangle\}_{j=0}^{k-1}$, ordered by increasing q_j parameter values. Assume for now that all keyframe IFs have the same domain D. Given a parameter value q, the parametric IF is defined as follows for any position $\mathbf{p} \in D$:

— If $q < q_0$, then $PIF_s(\mathbf{p}) = KIF_0(\mathbf{p})$.

— If $q \geq q_{k-1}$, then $PIF_s(\mathbf{p}) = KIF_{k-1}(\mathbf{p})$.

— Otherwise, $q_j \leq q < q_{j+1}$ for some $j \in [\![0, k-2]\!]$,

$$PIF_{s,q}(\mathbf{p}) = (1-\lambda) \cdot KIF_j(\mathbf{p}) + \lambda \cdot KIF_{j+1}(\mathbf{p}),$$

where $\lambda = (q - q_j)/(q_{j+1} - q_j)$.

If two subsequent keyframe IFs have *different domains*, we require that any domain in-between can be obtained via linear interpolation as well. For example, this is the case if the domains are both axis-aligned rectangles or both disks. The IF vector $PIF_s(\mathbf{p})$ is then only defined if $\mathbf{p}$ lies inside the interpolated domain.

The concept of keyframe IFs can be extended to more than one parameter. In that case, each keyframe will be associated to a point in a higher-dimensional parameter space. As mentioned earlier, though, we will focus on single-parameter examples because these are still relatively intuitive for non-expert users to design.

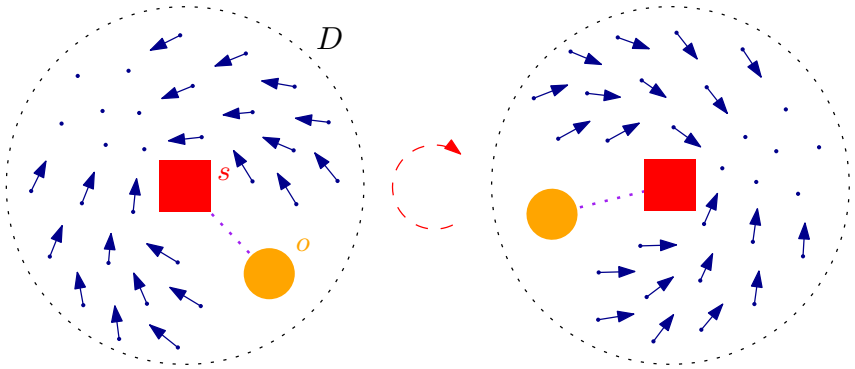


Figure 3.4 – Parametric interaction fields based on angular relation. Example of an IF that depends on the angular relation between the source s and an obstacle o.

Relations between objects

A parameter of an IF could also be a relation between two objects a and b. Possible examples are the distance between a and b, $\|\mathbf{p}_b - \mathbf{p}_a\|$, or the angle between the vector $\mathbf{p}_b - \mathbf{p}_a$ and the x-axis.

As a concrete example, Figure 3.4 shows a velocity IF that lets agents move behind a source s (typically an obstacle) to hide from another object o (typically a specific agent A_k). In this specific example, the parameter of the IF is the angle α between the vector $\mathbf{p}_o - \mathbf{p}_s$ and the x-axis. The effect of α is simply to rotate the IF: it does not affect the IF vectors themselves in the local frame, but it only changes how the IF is mapped onto the environment. In contrast to regular IFs, this mapping now no longer depends on the orientation of the source s.

Note that this example can theoretically be combined with keyframe IFs, where the keyframes determine the IF vectors and the angular relation determines the mapping onto $\mathcal{E}$. The result would be a parametric IF with two parameters.

In our IF editor, for simplicity, an angular relation between s and another object o is currently the only object relation that users can draw. A *distance-based* relation between two objects could be implemented with the help of keyframes again: the user specifies which two objects determine the distance parameter, and then they draw keyframes with different distances between these objects.

In line with these two examples, our program for sketching IFs contains two options for letting users draw a parametric IF. For example, users can specify 'keyframe' IFs for specific parameter values, after which any IFs in-between can be computed on the fly via interpolation. Angular relations between objects can be specified in this drawing tool as well. Chapter 4 will present these options in detail. For now, it suffices to know that parametric IFs exist, and that parametric and non-parametric IFs are used similarly in the simulation.

In deed, now that we presented a general overview of the method and the general concepts, the next chapter will focus on the design of IFs and more precisely will explain how IFs are sketched.

CHAPTER 4

SKETCHING INTERACTION FIELDS

As mentioned earlier, the analysis of the state of the art has shown that sketching is an optimal method for designing trajectories and, in particular, fields. It provides the user with a great leval of flexibility by using hand-free curves as input, allowing a large number of different IFs to be created. Moreover, sketching does not require any specific knowledge and can express ideas without precision, making sketch-based interfaces easy to learn even for beginners. This chapter presents how to sketch interaction and, more specifically, how to convert hand-drawn strokes into vector fields. The developed graphical interface for sketching IFs is presented as well as the mechanisms for designing more complex IFs.

We have developed a graphical interface in which users can intuitively sketch IFs, called 'IF editor'. This chapter describes the components of this 'IF editor' and their mathematical meaning for the IF being drawn. Figure 4.1 shows the editor with all the main functions, which was developed in C++ using Qt creator. This chapter also describes how the user, in practice should navigate through the editor.

1 Main Elements of the IF Editor

The user can draw three main types of elements in the IF editor, and a sketch can contain multiple elements of each type.

An **object** is anything that can serve as the *source* of an IF. In our IF editor, it can be an agent (visualized as a disk) or a polygon (which can represent an obstacle or something more abstract). One of the objects on the canvas can be marked as the source object s. Other (non-source) objects can be drawn as a visual aid, or to help define a *parametric* IF. We will explain this further in Section 3. To add an object, the user simply has to select the type in the object layout and click on its wanted position in the canvas. The size of a polygon can then be modified.

A **guide curve** is a curve $C_i : [0,1] \rightarrow \mathbb{R}^2$, with an associated magnitude and direction drawn by the user, that exactly specifies the IF vectors along that curve. For any point that lies on C_i (i.e. if the position $\mathbf{p} = C_i(\mathrm{t})$ for some parameter value t), the curve prescribes a vector $\mathbf{c}_i$ with magnitude v_i and direction $\frac{d}{d\mathrm{t}}C_i(\mathrm{t})$, tangent to the curve at each point of the curve. Figure 4.2(a) contains two examples of a guide curve. In the final IF, the vector $IF_{s,D}(\mathbf{p})$ at any position $\mathbf{p}$ will be an interpolation of the vectors proposed by all guide curves. Section 2 will describe this interpolation. In the IF editor, users can draw a guide curve as a piecewise-linear curve or as a freehand curve.

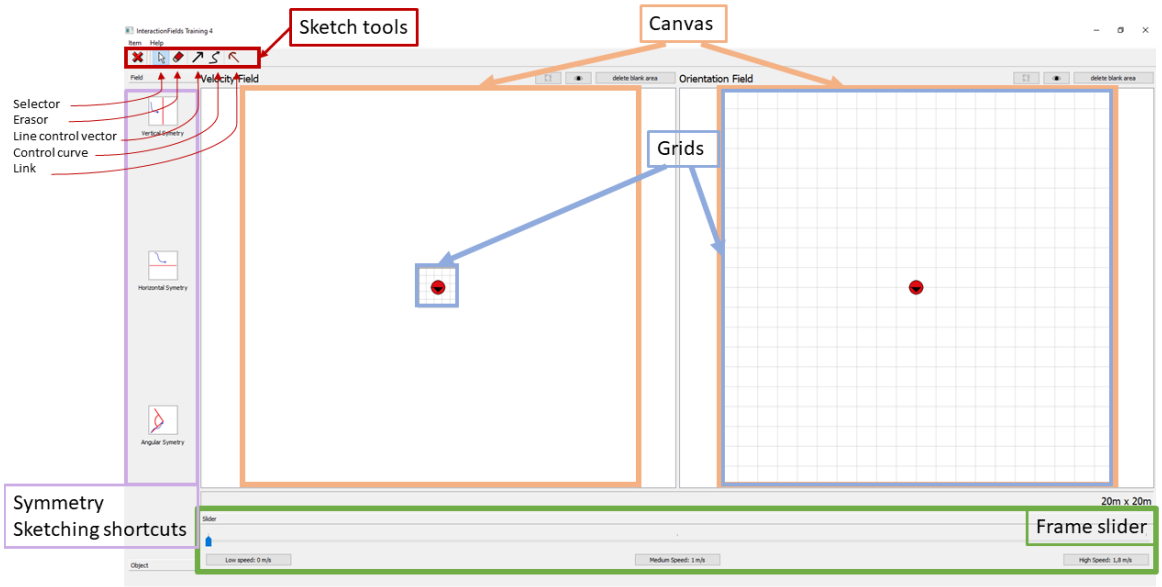


Figure 4.1 – IF editor overall interface, where the user can sketch an orientation field or a velocity field. The red rectangle contains the tools to sketch on canvas (in orange) a field. The user can: select guide curves to edit them (mouse icon), erase vectors (eraser icon) of the grid (blue), delete guide curves (red cross), add straight guide lines (black arrow) or free handed shaped guide curves (curved arrow). It can also link objects together (red link). Shortcuts are available so that users can more quickly draw the field by duplicating guide curves according to the desired symmetry (purple). In green, the keyframe slider enables users to navigate through all the frames of a parametric fields.

To sketch the guide curve on the editor, a user can choose between two types of curves to facilitate the drawing (guide curves and guide straight lines), which will simply change the number of handle available to edit the guide curves. A guide line simply has two handles where the guide curve has a number of handles proportional to its length. To sketch a curve, the user simply hand-sketches a scribble line, as displayed in Figure 4.3(a), that will be mapped to a guide curve. The number of handles can be reduced or increase for more precision over the shape of the curve, using the editor. Those handles are used to move around parts of the curves easily, as shown in Figure 4.3(b).

Finally, a **zero area** is a region $H_j \subset \mathbb{R}^2$, $j \in \mathbb{N}$, where the IF is 'empty', i.e., areas that propose either a zero vector or nothing at all. This is useful for many scenarios, most notably for letting agents stop moving when they have reached a certain area. For velocity IFs, H_j prescribes the zero vector, meaning that an agent will stand still when it is located inside H_j. For orientation IFs, H_j acts as a hole in the domain D, i.e. as a region where the IF does not propose any specific orientation. Figure 4.2(a) contains one example of a zero area. Note that zero areas always have priority over guide curves (cf. Section 2 for more details). In the IF editor, users can draw zero areas with a paintbrush tool, or they can erase IF vectors after converting their sketch to a grid. Figure 4.3(c) shows how vectors can be erased

using the eraser tool. Once the zero area has been defined, the corresponding vectors are saved separately.

In the IF editor, the user starts by defining a bounding shape D^b, which will serve as the IF domain D. The IF editor then creates a rectangular canvas on which the user can draw. In practice, the user simply uses a dedicated windows on which he/she can enter the dimension of the domain. Next, the user can draw elements onto the IF canvas to specify parts of the IF. Section 1 will describe these elements in more detail.

Finally, the program can convert a drawing into a discretized IF: a rectangular grid of vectors, with a user-specified level of precision. This conversion process, which we will describe in Section 2, uses an interpolation scheme to fill in any regions where the user has not drawn. Of course, the user can adapt this result if desired, by drawing additional elements and then rebuilding the grid. He/she can also choose and edit the dimension of the grid, as well as its resolution. This grid or matrix is used to display the results of the Interaction Fields in the cells of the matrix.

2 Converting a Sketch to an IF

After a user has drawn a sketch, it is then necessary to convert it to be usable as an IF.

Interpolating between guide curves

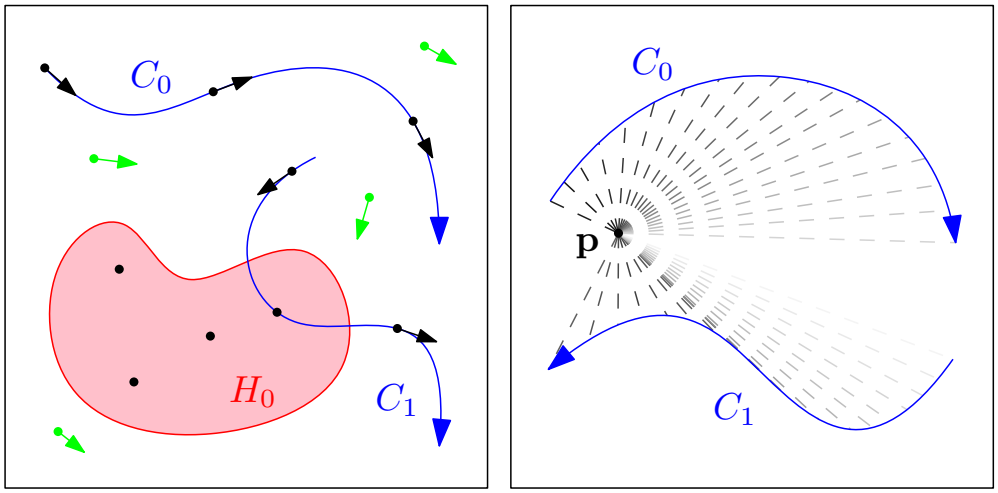
An important aspect of the conversion is to 'fill in' the IF for areas where nothing has been drawn. To infer a meaningful IF vector for any point $\mathbf{p}$ in the domain D, we interpolate between all vectors proposed along all guide curves. This interpolation is based on *inverse distance weighting* [Shepard, 1968], a commonly used method for estimating values among scattered data points.

Given a set of c guide curves $\mathcal{C} = \{C_i\}_{i=0}^{c-1}$, the estimated IF vector for a point $\mathbf{p} \in D$ (which may or may not lie on a guide curve) is the following:

$$\mathbf{u}(\mathbf{p},\mathcal{C}) = \frac{\sum_{i=0}^{c-1}\left(\int_0^1 w(\mathbf{p},C_i(t))\cdot\mathbf{v}_i(t)\,dt\right)}{\sum_{i=0}^{c-1}\left(\int_0^1 w(\mathbf{p},C_i(t))\,dt\right)} \tag{4.1}$$

Here, $w(\mathbf{p},\mathbf{q}) = \frac{1}{\|\mathbf{p}-\mathbf{q}\|^{\kappa}}$, and $\kappa \in \mathbb{R}^+$ is a power parameter that determines how strongly the influence of a curve point decays along with the distance to $\mathbf{p}$. Preliminary experiments have led to a use of $\kappa = 1.9$ in our implementation. This yields IFs where all vectors are meaningful even with a small number of guide curves. We remind the reader that users can still edit their drawing after the conversion, in case the resulting IF does not match their expectations.

In practice, the integrals in Equation (4.1) can be approximated by sums, using regularly spaced sample points on each curve. Figure 4.2(b) gives a visual impression of this interpolation scheme. Note



(a) IF editor elements **(b) Interpolation**

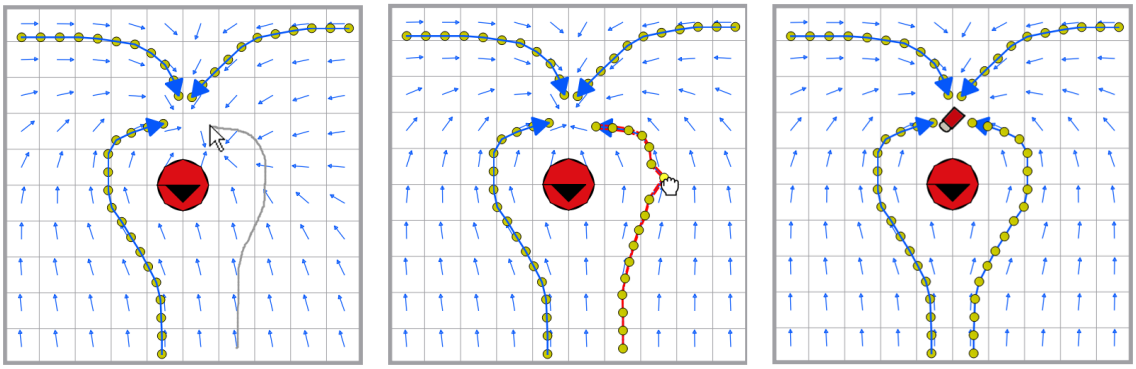
Figure 4.2 – Concepts of the IF editor. (a) The user can draw guide curves (blue) and zero areas (red) to specify IF vectors; example vectors are shown in black. IF vectors for points in-between will be interpolated (green). (b) For any point p outside all zero areas, the IF vector is a weighted average of all vectors along all guide curves, where weights depend on the distance to p.

that the number of samples does not affect the curve's importance; it only determines the precision by which C is approximated.

This type of interpolation has several useful properties. First, if a point $\mathbf{p}$ lies exactly on a curve point $C_i(t)$, then $\mathbf{u}(\mathbf{p}) = \mathbf{v}_i(t)$, and other curves do not matter (unless $\mathbf{p}$ is visited multiple times due to curve intersections). Second, if there *are* intersections between or within curves, they do not need to be handled explicitly: the interpolation scheme will simply produce an average vector at an intersection point. Third, the distance-based decay of a curve's influence is only *relative* and not *absolute*. Moving away from a curve point $C_i(t)$ does not 'shrink' the vector that it proposes; it only reduces the relative weight by which it is taken into account.

Figure 4.4 shows a number of examples of IFs for different guide curves. Note that the simplest example contains only one straight guide curve, and its IF contains a uniform vector everywhere in D.

This interpolation is very well suited for the sketching of the vectors of the grid because the results of each guide curves is intuitive to the users. We however noticed that the amplitude given by this interpolation is not necessarily intuitively understood by users, and therefore decided to provide users with the possibility of directly defining the vectors amplitude. We also ensured that the amplitude of the velocity fields depend on the object inside the fields. For Velocity IF, users can decide on the speed of each agent entering the field by scaling the vectors amplitude on the maximum speed of the agent. To do that, they select a color in the dedicated windows, displayed Figure 4.5, and paint over the vectors of the grid. The color selection is through a gradient representing a percentage of the agents maximum speed: fast is red (the highest red correspond to the maximum speed of the impacted agent) and low speed is blue



(a) Draw guide curve **(b) Edit guide curve** **(c) Create zero area**

Figure 4.3 – (a) Once the guide curve tool had been selected, a user can freely hand draw a guide curve of any shape: the gray line is the scribble hand free sketch and the blue lines are the corresponding guide curve. (b) Then the user modifies the shape of the curves, dragging the guide curve's handles. (c) To create a zero area, the user must select the eraser and erase the vectors which should have a amplitude of zero.

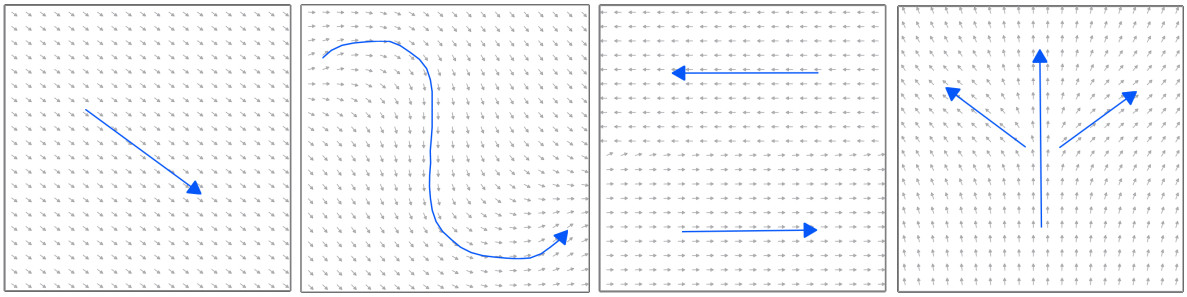


Figure 4.4 – Examples of guide curves (shown in blue) and their resulting IFs. The gray arrows are the IF vectors (following from the interpolation scheme of Section 2) on a 20×20 sample grid.

(the slowest speed of the impacted agent). This bring a new variety to the fields, the speed scales on the impacted agents' properties. Indeed, elderlies and cars do not have the same maximum or minimum speed.

In the case of orientation IFs, the amplitude does not impact the final result since only the direction is needed: the vector's magnitude is hence fixed to 1 so that C_i proposes unit vectors.

Computing the final IF

We now define the overall interaction field that can be obtained from a source s, a bounding shape D^b, a set of guide curves $\mathcal{C} = \{C_i\}_{i=0}^{c-1}$, and a set of zero areas $\mathcal{H} = \{H_j\}_{j=0}^{h-1}$.

For a *velocity* IF, the domain D is equal to D^b, and the velocity function $VIF_{s,D}$ works as follows for any point $\mathbf{p} \in D$:

— If $\mathbf{p}$ is inside any zero area $H_j \in \mathcal{H}$, then $VIF_{s,D}(\mathbf{p}) = \mathbf{0}$.

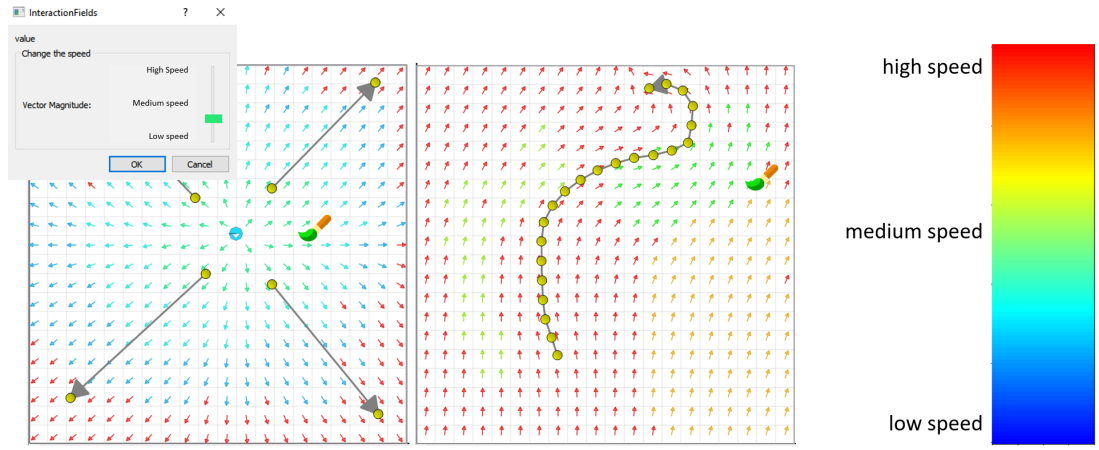


Figure 4.5 – To decide on the speed an agent a should have inside a field, a user can paint over the field's vectors attributing them colors. Each color corresponds to a percentage of a's maximum speed.

— Otherwise, $VIF_{s,D}(\mathbf{p}) = \mathbf{u}(\mathbf{p},\mathcal{C})$ (see Equation (4.1)).

For an *orientation IF*, recall that zero areas are treated as holes in the domain. In other words, the domain D is equal to $D^b - \bigcup_{i=0}^{h-1} H_i$, i.e. the set of points that is not covered by any zero area. For any point $\mathbf{p}$ in the remaining domain D, the final orientation function normalizes the interpolated vector from Equation (4.1) to unit length.

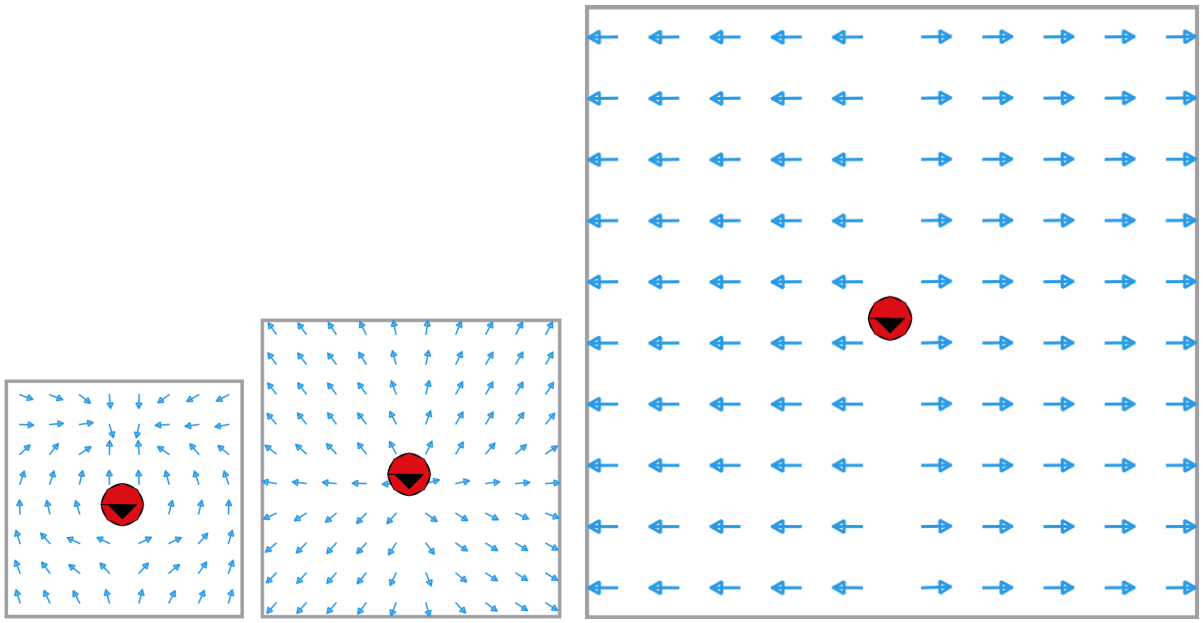
The IF editor finally converts a drawing to a grid by computing $IF_{s,D}(\mathbf{p}_i)$ for a set of regularly sampled grid points $\mathbf{p}_i$. The resulting grid of vectors can be used in the crowd simulation.

3 Sketching Parametric IFs

We conclude this chapter by explaining how users can draw *parametric* interaction fields. As a reminder from Section 2, a *parametric IF* is an IF that depends on additional scalar parameters.

We have discussed parametric IFs based on *keyframes* and based on *relations between objects*. To draw a parametric IF based on *keyframes*, the user can simply draw separate IFs and specify the corresponding parameter values. To do this on the editor, the user moves the slider seen in Figure 4.1, which represents the possible value of the current parameter. Once a field is designed for a value of the slider, a key frame is automatically defined, as illustrated in Figure 4.6. This process resembles very much the one used in animation tools to draw keyframe animation. The user can then navigate along the slider and interpolated fields will be displayed. To draw a parametric IF based on an *object relation*, the user can draw a line-segment connection (a *link*) between the two relevant objects. As mentioned in Section 2, the IF editor currently only supports a link between the source s and another object o, and this link implies an angle-based relation between s and o. This link can be seen in the tool panel in Figure 4.1.

The next chapter will focus on how to use the now sketched fields to propel agents and obtain a final simulation.



(a) Key parameter: 0 (b) Key parameter: 0.5 (c) Key parameter: 1

Figure 4.6 – Examples of guide curves key IF sketched for different key values of a selected parameter.

4 Discussion

The IF editor converts sketches to interaction fields using inverse-distance weighting of guide curves. While this type of interpolation has several advantages (as explained in Section 2), there may be situations where the result is not yet ideal. For example, a point in the IF is currently always influenced by *all* guide curves to some extent, even by (parts of) curves that are far away. To prevent this, we could let users control the influence distance of a guide curve, so that far-away points are ignored. Finally, we have deliberately decided that velocity IFs prescribe absolute velocities, and not relative velocities (or even acceleration vectors). Such alternative representations could make certain scenarios easier to model, but they would strongly reduce the method's intuitiveness and user-friendliness. Similarly, applying concepts such as incompressibility may lead to smoother velocity fields, but not necessarily to a more intuitive user experience.

CHAPTER 5

IMPLEMENTATION AND ANIMATION

The third and forth chapters of this thesis respectively defined an IF and explained how to create one through sketching. This chapter focuses on how to include IFs into a framework to locally steer agents in crowd simulation. The first section will focus on the crowd simulation framework, that first aims at obtaining trajectories in 2D. The second section will explain how we can then use such trajectories to obtain a simulation in *3D* including animated virtual characters.

1 Implementation

1.1 Applying IFs During the Simulation

Once the IFs are sketched, they are used directly as input of a 2D simulation software; from that we can obtain the trajectories of each agent. The first section of this chapter will focus on integrating IFs to such a simulation.

As mentioned in Chapter 3, an IF is defined relatively to a source s. During the crowd simulation, the position $\mathbf{p}_s$ and orientation $\mathbf{o}_s$ of s can change over time, especially if s is an agent. To apply the function $IF_{s,D}$ at runtime, the IF should first be translated and rotated to match the current values of $\mathbf{p}_s$ and $\mathbf{o}_s$. Informally, if we see $IF_{s,D}$ as a pre-defined 'picture' around s, we should always line up this picture with how s is currently positioned and oriented. We call the result the *mapped IF*, and denote it by $IF'_{s,D}$. Figure 3.2(b) shows an example.

It is important to note that this mapping can remain implicit during the simulation. There is no need to translate and rotate complete IFs at runtime. For any position $\mathbf{q} \in \mathcal{E}$, we can easily compute the relevant IF vector $IF'_{s,D}(\mathbf{q})$ by applying the inverse mapping to $\mathbf{q}$. Therefore, an agent A_i can easily compute the IF vector $IF'_{s,D}(\mathbf{p}_i)$ for its position $\mathbf{p}_i$ by applying the inverse mapping to $\mathbf{p}_i$.

One special case is worth mentioning: if the source s is the entire environment $\mathcal{E}$, then $D = \mathcal{E}$ as well, and there is no mapping to apply during the simulation ($IF'_{s,D} = IF_{s,D}$). Such an IF is similar to a navigation field [Patil et al., 2011]: it prescribes vectors for the whole environment, and not for the neighborhood of one specific object.

The purpose of an IF is to model a single type of behavior around a source, so most simulations will feature multiple IFs at the same time. As part of the scenario design, the user should specify for each IF which objects *emit* it and which agents *respond* to it. This means that users can choose which IF impacts

which agent as well. Consequently, it is possible for agents to respond to only *some* IFs and to ignore others, i.e. to model different behaviors for different agents.

At any moment in the simulation, each agent A_i should respond to the relevant interaction fields emitted by nearby sources. To this end, let $\mathcal{I} = \{VIF_{s_j,D_j}\}_{j=0}^{k-1}$ be the set of all *velocity* IFs to which A_i can respond *and* that currently have $\mathbf{p}_i$ in their mapped domain. The *IF velocity* $\mathbf{v}_i^{\mathrm{IF}}$ for A_i is defined as a weighted average of the vectors that these IFs propose:

$$\mathbf{v}_i^{\mathrm{IF}} = \frac{\sum_{j=0}^{k-1} VIF'_{s_j,D_j}(\mathbf{p}_i) \cdot w_j}{\sum_{j=0}^{k-1} w_j} \tag{5.1}$$

where w_j are weights to prioritize between IFs, e.g. to increase the influence of an IF as an agent moves closer to the source. It will often be sufficient to use $w_j = 1$ for all j. For orientation IFs, we define *IF orientation* $\mathbf{o}_i^{\mathrm{IF}}$ for A_i analogously, the only difference being that we explicitly normalize the result.

1.2 Combining IFs With Other Simulation Components

There are several ways to combine the IF velocity and orientation with other simulation aspects (such as collision avoidance). In most traditional crowd simulations, the behavior of each agent A_i per simulation frame is already subdivided into multiple steps:

1. Compute a *preferred velocity* $\mathbf{v}_i^{\mathrm{pref}}$ that would send the agent towards its goal, possibly with the help of a global path.
2. Compute a *new velocity* $\mathbf{v}_i^{\mathrm{new}}$ that stays close to $\mathbf{v}_i^{\mathrm{pref}}$ while following local rules for collision avoidance, group behavior, etc. This yields an acceleration $\mathbf{a}_i := (\mathbf{v}_i^{\mathrm{new}} - \mathbf{v}_i)/\Delta t$, where Δt is the length of this simulation frame in seconds. Both $\mathbf{v}_i^{\mathrm{new}}$ and $\mathbf{v}_i$ are typically clamped to a maximum walking speed $v_i^{\max}$ to prevent unrealistically large velocities.
3. If the agent is currently colliding with other agents or obstacles, compute *contact forces* $\mathbf{f}_i^{\mathrm{c}}$ and update the acceleration: $\mathbf{a}_i := \mathbf{a}_i + \mathbf{f}_i^{\mathrm{c}}/m$, where m is the agent's mass (usually 1).
4. Update the agent's velocity and position via Euler integration method:

$$\mathbf{v}_i := \mathbf{v}_i + \mathbf{a}_i \cdot \Delta t, \quad \mathbf{p}_i := \mathbf{p}_i + \mathbf{v}_i \cdot \Delta t.$$

To add *velocity* IFs to the system, we have the choice between letting the IF velocity $\mathbf{v}_i^{\mathrm{IF}}$ (Equation (5.1)) influence an agent's *preferred* velocity (in step 1) or its *new* velocity (in step 2). We will use the first option in our implementation. This allows for an intuitive combination of IFs and collision avoidance, where IFs play an 'advising' role and collision avoidance has the final say. The navigation fields of Patil et al. [2011] are also used in this way.

Thus, we use IFs as an alternative way to compute a preferred velocity $\mathbf{v}_i^{\text{pref}}$. It is also possible to let $\mathbf{v}_i^{\text{pref}}$ depend on IFs *and* on other factors (such as goal reaching) at the same time. We will use this in some of our example scenarios; Section 1.3 will describe the underlying simulation settings. As mentioned earlier, most crowd simulations do not explicitly control the agent's orientation $\mathbf{o}_i$. Thus, *orientation* IFs can be trivially added to the simulation loop in a separate step:

5. Compute the IF orientation $\mathbf{o}_i^{\text{IF}}$. If $\mathbf{o}_i^{\text{IF}} \neq \mathbf{0}$, update the agent's orientation as $\mathbf{o}_i := \mathbf{o}_i^{\text{IF}}$. Otherwise, keep $\mathbf{o}_i$ unchanged, or update it in a 'traditional' way, e.g. as an average of $\mathbf{v}_i^{\text{pref}}$ and $\mathbf{v}_i^{\text{new}}$.

1.3 Crowd Simulation Framework and Settings

We have implemented the IF editor and an IF-enriched crowd simulation in platform-independent C++. To convert a drawing to an IF, guide curves are sampled at curve-length intervals of 0.1 meters. Our IF-enriched crowd simulation has been implemented by extending UMANS [1], an existing real-time agent-based crowd simulation framework [van Toll et al., 2020], to support interaction fields. The simulation represents each IF by a grid. We compute an IF vector using bilinear interpolation between the nearest grid cells. For parametric IFs based on keyframes, recall from Section 3 that any interpolated IFs are not explicitly computed. However, we sometimes *visualize* an interpolated IF for the sake of illustration.

In line with other research, our simulations use Euler integration method and a fixed frame length $\Delta t = 0.1$ s. Each agent has a disk radius of 0.3 m, unit mass, a preferred speed of 1.3 m/s, and a maximum speed of 1.8 m/s. For contact forces in case of collisions, we use the model of Helbing et al. [2000] with coefficients $K_{\text{ag}} = \frac{5000}{80}$ for agent forces and $K_{\text{obs}} = \frac{2500}{80}$ for obstacle forces. These values are commonly used in literature when the agents have unit mass.

Next to these overall simulation settings, each agent A_i will use one of the following *behavior profiles*:

— *IFs-Only*: A_i uses the IF velocity $\mathbf{v}_i^{\text{IF}}$ directly as the preferred velocity $\mathbf{v}_i^{\text{pref}}$ and as the new velocity $\mathbf{v}_i^{\text{new}}$. There is no additional goal reaching or collision avoidance.

— *IFs+GoalReaching*: A_i computes $\mathbf{v}_i^{\text{pref}}$ as the average of $\mathbf{v}_i^{\text{IF}}$ and a velocity that sends A_i to a predefined *goal* at the preferred speed. There is no collision avoidance, so $\mathbf{v}_i^{\text{new}} := \mathbf{v}_i^{\text{pref}}$.

— *IFs+RVO*: A_i computes $\mathbf{v}_i^{\text{pref}}$ using IFs. It then computes $\mathbf{v}_i^{\text{new}}$ using the RVO algorithm for collision avoidance [van den Berg et al., 2008], using the default settings suggested by its authors. Overall, RVO looks for a velocity close to $\mathbf{v}_i^{\text{pref}}$ that has a low collision risk.

— *UserControl*: A_i receives $\mathbf{v}_i^{\text{pref}}$ and $\mathbf{v}_i^{\text{new}}$ directly from a user (e.g. via keyboard or controller input). The agent still receives contact forces in case of a collision. In our figures and videos, user-controlled agents will always be visualized in red.

Of course, and most importantly, each scenario will use its own specific *interaction fields* to model specific types of behavior, and different agents can emit and receive different IFs.

1. For more information about UMANS: https://project.inria.fr/crowdscience/project/ocsr/umans/

Computational performance In terms of performance, our scenarios are too small for meaningful time measurements. However, the software that we use as a basis can simulate tens of thousands of agents in real-time. It is well-known in our community that the *nearest-neighbor queries* between agents is the least scalable simulation task, which will dominate the overall running time when the crowd is large. *Collision avoidance* can also be an expensive task, depending on the algorithms used. Relatively, IFs have very little impact on the simulation complexity. In a simulation frame, each agent A_i performs simple arithmetic operations for each perceived IF. This is similar in complexity to e.g. force-based collision avoidance. Thus, nothing prevents IFs from being usable for large crowds.

2 Character Animation

Section 1.3 explained how IFs can be coupled with a crowd simulation software to obtain 2D trajectories. To generate a final 3D simulation, the trajectories must then be animated. We will present in the second section the process we chose to animate 3D characters using IFs.

2.1 Coupling With Character Animation

To visualize our results using animated 3D characters (as displayed in our supplementary videos listed Table 6.1), we have connected our IF crowd simulation to the Unity game engine. Synchronizing a 2D simulation (of 10 FPS) with an animated 3D scene (of a higher framerate) is not a trivial task. There are at least two options to choose between:

Simulation priority: Let the 3D characters move exactly to the positions produced by the crowd simulation, and use interpolation to fill in the additional animation frames. For body animation, apply a suitable motion clip to each character, accepting possible artifacts such as footsliding.

Animation priority: Use the *output* of the simulation as *input* for an animation system that chooses an appropriate motion clip per character. The chosen animation determines where a character actually moves, and this overrides the simulation results.

The first option is often used in crowd simulation papers, whenever a perfect correspondence to the simulation is more important than animation accuracy. The second option is popular for controllable characters in games, where the animation should be smooth and natural. It can also help filtering out motion for which no animation clip exist, such as fast backward motion or sudden rotations. This solution gives more realistic results and corrects uncanny output trajectories of IFs.

For crowd simulations with IFs, while we see use cases for both options, in our supplementary videos listed Table 6.4, we consistently use the second option, based on a Unity plugin for *Motion Matching* [Animation Uprising, 2020].

3D Integration To convert the 2D scenarios into 3D simulations, we integrated our framework with Unity in 3D. Figure 5.1 illustrates the components of this new 3D framework. We use a library version

of our 2D simulation software *UMANS* (shown in purple in Figure 5.1). The library is responsible for managing the IFs parameters, computing the final output of IF according to the scene settings, and sending the updated states (e.g., orientations and velocities) of the agents to the 3D simulation. Using this library enables us to take advantage of the options already built in without further development. For this reason, we can still combine the IF technique with other crowd simulation algorithms that provide a wide range of models, as described in Section 1.3. The 2D simulation library is also responsible for updating the parametric IF according to changes in the 3D simulation (e.g. position and velocity of the user). The main looping simulation engine is the 3D simulation component called *CrowdMP* [2] which was developed in Unity (orange in Figure 5.1) to originally run crowd simulation experiments in 3D. IF has been included into a new version of *CrowdMP* that now monitors the simulation. It first builds the scene and sends all the information to the 2D simulation library. At each frame, *CrowdMP* sends the new position of the agents and the player to the *UMANS* dll and requests updates from *UMANS*, which are then sent to the *Motion Matching* animator. We will now describe the functions of this animator component.

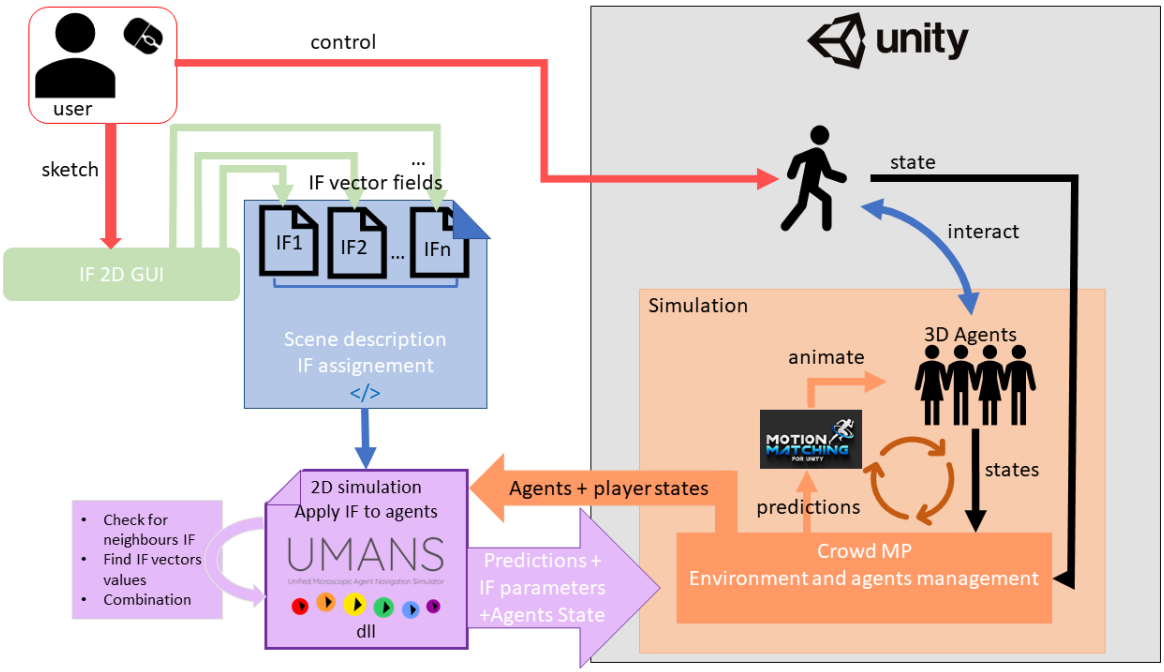


Figure 5.1 – IF framework for 3D simulation. Each block of a different color is a component of the final framework. In green, the 2D Editor to sketch IF in 2D. In purple, the 2D simulation software *UMANS* to apply IFs. The xml describing the scene is in blue. Finally, the gray block includes components built in Unity, the 3D simulator in orange, including *CrowdMP* and the Motion Matching animator. The inputs of the user are in red.

2. For more information about *CrowdMP*: https://gitlab.inria.fr/OCSR/crowdmp/crowdmp

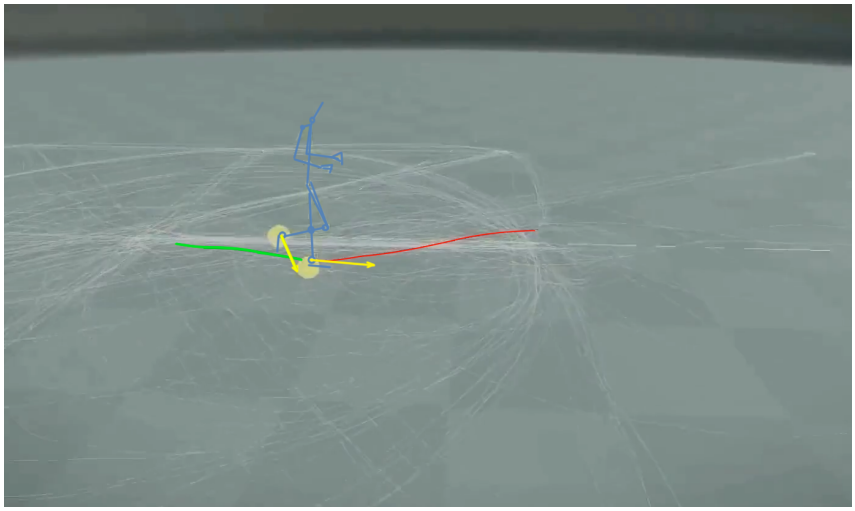


Figure 5.2 – Motion matching compares poses according to the position (yellow sphere) and velocity (yellow arrows) of key joints, the future (red line) and past (green line) trajectories. The white lines represent the overall data set trajectories.

2.2 Animating Characters Using Motion Matching

Traditionally, character animation uses an animation state machine [Unity, 2016]. In this model, each state of the state machine is a set of animations that are manually selected. The transitions between these states are also defined manually by setting arbitrary parameters to eventually transition from one state to another. A state is usually triggered by an action (e.g., pressing a key makes the character jump). Rather than predefining animation clips for each state, with all the transition, Motion Matching matches an input trajectory to a fitted animation. This significantly reduces the time it takes to animate a sequence. The basic idea is to use an animation database that is periodically searched for the frame that best matches a set of properties, such as the current position of the character's feet or the future trajectory. When the best matching frame is found, the animation playback continues from that point and blending between animations is inserted to eliminate discontinuity. To find the correct animation sequence through the motion capture data, Motion Matching uses a cost function (also called an objective function) to find the best matching frame. The cost function is the sum of several components represented by Figure 5.2:

— Comparison with the future trajectory: to ensure that the input trajectory will correspond to the matched animation.

— Comparison with the current pose: to ensure the continuity of the animation. To do that they perform position and velocity matching but only considering key joints. For example, a biped locomotion would require left and right foot joints.

— Comparison of the history of the previous position and velocity to match against a forward and backward time horizon. This allows to distinguish between monotonic (straight lines) and non-monotonic (turns) motions as well.

To optimize the method and avoid searching through all the motion data, Buttnër and Clavet [2015] suggest to use a nearest neighbor search algorithm that would output the nearest frame neighbours with the closest trajectory.

Rather than specifying the fine-grained animation logic via a state-graph, Motion Matching enables animators to specify the properties of the animation which should be produced. At the low level control, animators can choose the dataset but also the key joints to match with animation clips. Animators can also choose the ratio between the quality of the motion and the responsiveness to the input trajectory through a number of parameters. When combined with large amounts of data, Motion Matching proves to be a simple and effective way of dealing with the vast number of possible transitions and interactions that are required by a modern AAA video game. Additionally, since Motion Matching plays back the animation data stored in the database as-is, with only simple blending and post-processing such as inverse kinematics applied, quality is generally preserved, animators retain a level of control, and the behaviour can be tracked and debugged with appropriate tools. Finally, since it has minimal training/pre-processing time, adjustments can often be made in real-time, resulting in quick iteration time.

2.3 *MxM* Plugin for Unity

This method was quickly adopted by many studios due to its simplicity, flexibility, controllability, and the quality of the motion it produces. Claassen [Animation Uprising, 2020] proposed a version of Motion Matching implemented in the Unity game engine called "*MxM*", available directly from the Unity Store [3]. This "on the shelf" solution seemed appropriate to us because it delegates character animation entirely to the animator. Figure 5.3 shows how *MxM* is connected to the rest of the IF framework. The input that *MxM* needs is the trajectories of the virtual characters. For this, we directly used the output of our 2D *UMANS* simulation system, that are processed from IFs sketches. These inputs had to be modified to meet the requirements of *MxM*. They essentially consist of the position and orientation of each agent for several n frames in the future. The number of frames can be modified given the computational cost of each step into the future. To reduce this cost, we decided to take as a prediction only the results of IFs, without taking into account any other behavior component, as if we were always in the case of "IFs-only" profile behaviors (as explained in Section 1.3). Figure 5.4 illustrates this process. Another solution is to assume that the future velocity of each agent is linear. This solution is used, for example, with a user-controlled agent, but the resulting predicted trajectory and thus the quality of responsiveness in the 3D scene is lower. We found that at high speed and assuming that the virtual character maintains its linear velocity, characters tend to just continue in straight lines until getting out of the fields and not answering to the simulation anymore.

After receiving the predicted trajectory, *MxM* searches for the next best animation clip from the previous pose applied to the scene (see Figure 5.3), and as previously explained, more particularly based

3. Link to MxM Unity store page:
https://assetstore.unity.com/packages/tools/animation/motion-matching-for-unity-145624

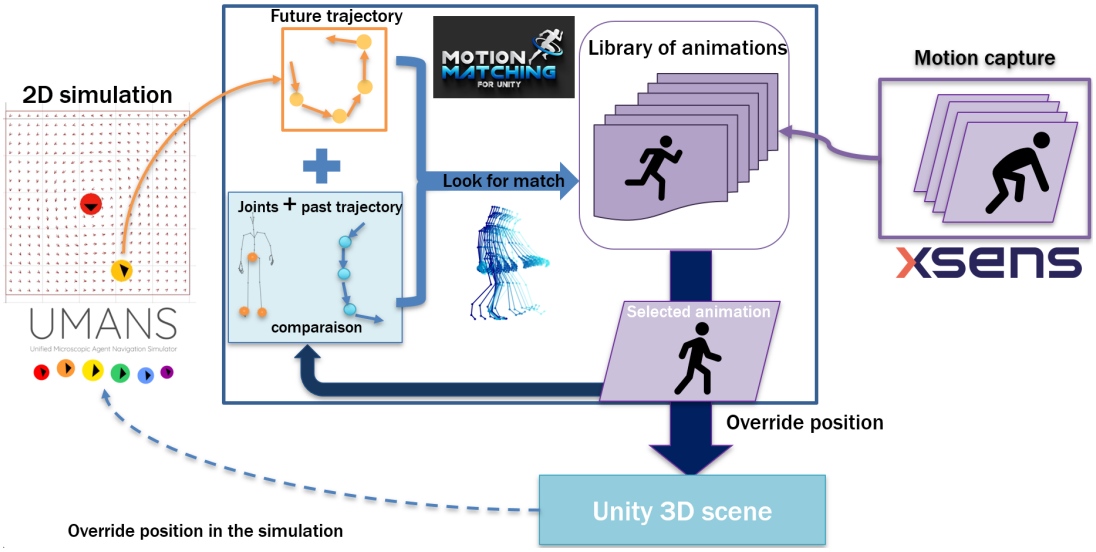


Figure 5.3 – *MxM* integration into our framework. First, motion must be captured (using X-sens or others) to be processed and build up a animation library. From this library, *MxM* will choose the animation matching the positions and orientations predicted by *UMANS* dll.

on the selected joints position and velocity and the past trajectory of the pose. Once the next pose is selected, it is applied onto the 3D character, then we can loop back to *UMANS* by sending the current position of the character and overriding the simulation. This is not a mandatory phase and depends on the scenario. The drawback of the approach is however that characters can be stuck in a local minima and stop moving if the current position of the characters in *UMANS* is overwritten by *MxM* or that the errors between the two are two big inducing unrealistic trajectories.

After combining *MxM* and *UMANS*, we obtained satisfying results, responsive and realist enough to animate the scenarios faithfully. We can also use *MxM* to increase the realism of our simulation. Interaction Fields is a “what you see is what you get” sketching technique, which means that the sketched fields (after interpolation of the guide curves) are then directly used in the simulation without any modification. The output simulated trajectories are directly related to what users draw. In addition, to give users freedom in designing agent behavior, our technique deliberately controls velocities and orientations separately, and it allows these vectors to change quickly during the simulation. Also, users are free to draw velocity IFs with ‘local minima’ where agents end up standing still (e.g., using zero areas or by drawing curves that cancel each other out). This freedom of design may cause agent behavior that is not realistic or ‘human-like’, which may be seen as a disadvantage for some applications, especially when animated 3D characters are involved. This can result in users drawing unrealistic trajectories, which is especially noticeable with combined orientation and velocity IFs. Potential examples include unrealistic acceleration, velocities, unfeasible successive direction... Indeed, in 2D simulations; making an agent running backward at full speed is easily doable. Using Motion Matching therefore enables us to filter out

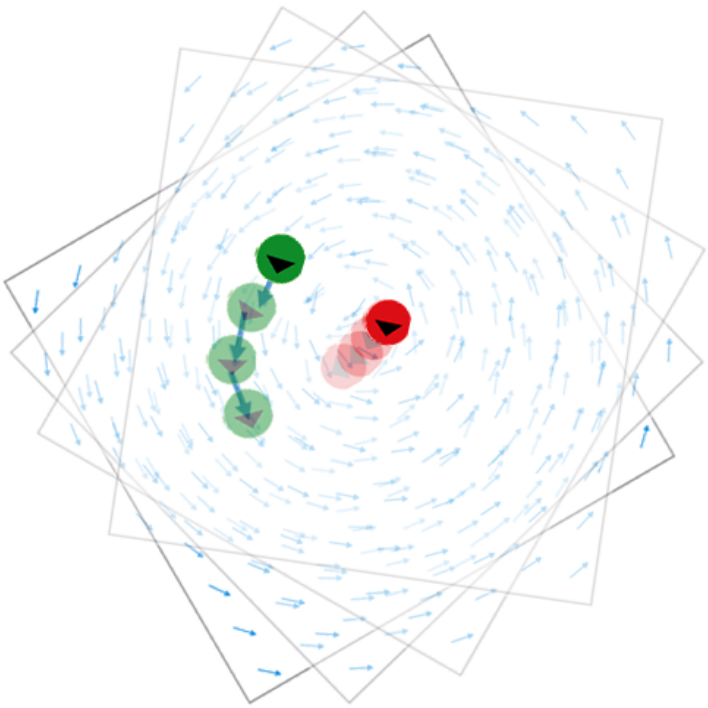


Figure 5.4 – To build the prediction required from Motion Matching, several steps forwards are computed over Interaction Fields outputs. The future positions and orientations (transparent agents in red and green) are stored and send to *MxM*

those unrealistic trajectories by using data from a real life animation capture dataset. In the aforementioned example, the final outcome could either be a final animation where the character runs backward at a lower speed available in the dataset, or an animation with a change of direction where the character now runs forward, according to the animator choice.

Note that filtering IF trajectories with realistic motions is controversial and was left as is in the 2D simulation. In *UMANS* 2D simulation system, users have the final say in the final simulation, even if the results are unrealistic. In the 3D simulation, *MxM* will have the final say in the trajectories, giving users less control over the simulation (see Figure 5.1).

Even if users have less control on the simulation when adding 3D animation, *MxM* provides several control levels trough parameter. E.g. the animator can choose the ratio between the realism of the motion and the reactivity to the input trajectory, but the various importance of the terms of the cost function can also be changed. The animator can put more or less weight on each joint's velocity or position as well as the trajectory. Other tuning are possible like the way of blending between the motion, the pose to favor or to trigger, etc. However all this parameter tuning is actually very complex to handle, as mentioned in Section 2. *MxM* is a tool that is not necessary accessible to novices and each setting should be tested out for each different scenarios, which can be time-consuming.

2.4 Library of Motion Capture

As described in the previous section, we have created an implementation where the 3D character animation system has the final say over a 2D agent's trajectory, thus overriding the simulation result if necessary. The realism of the trajectory is ensured by taking real life motion as data for animating the resulting IF trajectories. This way, the (transitions between) motions of an agent are limited to what the provided animation clips support. Note that this also allows for the personalization of behavior by providing different animations per character.

Motion Capture (also referred to as MoCap) is the process of digitally recording the movement of people. In our examples, to be presented Section 1, the motion capture data were realized using the X-sens system [X-sens, 2000]. By putting on a suit, constituted of wearable inertial sensors at every important body articulated joints, the orientation, altitude and positioning data of all the joints are captured and transcripted, using a biomechanical model, into 3D animation. Ubisoft presented the motion capture strategy they underwent for the gameplay of "for Honor" [Ubisoft, 2016] during a 2016 GDC (Game Development Conference) talk. During this talk, Kristjan Zadziuk [Ubisoft Toronto, 2016] presented "dance cards", i.e., navigation routes that actors have to follow to obtain optimal motion capture (motion data that cover a wide range of locomotion without too much recordings). Claassen [Animation Uprising, 2020] used those information to create a documentation about the important dance cards to record to have the most complete data set possible for *MxM*. Note that those dance cards, as *MxM* in general, focus a lot on action type video game animation and gameplay. This means that they focus on very dynamic locomotion and animation (like strafing) that are very specific to AAA video games, which we therefore adapted to fit our scenarios (e.g. by focusing more on small steps).

For our examples, one motion capture session lasts approximately two hours for a locomotion session, including the equipping the actor. Several dance cards are necessary to record for each actor.

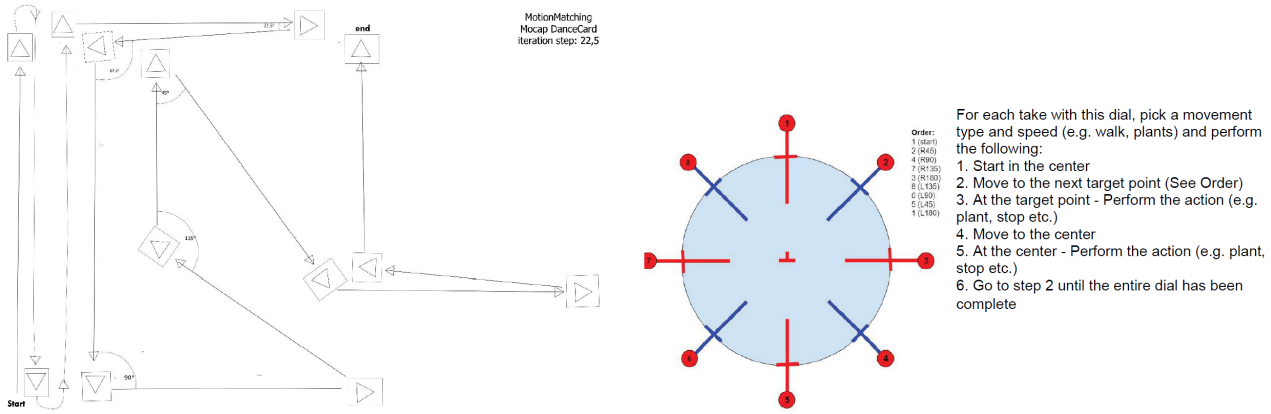
— Locomotion dance card: to realize classic walking cycles with various turns and speeds. Each angular turn should be incremented by a specific angle from -180 to 180: 45 degrees is the minimum recommended. 22.5 degrees provides better results at the cost of significantly more effort. The former was chosen, following the dance card displayed in Figure 5.5(a).

— Transition dance card: realize first ellipses at different speeds then including acceleration and deceleration.

— Snake dance card: the idea is to register quick turn and avoidance movements at different speeds and while strafing. The actor starts at one end of the capture volume and runs to the other end in a winding 'S' pattern and repeat this action at different widths for different speed.

— Slow step dial: here the actor has to achieve slow subtle steps in all directions by starting in the center of a circle and go to equally distributed points on the circle as in Figure 5.5(b).

— Dial strafing: the idea is to strafe in all directions, taking a similar map than the previous one.



(a) Dance Card Locomotion (22,5° angular step) **(b) Dial Slow Step**

Figure 5.5 – Example of dance cards for *MxM* motion capture requirement.

Such motion capture sessions are tedious and require time-costing investments before being able to judge the results. The goal of IFs is to refine interactions symbolized by precise trajectories hence the importance to have animations that match the most the input trajectory. To ensure that, an important amount of animations is required. Ideally, those previous dance cards should be performed by diverse actors if we want credible variety in the results or, at the very least, it should be performed with various motions if the scenario requires certain properties of the agents. For example, in the hide and seek scenario (to be presented Section 1.1), one would expect that the hidden character should act differently than the seeker character (being afraid or silencious). This also requires specific motion capture session with different walking directions, postures... However, the dance cards are good material and in the total of the 3 years of this PhD, only 15 hours were necessary to obtained the 3D scenarios used in Chapter 6.

CHAPTER 6

RESULTS AND EVALUATION

After presenting the theory and implementation details of IFs, this chapter focuses on the practical results of our approach. First, Section 1 presents qualitatively a number of complex scenarios realized with our method. Then, Section 2 presents the evaluation of the usability of IFs through a user study that will be presented.

1 Demonstration of Results

This section shows the capabilities of interaction fields in a number of example scenarios. Our main purpose is to demonstrate specific features of IFs (such as the use of parameters), and to show that these can easily be combined into more complex scenarios.

For each scenario, we will show the input IFs created in our editor, as well as screenshots of the resulting simulation. All simulation screenshots include a grid with cells of 1×1 m, to illustrate the scale of the environment. For visualization purposes, we also show several IFs mapped onto the environment. Recall from Section 1.1 that the simulation itself does not need to compute any mapped IFs.

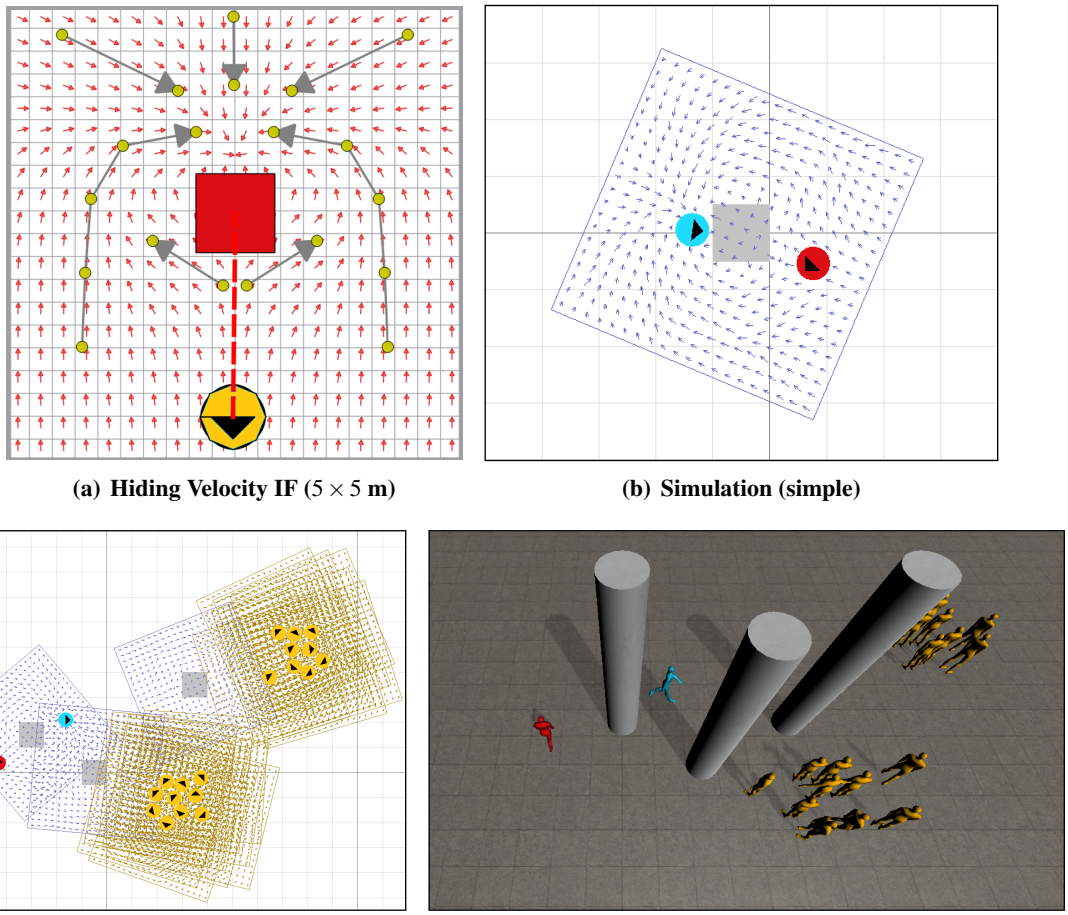
We also invite the reader to watch the supplementary videos listed Table 6.1, which shows several results in motion, including the IF design process and combinations with 3D character animation.

This section will illustrate IFs in application using simple IFs, as well as parametric IFs. One PIF depends on the relationships between a seeker's position and a hiding spot and another one adapts to the source speed to repel further the neighbours when the source is moving fast.

1.1 Scenario 1: Hide and Seek

Our first scenario uses an angle-dependent parametric velocity IF to let an agent hide behind an object. This IF, shown in Figure 6.1(a), was drawn using 7 guide curves and a rotation link. In the simplest version of the scenario, one obstacle O emits this IF, with a user-controlled agent A_0 as the linked object.

An agent A_1 with the *IFs-Only* profile is receiver of the IF. As the user moves A_0 around, A_1 automatically hides behind O depending on where A_0 is located. Figure 6.1(b) shows a screenshot of the simulation.



(a) Hiding Velocity IF (5×5 **m)**

(b) Simulation (simple)

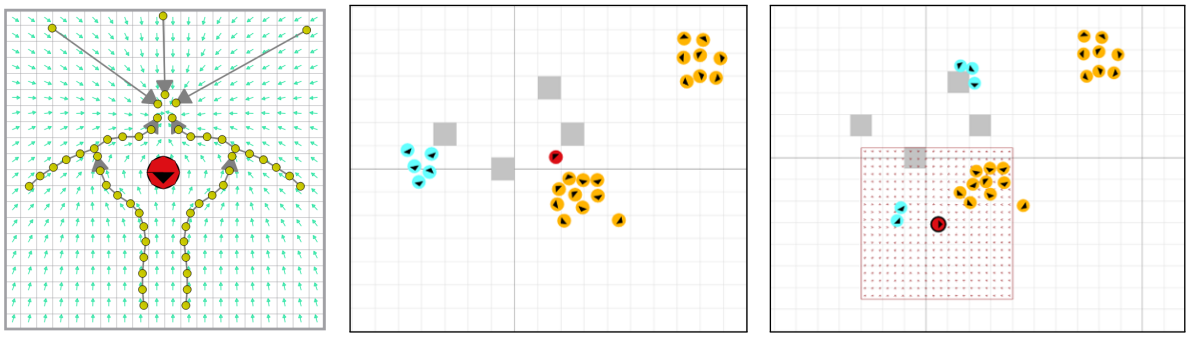
(c) Simulation (extended)

(d) 3D visualization

Figure 6.1 – Results for the *Hide and Seek* scenario (Section 1.1). (a) A velocity IF with a rotation link (red dashed segment) between the source (square in red) and a second object (orange). Guide curves are shown in grey. The red vectors indicate that the impacted agents will go at their full speed.(b) A simulation where the blue agent uses this IF to hide from the user-controlled red agent. (c) A simulation where the blue agent can hide behind all obstacles and orange agents, each emitting the same IF. (d) A 3D impression with the two main agents on the left.

Hide and seek

The scenario can be made more complex without introducing any new IFs. In the extended scenario shown in Figure 6.1(c), we have added several obstacles and agents (with the *IFs+RVO* profile) that all emit the same IF. Consequently, the agent A_1 hides behind whichever object is nearby, treating obstacles and agents in the same way. The extra agents do not respond to any IFs, but they use collision avoidance to make way for the user if necessary. Figure 6.1(d) and Table 6.1 video 1 visualize the scenario in 2D and 3D.



(a) Following Velocity IF (7×7 m) **(b) Simulation where all blue agents hide** **(c) Simulation with 2 followers**

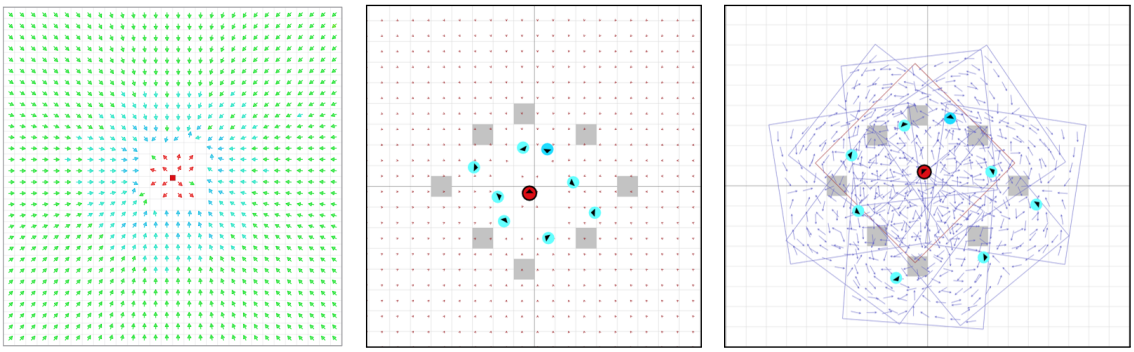
Figure 6.2 – Results for the *Several Hiders* scenario (Section 1.1). (a) A velocity IF with guide curves in grey and the field define a local minimum behind the source (in red). The vectors were colored in green so that the impacted agent would go at their medium speed. (b) A simulation where the blue agents can hide behind all obstacles and orange agents, each emitting the same IF. (c) When the blue agents are less than 1 meter away from the red agent, they are impacted by the red IF and follow the red agent.

Several hiders

Another possible extended scenario is to have several hiders. We keep the same environment but the seeker has several agents to catch. All hiders are impacted by the same IFs and hide behind obstacles and agents alike. An example is displayed in Figure 6.2(b). In addition, when one hider is less then 1 meter close to the user-controlled agent, it switches group and becomes a "follower". The user-controlled agent emits another simple velocity IF, displayed Figure 6.2(a) that makes the followers follow it (see Figure 6.2(c)). This example shows a more realistic version of the hide and seek game with several hiders and the found hider following the seeker after being caught. However, please note here that the change between the "hider" (meaning being impacted by the hiding IF) and the "follower" state (being impacted by the following IF) is scripted in this version. Video 2 in Table 6.1 shows the resulting simulation in 2D.

Scary giant

A final scenario using a hiding IF is the *Giant* scenario. Here imagine that the user-controlled agent is a giant that scares other agents. In this scenario, there is one user-controlled agent, eight obstacles and eight afraid agents. When the user-controlled agent moves, obstacles emit fields which make other agents hide behind. When the user-controlled agent does not move, obstacles stop emitting their IFs (Figure 6.1(a)). On the other hand, when the user-controlled agent does not move, it emits the IF Figure 6.3(a) and when it moves, it does not emit any fields. This two fields combination leads to a scenario where, if the user-controlled agent keeps still, agents are slowly coming toward it (see Figure 6.3(b)) and when it moves (speed superior to 0.2 m/s) they flee and hide behind the obstacles (see Figure 6.3(c)). Compared to the last scenario, several differences can be pointed out. First, the hiding IF is now also parametric



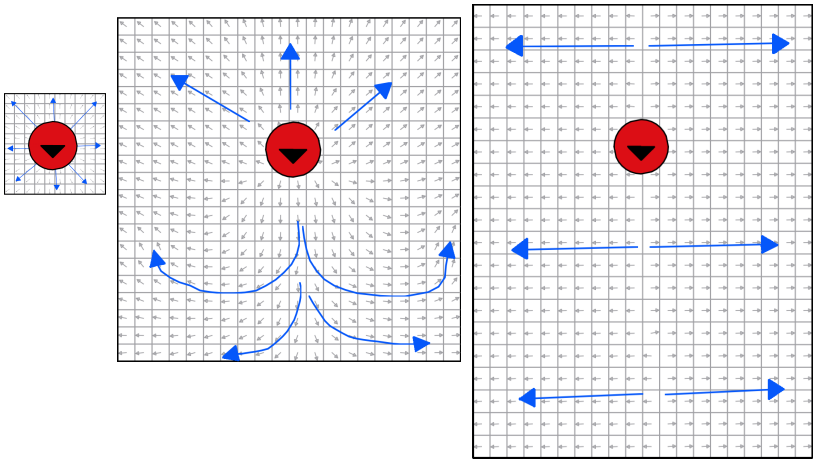
(a) Velocity IF (30×30 **m)** **(b) Simulation with still red agent** **(c) Simulation with moving red agent**

Figure 6.3 – Results for the *Giant* scenario (Section 1.1). (a) A velocity IF around the source (in red), the color of the vector show the relative speed the impacted agent should take (from fast in red to slow in blue). (b) A simulation where the blue agents are going slowly toward the red agent, impacted by the red IF (sketch in (a)) (c) When the red agent moves, it does not emit a IF any more and the agents are assigned to hide behind one obstacle each.

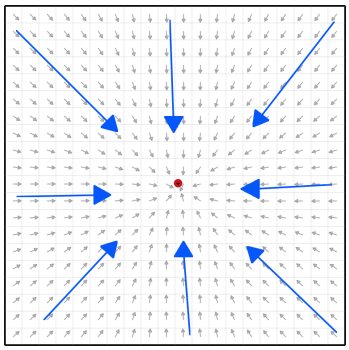
relatively to the speed of the user-controlled agent (as is IF Figure 6.3(a)), when the speed is inferior to 0.2 m/s, the obstacles actually emit a "empty" IF (with only null vectors or without guide curves). This means that this scenario does not require any scripted condition, compared to the several hiders scenario. Second, this time, each obstacle's IF only impacts one of the agent, which means that each agent always hides behind the same obstacle. For a more clear comprehension, we invite the readers to watch the video 3 in Table 6.1. This scenario was adapted in VR and will be presented Chapter 7.

All those scenarios, use the same IF as a main component to make agents hide, which shows than one IF can be used for different scenarios. Many other scenarios could be though off, such as letting all obstacles impact all agents in the *Giant* scenario or to make the hider run after the seeker once caught in the *Hide and Seek*, to mimic more a tag game.

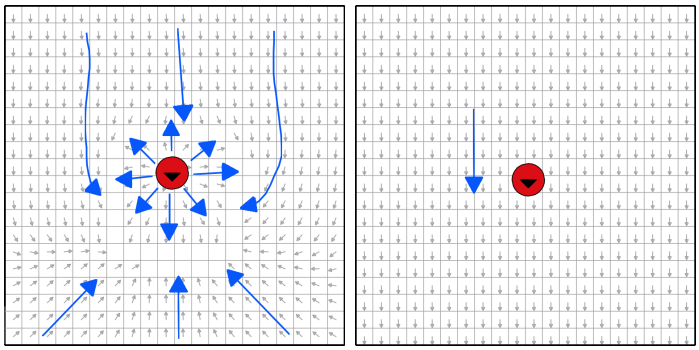
1.2 Scenario 2: VIP in a Crowd



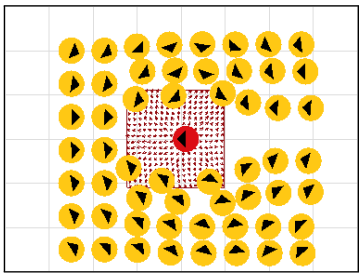
(a) Velocity IF perceived by the crowd, for $v = 0$ m/s (1×1 m), 1 m/s (3×3 m), and 1.8 m/s (3×4 m)



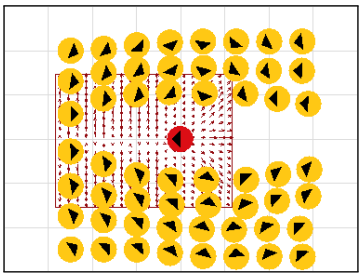
(b) Orientation IF perceived by the crowd (20×20 m)



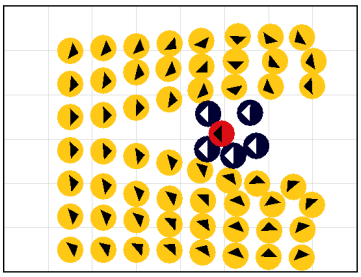
(c) Velocity IF perceived by the bodyguards, for $v = 0$ m/s and $v = 1$ m/s (5×5 m)



(d) Simulation (VIP speed: 0.6 m/s)



(e) Simulation (VIP speed: 1.8 m/s)



(f) Simulation with bodyguards

Figure 6.4 – Results for the *VIP in a Crowd* scenario (Section 1.2). (a) Keyframes of the velocity IF used by the crowd. (b) The orientation IF used by the crowd. (c) Keyframes of the velocity IF used by the bodyguards. (d–e) Simulation examples with different speeds for the VIP (in red). The interpolated IF is shown as well. (f) Simulation example with bodyguards (in dark blue). Here, all IFs are omitted for clarity.

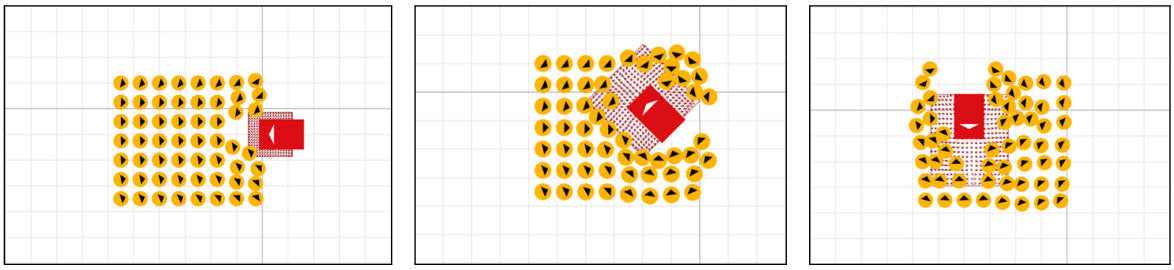
For the scenarios of this section, the vectors of the IF sketch will be always colored in grey for more clarity. The reader should however keep in mind that in the editor, those vector were colored in red, meaning that the impacted agents should always go at full speed. Next, we show an example where a crowd makes room for a user-controlled 'VIP' agent. To model this, we make the VIP agent emit two IFs: a parametric velocity IF that depends on the source speed (Figure 6.4(a)) and an orientation IF that makes agents look at the source (Figure 6.4(b)). For the velocity IF, the domain grows and the pushing effect becomes stronger as the speed increases.

The simulation features a small crowd of agents with the *IFs+GoalReaching* profile. The goal of each agent is set to its starting position, so that the agents move back to their old position after the VIP has passed. Figures 6.4(d) and 6.4(e) show how the crowd responds differently depending on the speed of the VIP agent.

Finally, we extend the scenario to include five 'bodyguard' agents with the *IFs-Only* profile. We make the VIP agent emit another velocity IF to which only the bodyguards respond. This IF (shown in Figure 6.4(c)) is parametric again: it lets the bodyguards align with the VIP when it is moving, and (re-)group around the VIP when it stands still. The latter keyframe IF uses zero areas to let the bodyguards stop in a circle around the VIP. Furthermore, the bodyguards themselves also emit the same pushing IF as the VIP. Figure 6.4(f) and video 5 Table 6.1 show an example of the simulation with bodyguards.

Very easily, the VIP can be replaced by an emergency vehicle that fends the crowd using the exact same IF. Figure 6.5 shows screenshots of such scenarios and video 3 Table 6.1 shows the resulting simulation.

All examples related to the VIP scenario are displayed video 4 of Table 6.1.

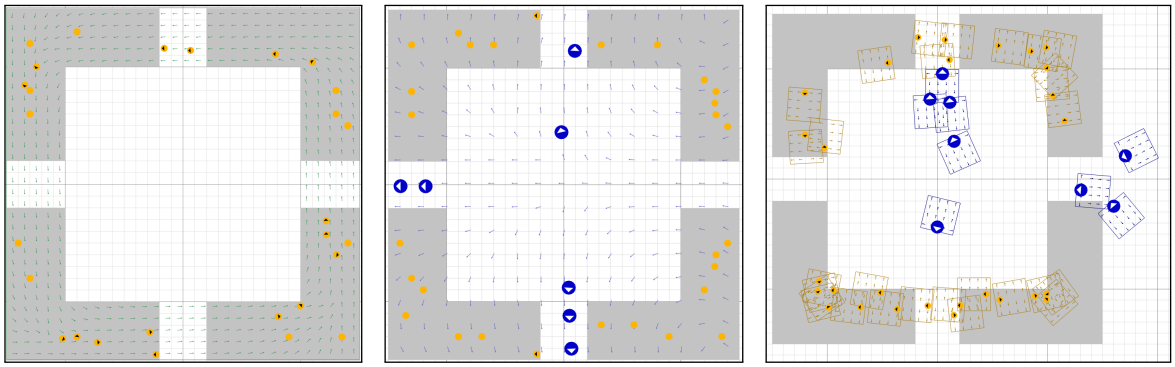


(a) Simulation (vehicle speed: 0.4 **m/s)** **(b) Simulation (vehicle speed:** 1.6 **m/s)** **(c) Simulation (vehicle speed:** 1.8 **m/s)**

Figure 6.5 – Results for the *Ambulance in a Crowd* scenario (Section 1.2).(a–c) Simulation examples with different speeds for the emergency vehicle (in red). The interpolated IF is shown as well.

1.3 Scenario 3: Crossroad

The crossroad scenario is a classical scenario of the literature, here we will show a version obtained with IF. Note that we did not use any scripting for this scenario. This scenario simulates a cross road,



(a) Pedestrians flow **(b) Cars traffic field** **(c) Cars stop**

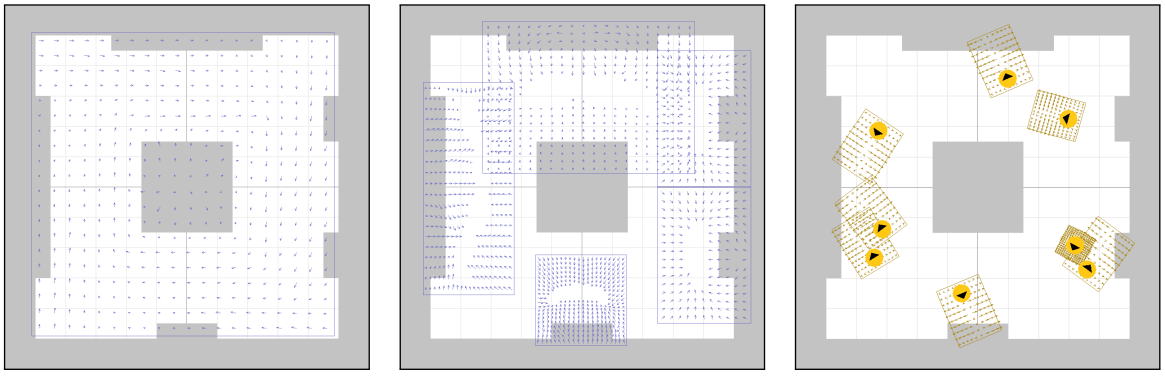
Figure 6.6 – Results for the *Cross-Road* scenario (Section 1.3). (a) Velocity Fields to make one group of agents navigate counter clockwise (green) on the crossroad. (b) Traffic velocity and orientation IF for the cars. (c) Example of a stop when a pedestrian cross, in yellow the velocity IF emitted by pedestrians and in blue the one emitted by cars.

where 33 pedestrians walk on a sort of roundabout (Figure 6.6(a)). The pedestrians only answer to this environment IF, they avoid each other using ORCA (*IFs+ORCA*). To do this, a simple velocity IF is emitted by the environment to make the pedestrians navigate. More precisely, there are actually two mirrored fields. One that makes half of agents circle clockwise and another that makes the other half circle counter clock wise, the latter is illustrated in Figure 6.6(a). The environment is also source of the velocity and orientation IF (identical) responsible for the traffic of cars, as shown in Figure 6.6(b). Having an orientation IF identical to the main navigation IF ensures that the cars do not rotate abruptly if they reverse. Cars only answer to IF (*IFs-Only*), and their motions are ruled by three different velocity IFs. The velocity IF emitted by the environment to model traffic, velocity IFs emitted by the pedestrian and IFs emitted by other cars. Indeed, pedestrians are sources of a velocity IF that makes cars stop when they are 1.5 meters away from them (in yellow Figure 6.6(c)) when crossing. Similarly, each car is source is of an IF to avoid them to crash into each-other, illustrated in blue Figure 6.6(c) as well. Video 7 of Table 6.1 displays the results of the simulation.

1.4 Scenario 4: Museum

This example is a museum scenario where 8 *IFs+RVO* agents move through a corridor and look at paintings. The central pillar emits two velocity IFs for walking around it in a clockwise or counter-clockwise way; each agent uses one of these two IFs, as in the previous scenario presented Section 1.3. Figure 6.7(a) shows the clockwise IF. Please note that the two IFs were colored in the IF editor to propel impacted agents at a slow pace. Each agent has a different attributed maximum speed, making them all have a different speed while visiting the museum. Next, each painting emits a velocity IF with a zero area that lets agents stand still at a certain distance from that painting; these IFs are shown in Figure 6.7(b).

Each painting also emits an orientation IF that lets agents face the painting; we have omitted these IFs from our figures for clarity reasons.



(a) Velocity IF (corridor) **(b) Velocity IFs (paintings)** **(c) Simulation**

Figure 6.7 – Results for the *Museum* scenario (Section 1.4). (a) One of the velocity IF for walking around the central pillar. (b) The velocity IFs for all five paintings. (c) Screenshot of the simulation, also showing the parametric IFs around standing and moving agents.

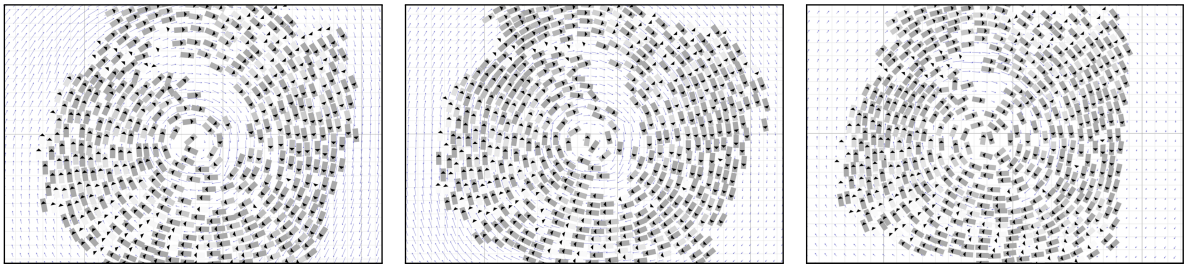
Also, each agent A_i emits a parametric velocity IF that prevents others from entering A_i's line of sight when it is standing still. This way, others will avoid A_i politely when it is looking at a painting. Figure 6.7(c) shows a screenshot of the simulation and the agents' IFs. Again, more results can be found in our supplementary video 8 of Table 6.1.

To let agents switch between walking around and studying a painting, we have added the ability to (de)activate IFs using timers. Whenever an agent enters the domain of a painting velocity IF for the first time, the agent will ignore the corridor IF for a number of seconds. When this timer has passed, the agent ignores the painting IF and uses the corridor IF again, so it continues exploring the museum. However, the *orientation* IFs stay active all the time, so that agents always face paintings that are in range. The timer system is not part of the IF technique itself, and it required some additional modeling/programming effort specifically for this scenario. Note that this and the *Hiders and seeker* scenarios are the only examples with such an extra system.

1.5 Scenario 5: Mooses

This scenario is a little different from the others and displays behaviors that are not human-like. In this scenario, we try to imitate how 500 mooses behave in a herd in winter. They tend to gather together around the weaker animals to protect them and keep warm while moving in a circular motion. To this end, 500 *IFs-Only* agents are impacted by a parametric IF emitted by the environment. The IF simply describes the circular motion of the moose, but to get an organic motion, the circle changes its shape by a parameter. In this scenario, the parameter is not the speed, but a manual parameter that can be changed over time by pressing keyboard keys. Video 9 shows of Table 6.1 the comparison between the simulation

and the real footage of mooses hords.



(a) Simulation **(b) Velocity IFs (paintings)** **(c) Simulation**

Figure 6.8 – Results for the *Moose* scenario (Section 1.5). (a–c) The simulation in 2D of the *Moose* with the previously sketched velocity parametric IF.

In this section we have shown IFs in use, using parametric IFs to realize a complex scenario that would not have been so easy to realize with other techniques. Now we can ask whether these scenarios are indeed as easily reproducible as we claim. The next section describes the user study we conducted in which several participants took part to test the usability and intuitiveness of the method. The scenarios implemented range from a simple velocity IF to the hide and seek scenario presented in Section 1.1.

Video	Title	Description	Link
1	Hide and Seek Section 1.1	This scenario demonstrates parametric IFs that use relation between objects. This video shows the Hide and Seek scenario from sketching to 2D and 3D simulation.	https://youtu.be/cfxA8VLozX4
2	Several Hiders Section 1.1	We add more hiders to the Hide an Seek scenario, and when hiders are close enough of the seeker, meaning they are found, they follow it.	https://youtu.be/9HOMNvyczm0
3	Scary Giant Section 1.1	An alternative version of the Hide and Seek scenario. This video shows all fields of the scenario. First with only one hider and one obstacle and then with 8 obstacles and hiders. For more visibility we show fields separately (first only one obstacle fields then the seekers then all of them).	https://youtu.be/Zzkf7hfeeXg
4	VIP Section 1.2	Parametric IF with the speed of the red agent (VIP) as parameter. The video shows the sketching of the field and the simulation in 2D and 3D.	https://youtu.be/UceBPVHKeRo
5	VIP and Body Gards Section 1.2	With the same parametric IF as the previous video plus bodygards and dedicated field for them we obtain this scenario (2D and 3D simulation). This video shows the fields emitted by the VIP that only impact the body gards.	https://youtu.be/EnDGghqDCnU
6	Emergency Vehicle Section 1.2	If we replace the "VIP" by a vehicle, we can obtain a emergency vehicle crossing the crowd. The video displays the field emitted by such vehicle.	https://youtu.be/xO5ROkugCec
7	Crossroad Section 1.3	The crossroad scenario, several IF are used as navigation fields but pedestrians and cars emit velocity IFs as well to avoid collision. Those IFs are displayed at the beginning of the video.	https://youtu.be/vDGAqxqe2CE
8	Museum Section 1.4	Agents are visting a museum, this video shows the museum scenario in 2D and 3D.	https://youtu.be/U4pxTQb_f2c
9	Mooses Section 1.5	Video compares the Mooses scenarios with video shootage of mooses.	https://youtu.be/22VLDFC2vjc

Table 6.1 – Video simulations of the scenario Section 1

2 User Study

To evaluate the efficacy of IFs and the IF editor for non-expert users, we conducted a user study with 22 users who were familiar with computer animation but not with IFs. Our goal was to evaluate how easily they could learn to independently sketch IFs to design specific agent interactions. Please note that the scenarios in this study are different from those in Section 1: thus, the user study shows even more examples of IF use cases.

It is important to point out that our general goal was not to evaluate the actual design of the GUI, but rather the ease and ability of non-expert users to learn and independently leverage the functionalities of IFs to design scenarios involving interactions of increasingly complexity.

2.1 Pilot Study

Prior to the actual study, we conducted an online pilot with 6 participants who were familiar with computer animation but not with interaction fields. Our goal was to collect initial feedback through an

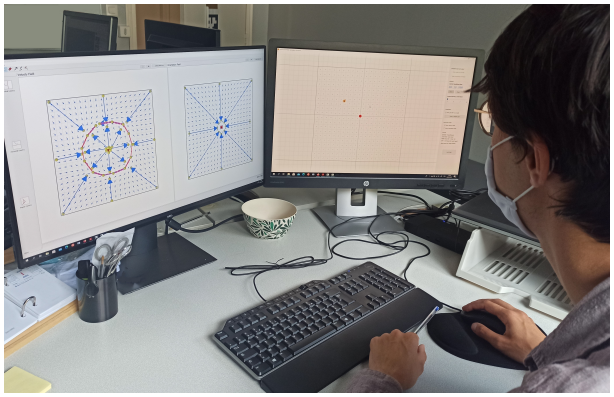


Figure 6.9 – Photo of a user conducting our user study.

online form about the difficulty and duration of the initial training and tasks we intended to include in the experiment. The purpose was not to evaluate the actual design of the GUI, but rather improve upon or include specific GUI functionalities based on user feedback. Following the pilot study, we identified a need for reorganizing the experiment to both shorten the duration and facilitate the learning process, in particular in terms of the number and order of the tasks to perform.

2.2 Protocol

For the subsequent user study, participants were invited to take part in the experiment at our research institute. Upon arrival, they were first asked to read and sign a consent form. The experiment itself used two 24-inch screens and a mouse and keyboard, as shown in Figure 6.9. The setting as well as one participant realizing task 6 are displayed video 6 of Table 6.4. On the left screen, participants saw the IF editor GUI in which they could draw interaction fields. On the right screen, they saw the resulting simulation, in which the drawn IFs were emitted by a source (in red) and received by another agent (in yellow). Participants were always allowed to refine their IF sketches on the left and play a new simulation on the right, until they were satisfied with their result and marked a task as completed.

Instructions were displayed on the computer screens as the experiment progressed. The experiment consisted of an initial exploration phase, followed by five scenarios of increasing complexity. The total duration of the experiment depended strongly on the participant, but did not exceed 2 hours.

Exploration. The aim of the exploration phase was to give participants a first glance of IFs. They were given a short video-guided introduction of its uses and applications, and the opportunity to freely explore our IF tool for ten minutes. They were told to play with the GUI, to test simple drawings, and to visualize results in the simulation window. This stage helped us gauge the initial learnability of the GUI and the users' first impression. Participants were also allowed to interact with the experimenter and to ask questions.

Scenarios. After the exploration phase, participants were asked to draw IFs for specific agent behaviors in a sequence of scenarios. Each scenario started with a training example, followed by one to three evaluation tasks. The seven evaluation tasks are summarized in Table 6.3.

— **Training tasks.** Each scenario started with a training example covering a specific concept of IFs (e.g., controlling velocity, controlling orientation, creating parametric IFs.). For each concept, participants were provided precise instructions to use the new functionality effectively. They were first instructed to attempt to draw the matching field by themselves, and then to follow a video tutorial that showed an approved way of designing the expected field. At the end of each training, participants were asked to answer two questions on a 7-point Likert scale: "I understood the concept of this training" and "I am satisfied with the ease of completing the training". All tutorial videos are accessible in Table 6.4 (video 1 to 5).

— **Evaluation tasks.** After each training, participants performed one or more evaluation tasks, to evaluate their ability to apply the functionalities learned in the previous training. They were only provided with written instructions describing the task to achieve (see Table 6.3) and a video showing the expected agent behavior (but not the expect IFs). While performing the evaluation tasks, participants were not allowed to ask questions to the experimenter. They were instructed to stop when they felt that their simulation was similar enough to the expected result presented in the video. At the end of each task, participants were asked to answer the following question on a 7-point Likert scale: "I am satisfied with the ease of completing this task" and "I am satisfied with the end result".

Final questionnaire. At the end of the experiment, participants were asked to answer a general questionnaire about the usability of the IF editor. This questionnaire is based on the System Usability Scale Questionnaire (SUS) [Brooke, 1996], which is commonly used to evaluate the perceived usability of commercial tools. However, since our current tool is still in development and was not designed for a commercial use at this stage, we decided to create our own questionnaire, see Table 6.2. We used 5 questions from the SUS questionnaire (see questions 2-6 in Table 6.2) and included two questions about error management (questions 7 and 8 in Table 6.2) and significance of code (questions 9 and 10 in Table 6.2) from [Assila, Ezzedine, et al., 2016]. An additional question was added at the beginning of the questionnaire referring to the training for the system (question 1).

2.3 Results

Participants. Twenty-two participants (7 women, 15 men; age: 28.4±8.0, min: 22, max: 62), volunteered for the study. They were all naive to the IF sketching tool, but had some knowledge of 3D animation or crowd simulation. They were recruited via internal mailing lists amongst students and staff. Participants gave their written and informed consent prior to the experiment. The study conformed to the declaration of Helsinki, and was reviewed and accepted by our local Ethical Committee (COERLE).

To evaluate the ease and ability of non-expert users to learn and independently use IFs to design specific scenarios of interactions, we analysed both self-reported user experience with the sketch tool and expert evaluation of participants' simulation results.

Self-reported user experience. To evaluate participants' experience with each scenario, they answered two questions on a 7-point Likert scale after each training and evaluation task. These questions were chosen to assess their understanding of the concepts presented, the ease with which they performed the tasks, and their satisfaction with the agent behavior they designed.

The answers (summarized in Figures 6.10(a) and 6.10(b)) show that participants found IFs easy to learn and to use, and that they were very satisfied with the results they could produce. In particular, participants understood each concept of the training, and they managed to apply these concepts easily in both the training and evaluation tasks. However, we noticed a tendency for slightly lower ratings for Task 3, which was the first task to introduce the concepts of the relativity and orientation of a moving source, while simultaneously requiring to define a large zero area. We believe that these two concepts might have been relatively hard to process at the same time. Nevertheless, the results show that participants could design the requested behaviors to their own satisfaction.

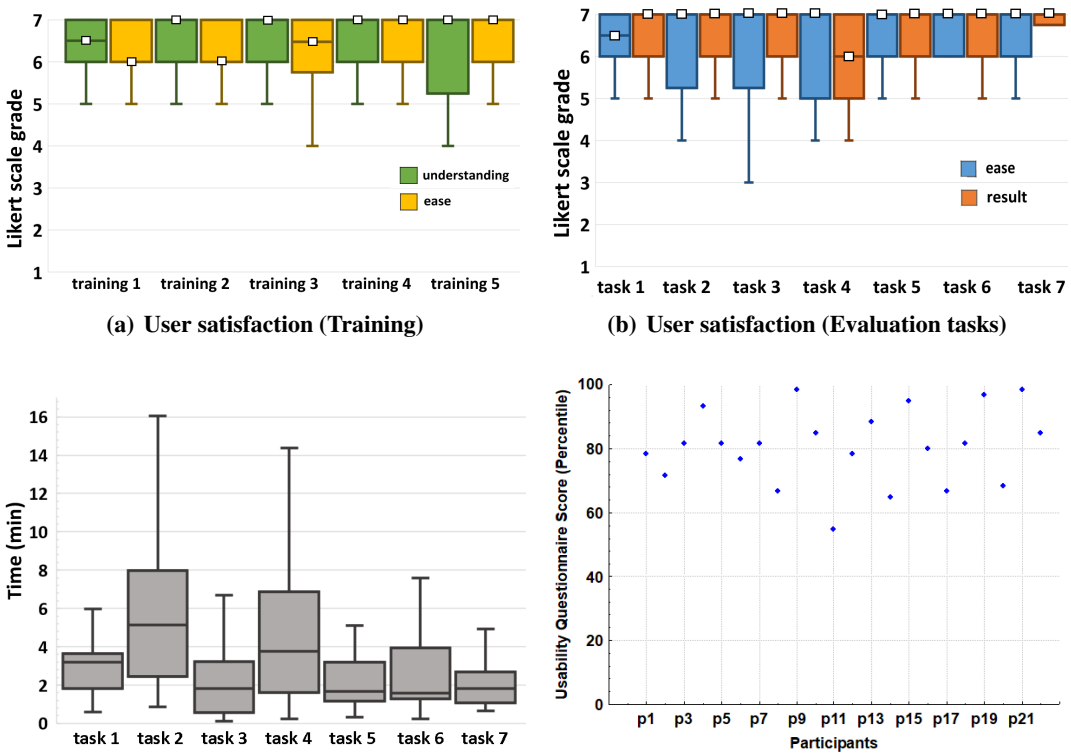
Response time. To evaluate how quickly participants were able to design agent behaviors, we recorded the amount of time spent by participants drawing IFs for each task. The timer of each task started with the first input to the sketch and ended with the last without interruption. Participants took on average 3.09 minutes to realize a task. Figure 6.10(c) shows that the majority of participants could quickly sketch their field to their satisfaction. However, some participants took a longer time to adjust, as indicated by a high standard deviation for some tasks. The inter-individual variability of the time stays acceptable with a maximum of 16 minutes for one participant in task 2.

Visual interpretation of results. Of course, not all participants drew the exact same interaction fields. Figure 6.11 gives a visual impression of the 'average' IF that participants drew for each individual task, as well as the variation among participants. It also shows the resulting agent trajectories for different participants in different colors. These results are also shown in motion in the dedicated supplementary video 7 of Table 6.4. While the IFs and trajectories vary among participants, the overall behavior remains visually comparable to what they were instructed to design.

Figures 6.11 (a) and (b) show a relatively high variance in the grid cells around a 5m distance from the source (where agents were expected to stop in these two tasks). This suggests that participants could not sketch the stopping distance constraints very precisely. This is explicable because of our grid-based IF representation: zero areas are approximated by empty cells, and not all participants selected the exact same cells to be empty.

Some visual differences in the drawn fields can also be explained by the fact that participants interpreted some task instructions (e.g. words such as "towards" or "in front") in different ways. Such

differences can be observed in Figures 6.11 (c) and (d), where some participants erased all the cells below the source's position while others tried to mimic a field of view and only erased part of them. Those differences were however not visible during the simulation because the trajectory of the source and the initial positions of the responding agents were fixed.



(a) User satisfaction (Training)

(b) User satisfaction (Evaluation tasks)

(c) Task completion time

(d) Usability scores

Figure 6.10 – (a) Boxplots showing the median ratings, interquartile ranges, and maximum/minimum ratings (outliers excluded) of the understanding of the training (green boxplots) and the ease of completing the training (yellow boxplots) for the 5 training tasks. (b) Average ratings and ranges for the 7 tasks participants completed independently. The blue boxplots represent participants' ease of completing the task and orange their satisfaction with the final result. (c) Boxplots representing the time to complete each task. (d) The final usability scores (in percentiles) for each participant.

Expert evaluation of results. To further evaluate participants' simulation results, 3 experts amongst the authors rated the agent behavior that resulted from the IFs of each participant. Prior to the experiment, we defined a list of evaluation criteria per task, describing objectively the behavior that should be captured by the IFs of that task. These criteria are given in Table 6.3. Participants did not see these lists; they were expected to infer the requirements from their overall task instructions.

After the user study, each expert checked independently (i.e. without communicating with the other experts) which of these evaluation criteria were met for each individual result (i.e. for each participant's

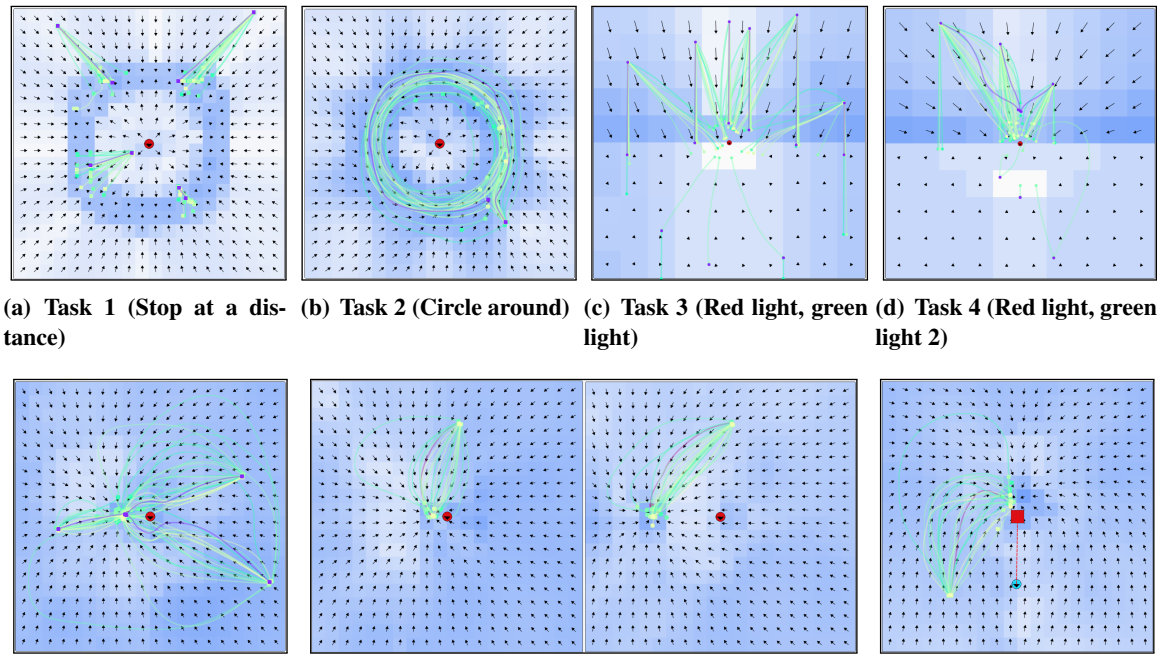
output in each task). Per task, we converted this expert evaluation to a rating on a scale of 0 to 10, based on the average number of satisfied criteria over all participants and experts. The resulting ratings per task are shown in Table 6.3 as well. As this Table shows, the overall scores were very high, indicating that almost all users were capable of drawing IFs that met all behavioral criteria. To quantify the reliability of these scores, we performed an inter-rater reliability test per task using Fleiss' Kappa. The average overall score was 0.76, which is interpreted as good reliability. The scores per task are shown in the last column of Table 6.3. The video available show the results of the participants' fields propelling the green agents. The video also illustrates how expert graded some of the tasks.

We acknowledge that the 'correctness of agent behavior' is scenario-specific and difficult to quantify, and that a review with external experts would be considered as even more reliable. Still, we are confident that this expert evaluation is meaningful: in combination with the purely visual results discussed before, it serves the purpose of assessing whether users understood the instructions.

IF usability questionnaire. To assess participants' general satisfaction with our IF editor, we presented them with a usability questionnaire of which the questions and results are given in Table 6.2. To compute an overall usability score, we first inverted the negative items scores (8 – value), and then computed an average value over all the questions. The final usability score was then normalized to a 0-100 scale to improve readability. The overall usability score averaged over participants was 58.36 ± 7.05 (using summation over the 1-7 scales), reaching a 80.6 percentile of the total score on the 0-100 normalized scale. When translated to the Sauro-Lewis Grading scale, the scores in this percentile receive a very high usability rating of A- (see [Lewis and Sauro, 2018], Table 1). As we are comparing a non-standard score of our custom questionnaire to a standardised measure of SUS, it is important to remember that the score is of a purely informative nature. Nevertheless, taken into consideration that our tool was not designed for a commercial use at this stage, and that it uses a simple interface, this grade indicates a very high usability performance.

2.4 Discussion

Our user study indicates that the IF editor is a powerful tool for non-expert users to design agent interactions. Recall that the design process itself is interactive as well: in our study, users could edit their IFs on the fly and immediately see the effect in the simulation. Overall, the IF editor allows users to design new types of behavior that typically take much more time and effort to model using traditional techniques. To improve the evaluation of IF, other metrics to objectively quantify the participants' results should be used; for one, other experts (than authors) could be asked to grade the results.



(a) Task 1 (Stop at a distance) **(b) Task 2 (Circle around)** **(c) Task 3 (Red light, green light)** **(d) Task 4 (Red light, green light 2)**

(e) Task 5 (Stay on the right) **(f) Task 6 (Parametric, left = low speed, right = high speed)** **(g) Task 7 (Hide behind obstacle)**

Figure 6.11 – Summary of the velocity IFs that the participants drew for all 7 tasks. The source of the fields is always the red object. The black arrow in each grid cell denotes the average IF vector for that cell among the IFs of all participants. The blue intensity of a cell indicates the variety among participants: it is the standard deviation of the Euclidean distance to the average IF vector. The green curves are the trajectories induced by the participants' fields for various starting positions. The purple curves are the trajectories induced by a 'ground truth' IF drawn by the authors before the user study. This trajectory corresponds to the video instructions given to the participants.

Questions	Mean	SD
1. It was easy to learn to use the system.	6.36	0.66
2. I found the system unnecessarily complex. †	1.77	0.97
3. Overall, I thought the system was easy to use.	6.23	0.69
4. I found the various functions in the system were well integrated.	6.05	0.84
5. I found this system very awkward to use. †	1.68	0.89
6. I needed to learn a lot of things before I could get going with the system. †	2.36	1.18
7. It was easy to make the software do exactly what I wanted.	5.82	1.18
8. Whenever I made a mistake using the system, it was difficult to correct it. †	3.50	1.60
9. The terminology was related to the task I was doing.	5.91	1.11
10. I was wondering sometimes if I was using the right function †	2.68	1.62

Table 6.2 – The average ratings and standard deviations for each item of the usability questionnaire in our study. † indicates negative questions, whose score were inverted for computing the final overall score. All questions were answered on a 7-point Likert scale (from 1: Completely Disagree to 7: Completely Agree).

Scenario	Task	Instructions	Expert rating criteria	Expert ratings	Inter-rater reliability Fleiss Kappa
S1	T1	The goal is to draw a velocity IF that makes the yellow agent move and stop 5 meters away from the red agent.	The yellow agent should: — be repulsed when too close to the red agent — be attracted when too far from the red agent — stop at a 5m distance — keep respecting this distance	9.13 ± 0.52	0.34 : Fair
S2	T2	The goal is to draw a velocity IF that makes the yellow agent circle counter-clockwise around the red agent at a distance of 5 meters, combined with an orientation IF that makes the yellow agent look at the red agent.	The yellow agent should: — repulsed when close to the red agent — attracted when far from the red agent — turn counter-clockwise — respect a 5m distance — look at the red agent	9.67 ± 0.52	0.87 : Very good
S3	T3	In this task, the agents play 'red light, green light'. The goal is to draw a velocity IF that makes the yellow agents move toward the red agent when the red agent is looking away, and stand still when the red agent looks at them.	The yellow agent should: — stop when the red agent is looking — move towards the red agent when it is not looking	9.77 ± 0.00	0.96 : Very good
	T4	The goal is to draw a velocity IF and an orientation IF that make the yellow agent — move and look towards the red agent when the red agent is looking away, — stand still while turning its back to the red agent when the red agent is looking towards them.	The yellow agent should: — stop when the red agent is looking — move towards the red agent when it is not looking — look away from the red agent when it is looking — look towards the red agent when it is not looking	9.58 ± 0.35	0.81 : Very good
	T5	The goal is to draw a velocity IF that makes the yellow agent stand and move on the right side of the red agent, 2 meters away from it.	The yellow agent should: — go to the right side of the red agent — respect a 2m distance to the red agent — maintain its relative position	9.44 ± 0.13	0.72 : Good
S4	T6	The goal is to draw a parametric velocity IF that makes the yellow agent stand and move on the right side of the red agent. Unlike in task 5, the yellow agent must stay close when the red agent is moving slowly (1 meter way), and remain further away when the red agent is moving faster (5 meters away).	The yellow agent should: — go to the right side of the red agent — move further when speed is high — move closer when speed is low — respect 5m distance at high speed — respect 1m distance at low speed	9.51 ± 0.79	0.62 : Good
S5	T7	The goal is to draw a parametric velocity IF that makes the yellow agent move behind the obstacle to hide from the blue agent, combined with an orientation IF that makes the yellow agent look at the blue agent. The obstacle is the source of the velocity IF. The blue agent is the source of the orientation IF.	The yellow agent should: — always go to the opposite side of the obstacle — look towards the blue agent	10.00 ± 0.00	1.00 : Very good

Table 6.3 – Descriptions of the five scenarios and seven evaluation tasks in the user study. Columns show the overall goal of a task (which the participants received as instructions), the agent behavior criteria used during the expert evaluation, and the resulting expert grade (averaged over all users and experts combined).

Video	Title	Description	Link
1	Tutorial video of training 1	Participants were asked to achieve the same scenario following the video. The goal was to sketch a simple velocity field.	https://youtu.be/gjJxByCIhSc
2	Tutorial video of training 2	Participants were asked to achieve the same scenario following the video. They were taught to use an orientation field on top of the previous velocity field.	https://youtu.be/sCi1H1DFWho
3	Tutorial video of training 3	Participants were asked to achieve the same scenario following the video. The source was now mobile, they had to draw a velocity and orientation IF keeping in mind it will translate and rotate.	https://youtu.be/xbFD-SCs3y8
4	Tutorial video of training 4	Participants were asked to achieve the same scenario following the video. They were taught to sketch a Parametric IF with keyframes, with a tutorial scenario similar to the VIP's.	https://youtu.be/o-IJV-qWJpo
5	Tutorial video of training 5	Participant were asked to achieve the same scenario following the video. They were taught to sketch a parametric IF with relation between two objects. This scenario resembles the hide and seek scenario.	https://youtu.be/gEMteaXhjr4
6	Task 6 filmed	A participant realizing task 6.	https://youtu.be/qlLlIcskAHY
7	Resulting scenarios	Results of all participants (green) and expected agent behavior (purple).	https://youtu.be/hiYEIdtLgT4

Table 6.4 – Videos related to the User Study and its evaluation Section 2.

3 Conclusion

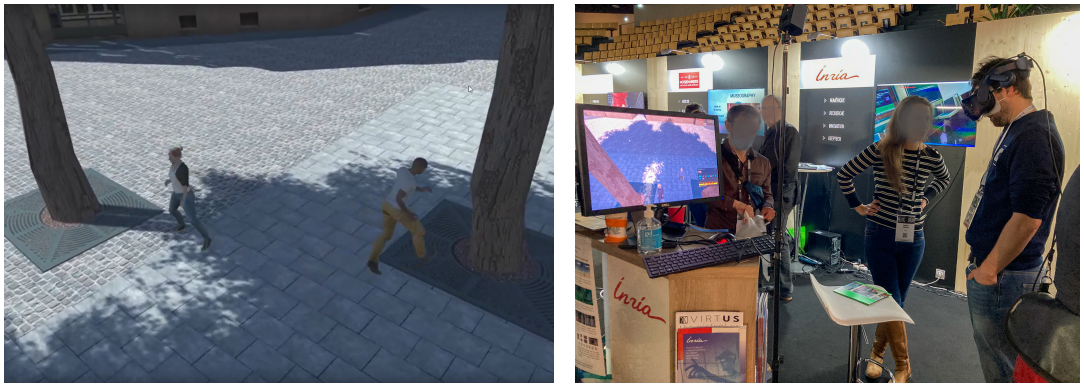
This chapter showed the interest and application of IFs. While our applications show that IF is a method that is intuitive and easy to apply, we can consider making it even more intuitive and straightforward. Indeed, in the user study, participants had to use two windows with two different softwares to only display 2D simulation results. It would be of interest to reduce the number of interfaces necessary to design scenarios directly in 3D faster. For this, VR is of interest since it allows to be able to assert the results quickly and, as it was highlighted in the survey of Bhattarcharjee et al. [2020], VR has underrated qualities to sketch for 3D. More importantly, it can provide novel ways of sketching in the 3D simulation, which we explore in the next chapter.

CHAPTER 7

SKETCHING INTERACTION FIELDS IN VIRTUAL REALITY: A PROOF OF CONCEPT

Virtual Reality is a tool increasingly used in crowd simulation. Section 4 has shown that VR is often used to validate the trajectories generated by simulation models [Ahn et al., 2012; Kim et al., 2016; Rojas et al., 2014b] or to study perceptual motor responses to different crowd environments [Kyriakou et al., 2017; Llobera et al., 2010; Sohre et al., 2017]. According to Ahn et al. [2012], every crowd simulation method should also undergo an immersive evaluation from the point of view of someone embedded into it. Additionally, VR is considered to have a high potential in providing novel intuitive ways of sketching applications [Bhattacharjee and Chaudhuri, 2020].

As a first step to analyze the interests of IF in VR, we integrated scenarios described in Section 1 in VR. For example the scenario described in Section 1.1 was translated in VR and presented during the VR exhibition Laval Virtual [Laval Virtual, 2022]. Many visitors of the exhibition could try out and be a scary giant (see video 1 Table 7.1 and Figure 7.1) that scares away the virtual agents. This scenario received a lot of positive feedback and the testers found the virtual agents expressive and responsive to their movements.



(a) View of the immersed user. **(b) VR scenario showcased at Laval Virtual 2022.**

Figure 7.1 – The 'Scary Giant' scenario presented in Section 1.1 in VR: view of the immersed user on the left and picture of a participant testing the demo at Laval Virtual 2022.

The positive responses of participants encouraged us to further develop IF in VR and to invest in

immersive sketching to investigate the advantages of sketching IFs in VR.

For this reason, this chapter presents a proof of concept to extend IF sketching possibilities using VR. We expect the following benefits:

— It will enable us to design more interactive scenarios where the user could quickly edit and change the IF and test the results in one application.

— Scenarios applied in VR will confirm the realism of the output behaviors of IFs more easily than on a 2D screen.

— Opening up the use of IF for scenario design in VR will also bring new possibilities for creating IFs. For example, tracking the users' position in VR to extract their trajectories and create IFs could be even more intuitive than sketching.

We will describe in Section 1 how we integrated VR into our current framework, and in Section 2 we will present the VR interface, which was developed to sketch IFs. In the last sections, we will discuss our observations about the tools and future works alternatives to continue in this direction.

1 VR Implementation of IF

In this section, we first describe how the new VR interface fits within the already existing implementation of IF.

IF was integrated in VR on top of the Unity framework used for 3D simulation presented in Section 2. We used the 'XR' Unity plugin [Unity, 2022] that was developed for Unity 2021.3. It aims at supporting applications in VR, Augmented Reality and Mixed Reality, and is generic enough to support several VR displays including Head Mounted Displays (e.g., HTC Vive, Oculus, Valve Index).

The prototype will be presented to investigate the uses of sketching. We did not re-implement in VR all functions available in the 2D editor. IF in VR does not aim to replace the 2D editor, as we believe the 2D editor is still helpful for a global view of scenarios, offering more functions. In other words, a classic process for using IF to create complex immersive scenarios would be:

— Define the scene settings (e.g., agents, obstacles) and roughly sketch IFs on a 2D screen.

— Once defined, try out the scenarios in VR.

— Change and modify IFs, by sketching or acting, for small changes.

In the current pipeline, IFs are to be sketched or defined in the usual 2D editor (in green in Figure 7.2) before immersing users in the IF VR sketching application. Designers must define the scene's elements (e.g., objects, agents, corresponding IFs) before being immersed in VR. Although they can choose to create "empty" IFs that will be sketched afterward in VR, the files of each IF must be defined before launching the simulation. These 2D outputs IF files contain all the important information about the IFs.

These files are the heart of the framework, as they keep updating the information of the different IFs used by the framework's various components.

Ideally, users could have the possibility to design the entire scenario in VR (e.g., create new IFs or new agents), but this would require more development and integration that would not serve research purposes.

Our current VR implementation is based on two modes: "simulation mode" and "sketch mode".

Simulation mode

The first mode is active when the simulation loop is running and where users can experience the scenario. The functions of this mode are similar to the 3D framework described in Section 2 and in Figure 5.1, but with an immersed user navigating the scene. *CrowdMP* is responsible for sending the user's state (position, orientation and speed) to the *UMANS* library and for requesting the updated states of agents and IFs (e.g., parameter value, predictions).

Sketch mode

The second mode enables users to sketch and modify IFs in VR. As illustrated Figure 7.2, this mode is represented by the yellow block. When a user selects the sketch mode, the simulation loop stops: the IFs and agents are then maintained as no update is given by *CrowdMP* nor *UMANS*. Users can then navigate in the scene and change IFs. However, *CrowdMP* still manages the VR interface by displaying IFs and updating the IF matrices according to the user inputs. The loop restarts with the modified IF once the user decides so. When the loop restarts, the IFs files (blue block in Figure 7.2) are modified according to the *CrowdMP* registered data. Indeed, in this prototype, *UMANS* and *CrowdMP* reload all the IFs data and interpolate IFs at each frame to propel agents accordingly. This could be optimized furthermore in the next version of IF VR.

The IF files can be modified in the 2D editor or in VR according to the user's inputs to a field (red inputs in Figure 7.2). An IF that has been sketched in 2D can be edited in VR by moving the handles, sketching new guide curves or creating zero areas. To implement this, i.e., being able to modify guide curves of a pre-sketched IF, the data representation exported from IFs was modified.

Our previous simulation approximated IFs by grids. This had limitations in terms of precision (e.g., for describing behavior around detailed obstacles) and scalability to large environments. To alleviate this, the simulation can also directly use the guide curves and zero areas from the IF editor, instead of importing the entire matrix of the vector field. This comes at a computational cost, though, as the simulation would now compute the interpolation at each frame instead of selecting the right vector of the matrix. We leave the two options available according to the situation.

In this version of IF in VR, IFs are not approximated by grids, but they still have a border, which makes the new computation continuous, but only inside the field's border. Keeping the border was a choice to ease the sketching by non-experts, as it is more intuitive (clear notion of where an IF has an

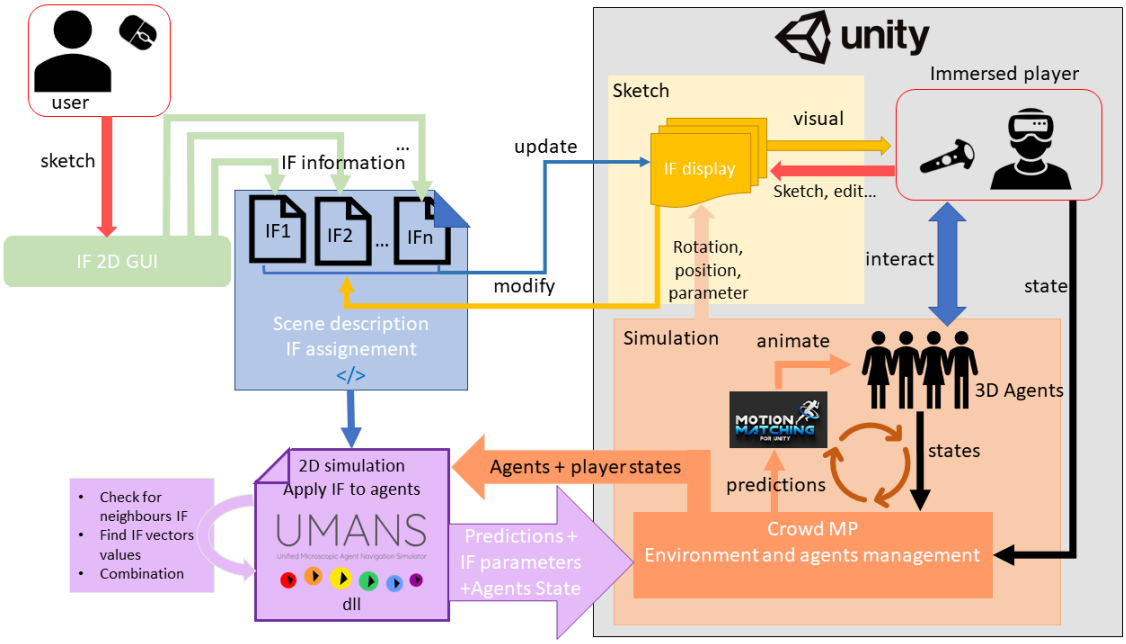


Figure 7.2 – Integration of VR in the IF framework. In "simulation mode", represented in the orange block, the user can now be immersed in VR and interact directly with agents. In "sketch mode" (yellow block), the simulation loop stops and the user can sketch or edit IFs in VR. The user alternates between the two modes.

impact) and to limit the impact of the fields to its boundaries. Depending on the scenario, one can take advantage of the strict size of the IF, for example to switch between behavior in the "cross road" scenario (Section 1.3), to make cars move or stop. In other scenarios, this could lead to mistakes, such as in the "Hide and Seek" scenarios (Section 1.1): if the hider leaves the IF, the behavior will stop unrealistically. Note that keeping the border or not is a choice that could be left to the users by simply removing the border if desired. This would come at a higher computational cost, though, as computing a velocity or orientation from one IF would no longer take constant time.

To summarize, the IF files are now composed of a set of guide curves $\mathcal{C} = \{C_j\}_{j=0}^{c-1}$. Each guide curve C_j includes a set of handles $C_j = \{h_i\}_{i=0}^{n-1}$. After any IF modification, the vector field must be interpolated again using the position of those handles. We refer the reader to Section 2 as a reminder of the interpolation process. The list of vectors used for interpolation is now $\mathbf{v}_k = (\mathbf{h}_{i+1} - \mathbf{h}_i)$ with $i \in [\![1, n-1]\!]$ and $k \in [\![0, n \cdot c - 1]\!]$. This means that the displayed matrix is interpolated at each input of the user. However the file of the IF is modified only when the sketching mode is over (yellow line in Figure 7.2).

The computation of the fields is now continuous, and the grid metaphor of the field displayed is only used for informative purposes (in the 2D editor and in VR). The velocity and orientation vectors are interpolated at each frame according to the position of the simulated agent.

We will now describe how a user can sketch IFs in VR.

2 Sketching in VR

In this section, we will now detail how immersed users can sketch IFs in VR. In our current implementation, the user navigates in the scene using two handheld (HTC Vive) controllers (illustrated in Figure 7.3) and is immersed using an HTC Vive Pro. During the sketch mode, users are given several options to edit IFs.

— Create new guide curves.

— Modify guide curves.

— Create zero area.

2.1 VR Interface

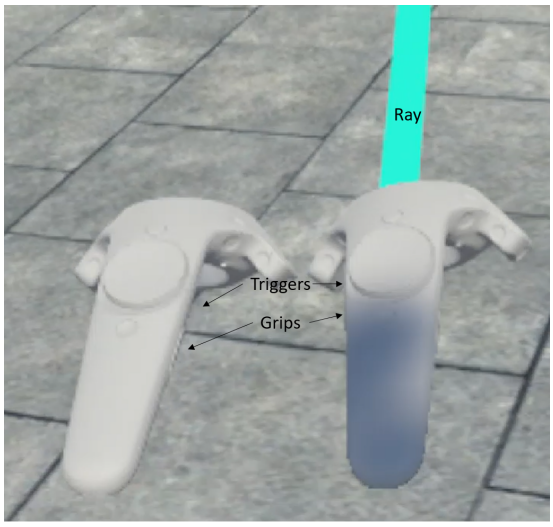


Figure 7.3 – 3D models of the controllers in VR. Those are HTC Vive models but our framework could work with many controllers (even if we recommend using Vive for consistency).

To switch to sketch mode, the user must first choose which field to edit or sketch. For this, the user can navigate between the IFs by hovering over the source with the ray interactor (see video 2 of Table 7.1 and Figure 7.4), which makes the preview of the vector field of the source's IFs appear. Once the user chooses a field to edit, pressing the grip of the controller emitting the ray (called "main" controller) displays a graphical interface, as in Figure 7.4. The user can then pause the simulation by clicking on 'sketch', hence switching between the two modes. Only one "main" controller is used to sketch and emits a ray, the other is used as a "secondary" controller which grip and trigger are used for other functions, which will be detailed later.

This interface is unique to every IF of the scene and enables to manage the options of fields. As shown in Figure 7.5, a user can choose to display the orientation fields or velocity fields emitted by the source

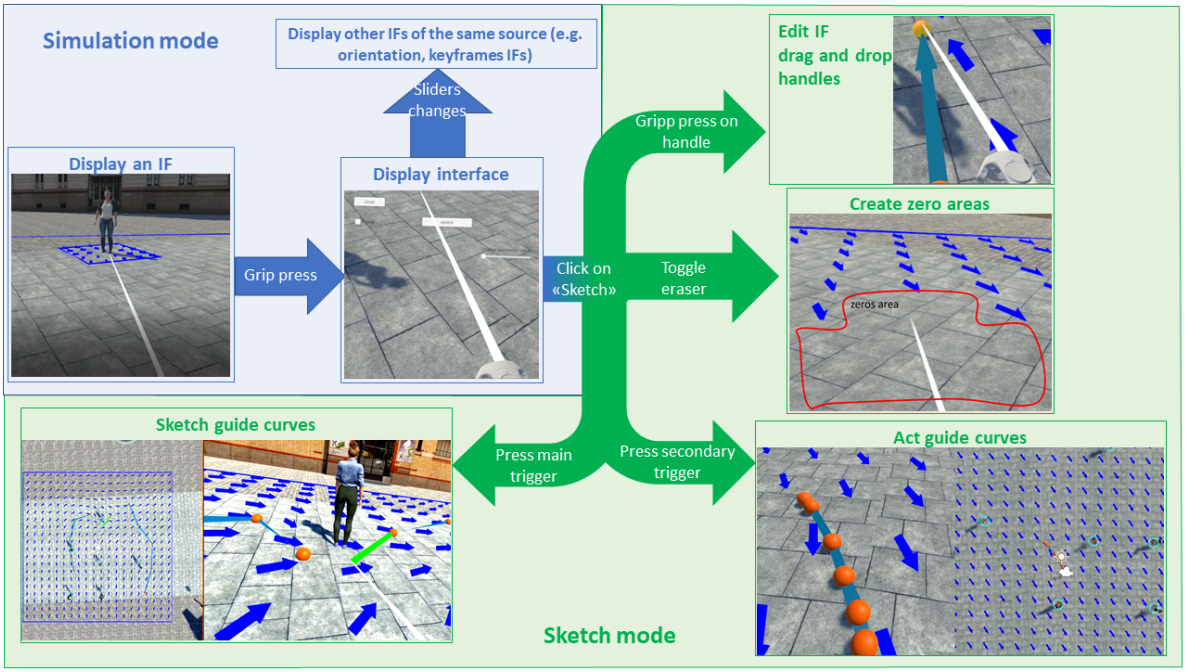
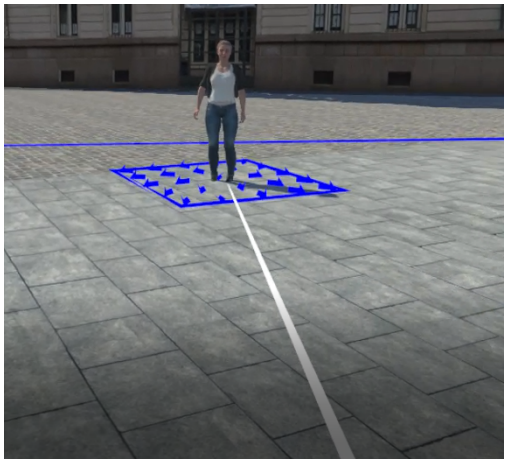
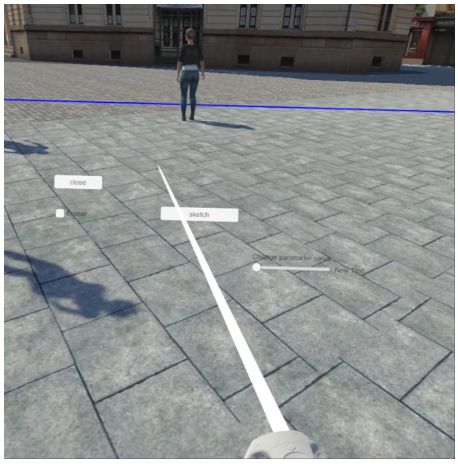


Figure 7.4 – An immersed user can hover over sources to display their IFs. Once an IF is selected, the user clicks on 'sketch mode' to start editing the IF. 'Sketch' mode enables to add new guiding curves by sketching them or acting them out, to modify guide curves by dragging and dropping handles, and to create zero areas by hovering over vectors when 'eraser' is toggled. In the top-down view, the user is represented by the white camera, the blue circles are the other agents, and the vector field of the IF being edited is in dark blue. Handles are in orange, and in yellow when selected, the preview guideline is in green, and the already sketched guide curve in light blue.



(a) Hovering over a field makes it appear.



(b) VR interface with the IF options for sketching and displaying.

Figure 7.5 – Selecting an IF to sketch on it. The vector field is in dark blue.

and to navigate through all of them. Once the sketch mode is active, the user can also choose to edit one of the keyframes (if any) of a parametric IF or can decide to go in 'erase' configuration. However, the main feature of the sketching mode is the sketching of fields.

In addition, as shown Figure 7.6 and video 6 of Table 7.1, users can press a controller's grip to display a top-down view in front of them. We will further explain the choice of this addition in Section 3, but we can resume its function as a global sketching facilitator.

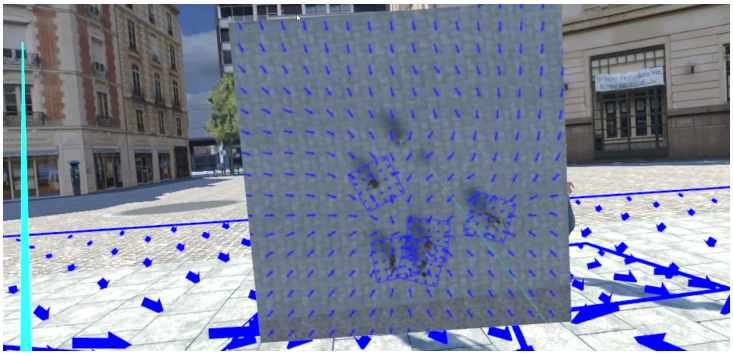


Figure 7.6 – Example of the view appearing at any time with a grip press.

2.2 Sketched Guide Curve

Following Arora et al's [2017] observation that mid-air sketching in VR is challenging, while sketching on virtual 2D planes reduces imprecision, we similarly chose to let users sketch IFs on virtual 2D planes. We do not use a physical surface to prevent limiting user's movements, and therefore directly sketch IFs on 2D planes on the virtual floor. To pursue this, IFs matrices are displayed on the ground (to not disturb the view or navigation), and users sketch standing from their own height. To ease this process, we do not use a stylus but rather a virtual ray, which is more commonly available in VR applications. This ray is emitted by the "main" controller, and is used to interact at a distance with fields and sources, as illustrated in Figure 7.3. Users can switch controllers of hand according to their hand preference to have more facility in drawing. Then the ray is displayed and stops when colliding with an IF, and the tip of the ray is responsible for sketching when the "main" controller trigger is pressed.

While the user is adding a guide curve to the IF, at every time interval Δt a new handle h_i is created where the ray tip collides with the field. This handle h_i is added to the list of handles of the current guiding curve $\mathcal{C}_j = \{h_i\}_{i=0}^{n-1}$. We refer the reader to video 2 of Table 7.1 and to Figure 7.4. Δt is a variable used to register the velocity of the motion of the user which can be modified. However, we found that a time interval Δt of 0.5 up to 1 second is appropriate for most situations. Having handles addition every Δt seconds allows to take into account the amplitude and speed of the motion. As we said in Section 2, up until now, the control of the speed was possible by vector painting and was detached from the guide curves. Here, however, the magnitude of the control vectors, hence the vector fields, is directly coming

from the speed of stroke sketching. This is why one should not take a too large Δt value in order to consider every change but a too small Δt would increase data storage. Between every Δt seconds, the preview of the future obtained line is displayed as a preview guide curve addition to the user, very much like in the sketch-based editor in 2D, see Section 1. When a line is closed (trigger unpressed), the guide curve is added to the set of guide curves $\mathcal{C} = \{C_j\}_{j=0}^{c-1}$ of the IF being edited.

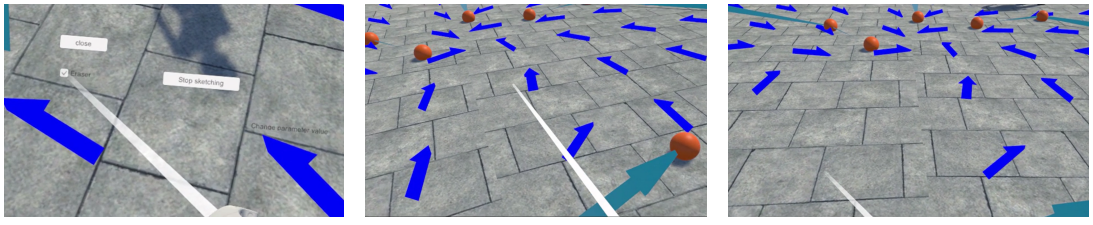
A handle h_i can be displaced easily in VR by selecting it and dragging it, as illustrated in Figure 7.4 and video 3 Table 7.1. This feature can be used to edit any IF that already contains guide curves.

2.3 Acted Guide Curve

From our point of view, an interesting feature of VR was the possibility for the immersed user to actually act out trajectories (vs. sketching them with a controller). Instead of visualizing the interaction's trajectory beforehand in their head, users could act them out, which we expect would improve the realism of the interactions as, according to Olivier et al. [2014], users in VR tend to reproduce trajectories observed in real life. To act interactions, we also offer the possibility to users to move in the scene to create guide curves. Users can just press a trigger and their trajectory is registered while the trigger is pressed by creating a guide curve with a handle every Δt seconds, exactly like sketching guide curves. This means that the speed of the motion of users will impact the velocity vectors of the IF and we benefit from the entire motion. An example of the process is displayed Figure 7.4 and video 5 of Table 7.1.

2.4 Zero Area in VR

Finally, immersed users can also create zero areas that work exactly as in Section 1. Users must first toggle 'Eraser' in the interface (Figure 7.7(a)) and can then draw over the vectors, using the ray, they want to nullify. An example is shown in Figure 7.7 and filmed demonstration is available in video 4 of Table 7.1.



(a) Toggle 'Eraser'. **(b) Before zero area creation.** **(c) After zero area creation.**

Figure 7.7 – An example of zero area designed in VR. First the user activates the eraser function, then the user can erase vectors by pressing the main grip and hovering over the vectors (in blue).

Video	Title	Description	Link
1	Scary Giant in VR	The Scary Giant scenario (presented in Section 1.1) was integrated in VR and showed at the Laval Virtual salon 2022: the immersed user is a giant that scares people behind the trees.	https://youtu.be/tDbr4-i2lqo
2	Sketching IFs in VR	Example of a user sketching guide curves in VR. This video also displays the sketching GUI to pause the simulation and the resulting simulation after sketching the IF.	https://youtu.be/wA-C705_EW8
3	Editing IF in VR	An immersed user can drag and drop the handles of the guide curves. It can be used to modify an already sketched field in the 2D editor.	https://youtu.be/NE-0Zr2nS9U
4	Zero Area in VR	An immersed user can create zero areas and erase vectors.	https://youtu.be/4W2RCF3MAl4
5	Acted Guide Curves	An immersed user can create guide curves with its own trajectory on the IF.	https://youtu.be/2JhSY5h0d9c
6	Sketching an IF with top down view	Demo of the features to sketch an IF in VR using the top down view.	https://youtu.be/F9O4FlA22OU

Table 7.1 – Demonstration videos of IF in VR, presented Chapter 7

3 Observations

This chapter presents the proof of concept of a VR interface for IF. This prototype was not tested by any other users than ourselves and we did not run any experiments on the tool. This section will nonetheless describe our preliminary personal feedback and impression about the VR interface, as well as discuss potential limitations of the implementation of our proof of concept of sketching IF in VR.

Our first major observation testing the VR interface is that close interaction seems to be easier to understand in VR. In the user study presented in Section 2, we observed that users had trouble strictly following the distances given as part of the task (see Table 6.3 and video 7 of Table 6.4). Our current impression is that the scale of the IFs might be harder to assimilate in the 2D editor, and we therefore expect it to be easier to give sense to them in VR, by facilitating subtle interactions design when being part of them at one's scale. For example, since the designer is now at the scale of the environment (and of the other agents), appreciating personal distances should be easier than with the 2D editor. In a few clicks, the user can check if the interaction is invasive and edit accordingly. We believe that apprehending such personal and subtle distances and being able to edit them quickly is a benefit of this new VR interface, and will enable the creation of subtle scenarios more easily. In addition, the design of personal space-related scenarios should be easier in VR because the designer could test out directly his/her own impression of the scenario. This is why we think that VR could be a good tool to refine IFs that were roughly sketched in 2D before. Please, note that we consider the VR interface as an addition to the 2D editor. However, it is interesting to investigate the benefits of IF VR to edit IF on top of the 2D editor outputs.

Even though VR facilitates close-up sketching, it seems, on the other hand, to be more challenging to be precise when sketching from a distance. We can put this observation in relation with the differences

in distance perception in VR studied by Knapp and Loomis [2003]. This is why we gave the users the possibility to see a current top-down view of the screen. We believe that this addition increases the usability of our tool in VR, as it gives the users a global view of the scene but also of their actions. While editing a large IF, they can sketch guide curves while looking at the view or check the view to know quickly what to improve and edit and where to sketch. Nevertheless, we do believe that other display options should be tested out for this view. For example, instead of pressing a button to make it appear or disappear, it could always be shown in the corner of the user's field of view. This would enable users to keep the information constant, more accessible and less cumbersome. However, it would be harder to analyze the view in detail since it would be smaller and not in front of the user anymore. Other extensions can be found; one would be to enable users to sketch on the top-down view, which would then have the optimal orientation according to Arora et al. [2017]. This would mean integrating the 2D editor in a VR setting, combining both benefits. A second option would be to follow Bönsch et al.'s [2020] approach and let users change their height to see to scene from the top when they wish. This solution is computationally simpler but could induce motion sickness or the reduction of subtle interactions discerning since users would not be at the same scale.

We also observed that designing an IF from scratch seems faster using the 2D GUI editor than VR. For instance, the motion amplitude is bigger in VR since the user is at the same scale as the environment. This is also why we believe that defining IFs scenarios entirely in VR would not be as efficient as creating the scenarios in 2D and that combining both is more optimal.

We also proposed in this chapter to enable users to directly act out guide curves based on their own tracked trajectories. We observed that acting a guide curve took more time than sketching one in VR although it seems easier to produce the desired trajectory. We believe that such an addition has a lot of potential, in particular to open the tool to even more people (not at ease with sketching for example) and that the results could be better than directly sketching the curves.

Finally, all these observations are currently only the results of our own tests and have not been scientifically evaluated yet. To this end, we would be interested in conducting user studies to compare using the 2D tool only and adding the VR tool to the 2D editor and to compare the two metaphors of sketching and acting guide curves. We also want to evaluate the usability of our new interface in VR.

The next section will discuss what studies should be developed to investigate the VR interface.

4 User Study Proposal

Based on the works discussed in the previous section, we hypothesize that:

— H1: The realism of the results is higher when acting the guide curves rather than sketching them in VR. It would be good to compare those two metaphors with a study.

— H2: The distance accuracy is higher when IFs are refined in VR (rather than only the 2D editor).

— H3: Usability for immersive scenario design is higher in VR than in the 2D editor because the VR interface eases the trial and error process.

In this section, we discuss several experimental studies that would be valuable to conduct in the future to evaluate the three hypotheses. In particular, we believe it would be interesting to carry out several user studies on different populations (i.e., experts and non-experts), and we present two different user studies to investigate IF VR interface.

The first user study would target non-expert participants to assert whether IF VR is accessible and intuitive to use. This study would also enable us to compare the different VR metaphors (acted vs. sketched guide curves) and interfaces (2D vs. 2D + VR) with many participants.

The second user study would investigate the usability and use of IF VR for expert users that are used to VR interfaces and could benefit from IF in VR.

4.1 Non-Experts User Study

The first study would investigate the efficacy and accessibility of the VR interface with non-expert users. Having this population as participants would enable us to investigate the interface similarly to in Section 2. This study would analyze the experience of non-expert participants with the tool and verify if IF VR is still an accessible method (concerning the results of the user study presented in Section 2). The goal of this study is first to see if numerous participants can easily sketch various scenarios in VR. Secondly, we would like to evaluate the benefits of acting out rather than manually sketching guide curves on the realism of the trajectories generated by IFs. Finally, we would investigate the advantages of VR IF design by novice users and see if they can create more accurate IFs when adding VR rather than in 2D only. In particular, we would also deduce which interface is more accessible to non-experts (between 2D and 2D + VR). Please, note that we do not oppose the 2D editor and the VR interface. Instead, we want to see how both combined can be used by novices. In particular, we want to investigate if refining IFs in VR after 2D sketching leads to more accurate results than using the 2D editor solely.

Protocol

For the first experiment, non-expert users would sketch IFs using the two sketching metaphors:

— 2D editor + ray sketching.

— acted guide curves

They would first be given tutorial videos about 2D and VR (with the ray or by acting), much like in Section 2. Then, they would be asked to create four scenarios using IFs. Those scenarios would be inspired by the ones already presented in Section 2, and would require using both simple and parametric IFs. They also include instructions with precise distances (as in the tasks in Section 2) to measure the differences in accuracy:

— T1: Sketching a Velocity IF making an agent going towards you and stopping at one meter from you.

— T2: Sketching a Velocity IF making an agent walking close to you (three meters close) on your right.

— T3: Sketching a Velocity IF making an agent coming close to you (one meter) when you stand still and running away (seven meters) when you start moving.

— T4: Sketching a Velocity IF making an agent hiding from you behind an obstacle (50 cm from the obstacle).

For each task, participants would follow the instructions of the tasks in the given order. However, they would be given two different interfaces to sketch IFs:

— Sketch IFs on the 2D editor and then refine the IF with ray sketching in VR.

— Sketch IFs entirely by acting out the guide curves.

Those two interfaces would be given in a random order for each task, and the participants would stop the task once satisfied with their results regarding the instructions. In the case of the acted guide curve, they will sketch the IF entirely, and in the other case, they will only edit the already sketched IF to their convenience. After each metaphor, they would answer the following questions on the 5-points Likert scale (from 1: disagree to 5: agree):

— I am satisfied with the realism of the end results trajectory.

— I am satisfied with the ease of completing the task.

After the overall study they would have to answer more questions on the 5-points Likert scale (from 1: disagree to 5: agree):

— I preferred sketching in VR than sketching in 2D.

— I preferred sketching using my own motion than using ray sketching.

Similarly to the user study presented in Section 2, participants will then answer the modified SUS questionnaire (Table 6.2). Those final questions will enable us to evaluate the usability of IF VR. After that they will be given the questions Table 6.2 for both metaphors. Those final questions will allow us to see if the usability is higher in one case or the other.

Data and analysis

The first question set (given after each task) has two goals: first, to check whether participants are satisfied with their design's realism and to evaluate each metaphor's usability. In particular, we expect participants to be more satisfied with the realism using the acted guide curve metaphor. In our opinion, sketching with one own motion is more intuitive because they do not have to learn about the interface, but it is harder to edit and modify.

For quantitative data, we would also compare accuracy in distances (e.g., 1m, 50 cm) between the 2D editor only and when refined in VR after. We think the results would demonstrate that precise distance interactions are better designed after being refined in VR, leading to more accurate results after VR editing. We would also compare the time required to sketch the IFs (2D editor + VR editing) and to act them out (acted guide curve). The second data set's answer would tell us which interface participants preferred using between the 2D editor, ray sketching in VR, and acted guide curves, as it is also interesting to know which interface non-experts prefer using. We believe that non-expert users would be more accustomed to 2D interfaces and might feel more comfortable using the 2D editor. However, this might not be the case for expert participants used to VR settings, which is the focus of the next study.

4.2 Experts User Study

To evaluate the efficacy of IFs and the IF editor for experts, we would like to conduct a separate user study to assert the method's usability in VR. In particular, we are interested in exploring whether IF VR can be used to design immersive scenarios. To answer this question, we will ask a few non-naïve participants to sketch IFs for their own needs. For this study, only a few experts accustomed to immersive scenarios design and animation, such as proxemics researchers or VR game developers that should benefit from IFs, would be required. It is interesting to call on experts for this VR interface because they would be more used to immersive interfaces. First, their impression would be less impacted by the discovery of VR or scenario design, and second, they would understand the challenges of creating for VR. In theory, they would have experienced other interfaces and ways of designing scenarios. They could easily compare this interface with what they already encountered and tell us if it resolves any issues they have faced in the past. Finally, they can test out directly scenarios of their own productions that could be of use to them and expend the test of IF VR.

Protocol

To explore these questions, experts would be asked to follow the same tutorial videos than the ones of the previously described user study so that they can learn about IF. They would be instructed to use the method of their choice to design the reactions of agents towards them, because we think it would be the more interesting to their lines of work. To this end, they would be able to sketch only the field they are the source of when immersed in VR for 4 several instructed scenarios first.

— T1: Sketching a Velocity IF making agents come toward you and stop when close.

— T2: Sketching a Velocity IF making agents following you.

— T3: Sketching a Velocity IF making agents walking beside you as a group.

— T4: Sketching a Velocity IF making agents flee when you go towards them.

At the end of each scenario we would ask them some questions on a 5-points Likert scale (from 1: disagree to 5: agree):

— I am satisfied with the ease of completing the task.

— I am satisfied with the end result.

After the instructed tasks, they would have a 'free task' of 15 min, where they can sketch a few interactions of their choices, giving them all the tools at their disposal. Finally, we would ask them to answer the SUS questions (Table 6.2) about the overall method. A questionnaire would be given as well on a 5-points Likert scale (from 1: disagree to 5: agree):

— I liked using IFs to create interactions.

— I think IFs could be useful in my work.

Finally, we would ask them specific additional open questions:

— How was the experience?

— Could you do what you expected using IF?

— What are the limitations of IF?

— What do you think could be the uses of IF?

— What would be the necessary extensions of IF?

— Do you think IF would enable the design of new types of scenarios?

— Do you think it would be easy to reproduce IF results with another technique?

— How can IF influence the design of immersive scenarios?

— What do you want to do as a designer using IF?

Those questions would lead to a more general discussion and explanation of their answer. We would also ask them for their general impressions, any comments they might have about the tool. In particular, we would ask them for precise feedback on their thoughts about the limitations of IF and possible extensions.

We would also ask them which method they would prefer for sketching IFs.

Data and Analysis

This study would be primarily observational, where we would continuously ask participants to express their thoughts while using the system. Similar to "think-aloud" [Dumas and Redish, 1999; Krahmer and Ummelen, 2004; Norgaard and Hornbæk, 2006] studies, they would share their observations and feedback with us on the fly. In this case, the study would be more like an interview and would be recorded (written notes and video or sound recordings). In addition, their responses to the questionnaires would give us an idea about the ease of use and usability of the tool. Their input would help us improve IF and give us pointers for the future. We would also measure the time they took to complete the task and examine the complexity and number of scenarios they could create during the free task. This study would tell us if IF could be of interest to professionals and what efforts we should make to improve it.

5 Conclusion

In this chapter, we presented a proof of concept of an interface for designing IFs while immersed in VR. To this end, we developed three main new features in VR. A user can access an IF in VR to edit an already sketched IF or sketch directly on an empty IF (previously created with an empty IF file). Users can modify guide curves by moving handles or create new zero ranges. Most importantly, the user can create new guide curves using two metaphors: sketching the guide curve using a ray interactor on the 2D-planned IF or acting on the guide curve directly by recording their own motion. Based on personal observation, we believe that VR IF would facilitate the design of immersive scenarios if it were used in addition to the already existing framework. Indeed, we believe that the 2D trajectories of VR would benefit from using both metaphors. Furthermore, we believe that the accuracy of nearby interactions is higher when VR is used and that using registered human motion to create interactions could increase the realism of the resulting scenarios. In conclusion, we have presented several potential user studies that should be conducted to investigate these hypotheses and validate that IF VR can be used intuitively by many and be valuable for experts.

CHAPTER 8

GENERAL CONCLUSIONS AND PERSPECTIVES

1 Conclusion

In agent-based crowd simulations, the behavior of each agent is usually described via rules and equations. After a review of the related work, we noted that those rules were often mathematically defined for a limited set of interactions, making the design of new interactions complex. The state of the art of authoring crowd techniques showed that none of them was enabling users to design new local interactions or behaviors. However, sketching seems like a powerful tool to create such new behaviors, and previous works on sketching interfaces highlighted the interest of sketching to create a method usable by many. Finally, the review of the use of Virtual Reality in crowd simulation showed that VR was more and more used in this area and that the integration of VR would benefit the design of virtual interactions.

In this thesis, we therefore propose *Interaction Fields (IF)* as an alternative way to model agent behaviors in crowds. IF aims to describe an agent's reaction towards a given object, called the source of the field. As such, an IF is defined relatively to this source and highly depends on its properties (i.e. rotation, position). An IF specifies the velocities or orientations agents should use around the source. In short, an IF is a vector field that rotates and translates along with its source. We developed other mechanisms to build up on this simple notion of IF to create parametric IF. A parametric IF is an IF that changes according to a parameter. This parameter can be a property of the source, like the source's speed or a relationship between objects.

IF is a sketch-based technique, which vector fields can be created by hand-drawing strokes around the source. Combined with an editor that computes IFs based on user sketches, we obtain a system for efficiently and intuitively sketching new agent behaviors. This thesis presented the various features available to sketch the fields by interpolation using guide curves and how to define empty spots called zero areas. IF enables users to intuitively sketch local agent interactions for real-time crowd simulations. The sketches can be edited and modified quickly in real-time, with the resulting simulation available in 2D. To our knowledge, this is the first method with this specific focus. Our scenarios show that interesting simulations can be obtained based on a few simple IF sketches. The value of using IF was demonstrated through several scenarios and our approach validated using a user study. Our user study indicates that

non-expert users can easily use the IF editor to draw agent behaviors that match overall instructions. All participants could design most of the tasks and the usability score was promising. Compared to traditional models, where the behavior of agents can only be influenced via parameters, IFs can be drawn directly and have a more intuitive visual effect on agent behavior. This makes IFs ideal for scenarios where a designer has in mind how agents should move without knowing how to define this motion mathematically. We acknowledge that there are also simulation tasks (such as multi-agent collision avoidance) for which highly successful traditional algorithms already exist. This is why we have shown that IFs can easily be combined with such algorithms; such a combination is more sensitive than attempting to use IFs for everything.

IFs can also easily be combined with other simulation components (such as collision avoidance) without affecting real-time performance. In particular, we demonstrated scenarios that are not easily reproducible with other techniques and not previously found in the literature. Our example scenarios show that complex simulation results can be obtained using a few simple sketches, while IF can be reused easily to multiply scenarios, making it a generic technique.

However, while it is common to visualize a crowd simulation from a 2D top view, we also presented a proof of concept extending IF sketching to VR as we expect it to enable the modeling of subtle personal interactions that are better judged from an agent's perspective, as it was observed in proxemics studies [Bönsch et al., 2018] or when evaluating crowd models [Rojas and Yang, 2013; Rojas et al., 2014a]. This is an important step towards evaluating our approach as well as integrating new advanced features specific to VR. To animate agents in 3D, we selected an on-the-shelf Unity animator that takes as input the trajectories of IF and select a corresponding animation from a data set of captured motion poses. We integrated our simulation into a VR environment where the user itself is embedded as an agent.

Thus, sketching IFs is an effective way to create particular behaviors in a crowd.

The concept of letting agents move according to vector fields is not revolutionary. As mentioned before, our method is inspired by the navigation fields (NFs) of Patil et al. [Patil et al., 2011]. NFs and IFs can both control the *global paths* of agents: an IF with the entire environment as its source is essentially an NF. However, IFs can also model the *local interactions between* agents because of two distinctive properties: they are defined *relative to sources* that can move during the simulation (such as other agents), and they can change according to *parameters* (such as the source's speed or a relation to another object). Our scenarios focused on these distinctive aspects, as well as on the ease of designing IFs visually.

Overall, we are confident that interaction fields give an unprecedented level of creative control over the steering behaviors in crowds. This is an important step towards fully immersive crowd simulations where all agents display human-like behavior, and where the crowd responds realistically to the user's actions. In addition, as highlighted in the introduction through the citation of sociologist Robert E. Park [1967], the extension of social interaction's models is a capital criterion to simulate realistically emerging collective behaviors.

2 Future Work

It is important to understand which types of interactions an IF can or cannot encode. Overall, a non-parametric IF describes an agent's velocity or orientation at different positions relative to another object. A parametric IF can change this behavior according to scalar parameters or to a relation between two objects. In theory, IFs support any behavior that can be described in such terms. However, behaviors depending on *more than two objects* are difficult to encode in an IF. Examples include moving towards the center of a group of agents or avoiding multiple agents at the same time without considering them individually. Other techniques in macroscopic [Treuille et al., 2006] or microscopic [Dutra et al., 2017; Moussaïd et al., 2010; van den Berg et al., 2011] approaches are more suitable for this.

There are a few other limitations to how we currently define and use IFs.

Scenarisation. Section 1 showed many different examples with several scenarios and derivatives. All those examples generalize and apply to many other situations. Section 1.3 and Section 1.4 illustrate one limit of IF as the method lacks ways of scheduling and combining the fields.

As mentioned, IFs are not meant to fully replace traditional algorithms, and techniques can easily be combined. When combining many IFs for different purposes, it is possible that the results average out and agents become indecisive. However, this is also the case for traditional methods, and modeling many behaviors simultaneously is a difficult problem in general.

Similarly, to broaden the application of IF, there is also a need to combine IF with complementary mechanisms for scenarisation, e.g., method for scheduling and a combining fields. To combine IF, we could create combination relationships between them. The basic combination types could be logic ones (e.g. AND, NOR, OR) to carefully set the hierarchical order of the IFs. In this case, when an agent is impacted by overlapping IFs, the hierarchy of impact would be decided by those rules, e.g., one IF would always prevail on others, or two IFs would have an impact only combined together. We could also imagine scenarios in which a specific event, e.g., a specific combination of two IFs, leads to the appearance of a new IF. For example, if two cars collide, a new IF would attract an emergency vehicle or cause other cars to avoid the accident. Another possible event could be a sudden noise that catches people's attention and causes them to react (e.g., an alarm would make office workers leave the building).

Setting up a complete scenario still requires some additional manual work, e.g. assigning IFs to sources and receivers and specifying the behavior profile of each agent. Likewise, there is not yet a reusable system for (de)activating IFs over time; we have scripted this manually for the Museum scenario in Section 1.4. Of course, our software can be improved to a fully integrated design pipeline with such additional features, but this is engineering labor that does not yield new research insights. However, adding the previously mentioned features (combination, time-scheduling, other parameters) is a must to make our tools versatile. A timeline could be added to the 2D editor to manage the time application of every IF, and ultimately, a combination relation should be added between each IF. The time schedule would allow to create lights in the crossroad scenario. We could also add weight to the diverse combina-

tion types to give more flexibility to the users. The combination hierarchy could also depend on certain criteria, e.g. to prioritize certain IFs only if certain conditions are satisfied. For example, turning around someone can be achieved in two different directions, where IF can only take one. To explore IF variants, we could capture different outcomes of an interaction or, better, incorporate variability in behaviors inside IF, but the question about how to simply design or edit this arises. All of those new features should be included carefully in the interface because we want to keep the intuitivity of it. As the interface will become more complex as new features are added, hence less intuitive, we should focus more on HCI to ensure usability.

Variability. IF does not currently bring any variation in the output trajectories: the same IF will give the same trajectory to any agent at any moment. Those are not realistic results and the lack of variability is an often raised issue in crowd simulation literature. Indeed, IFs cannot yet specify *variability*: the same situation will always result in the same agent trajectories. This makes it difficult to, for example, design how agents can move around an obstacle in different ways. One feature inducing variability is the definition of the speed by painting over the vectors, which results are limited. If agents have the same maximum speed, they will react to the field identically. The second option that gives a bit of variability is the ability to choose which group of agents is impacted by which field. Such variability could be added by letting the simulation choose (or interpolate) between multiple IFs. For example, the combination hierarchy could also change according to an agent. For example, if an agent is a car, it could answer in priority to a different field than if it is a pedestrian. This would facilitate the design of the crossroad scenario; a car would stop answering to the environment velocity field if it was impacted by a pedestrian's field and stop immediately.

Another way to bring variety and facilitate the use of IFs for large-scale environments would be to take inspiration from Yersin et al.'s method "Crowd Patches" [Yersin et al., 2005] and build a dataset of IFs. All IFs in this paper have been drawn by hand in the IF editor. However, for some types of behaviors with a clear geometric meaning (e.g., moving to a given point or following an agent), it should be possible to generate *automatically* an IF without guide curves. IFs could be generated based on video footage of real crowds, which may increase the realism of the trajectories as well. By leveraging machine learning, it may even be possible to detect patterns in such data. Another interesting direction for future work would be to enable IF authoring based on text input, e.g., associating words with IFs and building a 'grammar' to describe agent interactions. The goal would be to produce a lot of IFs for each 'sub-behavior'. This would yield a 'palette' of IFs from which many different types of crowds can be composed. IFs would be semantically classified, and a designer could select an IF palette to apply. IFs would be automatically combined according to preset rules that the designer could modify if necessary. The designer would choose behavior palettes through sentences such as: "agent 1 turns around agent 2", "car 1 stops to pick agent 3", and "group 5 follows agent 8". For each interaction, one IF would be randomly selected in the palette. The selected IF would change over time, bringing variation in the trajectories.

Generalisation. In addition, the concept of parametric IFs can still be explored further. As mentioned in Section 2, the IF editor currently only supports a link between the source s and another object o, and this link implies an angle-based relation between s and o. Other examples of scalar parameters could be the local crowd density (for creating density-dependent behavior) and the elapsed simulation time (for creating behavior that evolves temporally). For example, a parameter could be the number of agents inside a field, this would be useful for a photograph scenario, where the photograph would move back to get everyone in the picture according to the number of agents. We also have not yet considered IFs with *multiple* parameters, mostly because drawing keyframes in a higher-dimensional space is less intuitive. Also, we are interested in implementing other types of *object relations*, e.g., to create an IF that depends on the distance between two objects. Ideally, designers could choose any parameter, relationship they would desire.

Realism. IF does not enforce any *smoothness*, and trajectories may be jerky, especially when an agent enters or exits the domain of an IF. Coupling with character animation improves this significantly because the motion is less discontinuous and abrupt using the transition motion capture data set. On a simulation level, we could also consider letting the influence of an IF decay smoothly around the border of its domain. In future work, we plan to explore if/how such extensions can be integrated into a user-friendly tool. The animation of the 3D characters is a crucial aspect of what makes simulated humans expressive. As previously mentioned in Section 2, coupling IF with MxM enables to add personalization and realism of behavior by providing different animations per character. With that, we can multiply the scenarios by using different animation styles for one IF. However, this personalization comes with a cost, as discussed previously in Section 2, as a recording session is time-costing. Several hours for each personalized animation set are required. The quality and the amount of registered data directly impacts the quality of the results, and, as said previously, motion matching rests on this trade-off between the quality of the animation and faithfulness to the input trajectory. For this reason, without unlimited data or additional animation processing (e.g., online style variation such as [Mason et al., 2022; Yumer and Mitra, 2016]), we cannot ensure the quality of the results for any input IF. This means that to ensure the good quality of the animation, a motion capture session should take place for each scenario as well as each desired outlook or emotion (e.g., shy, angry, carefree). A solution to this issue would be to register the motion while sketching the field.

This work is already in progress since the trajectory of users can already be used to sketch a field. We could resolve both issues if we equip users with a motion capture suit while sketching the field. This solution could already be tested, but a lighter setup that the motion capture suit would make it easier to use and more approachable. An already existing possibility is to track several joints (e.g., head, hands, foot) and reconstruct the animation [Spirops, 2010] and add more trackers to obtain more faithful results [Lecon et al., 2021].

Nevertheless, using Motion Matching it is important to point out that the captured motions cannot

be used immediately after capture since Motion Matching must process them in its library before the scenario is executed. This means that the trial and error process would not be available anymore.

Another limitation of Motion Matching is that the memory usage (and runtime performance to some degree) scales linearly with the amount of data that is provided and the number of features to be matched. This results in a constant balance between the expressiveness of the system, the quality of the results, and the real-world memory and performance budgets. This leaves developers unable to combine it with powerful data processing methods such as automatic data augmentation. Unfortunately, this leads to a constant trade-off between the system's scalability and real work production requirements. This is why we ultimately doubt that motion matching is the optimal solution and other algorithms should be tested, where fewer animation data are necessary, using such tool as machine learning, for example, [Holden et al., 2020]. This requires more development and integration effort, and this is not where the contributions of this thesis lay.

However, we believe we could take out even more advantages from the VR interface. As previously said, all the future extensions of IF would be tricky to integrate into an intuitive 2D editor. Using the third dimension of a 3D interface could open new possibilities. Acting guide curves in VR is an example, as registering more data while acting would improve the interaction. A first example would be to track the trajectory and the orientation of the user at the same time to build up velocity and orientation IFs in one recording.To extend this principle, new Ifs could be defined for different properties of the body (e.g., head, torso orientation) or even for body joints (e.g., foot, hands, hips). Each interaction would be composed of layers of different IF joint types that follow different rules depending on the body part, thus forming 3D IF. For example, instead of just defining orientation on a 2D plane to make agents look at a painting, we could build a 3D IF determining the head agents' rotation in front of the painting. Another example would be an IF emitted by the handle of a door that would make agents open it by controlling the velocity of its hand joint in 3D. Various animations can be obtained by combining several 3D IF controlling different joints, although we would need to ensure the smoothness of the trajectories between 3D IFs switching. We could also describe interactions between 3D characters. For example, the shoulders of a character could emit a 3D IF, attracting other agents' hands to tap on it. 3D IF would necessarily require time-scheduling between fields to switch between animations. In other words, an immersed designer could act the reaction to virtual objects or humans in 3D. Acting out trajectories is an even easier way of defining interactions, and we hope it would keep IF intuitive to use. The processing time and animation techniques to animate 3D IFs (e.g., inverse kinematic) trajectories should also be more looked into. In addition, it would intertwine the trajectory of the IF and the animation of the results, combining both processes. As a result, we believe the output realism of the trajectory would be superior to IFs sketched in 2D as well.

We believe that exploring such a potential direction is interesting for bringing the creation of 3D content to a new level of realism and intuitivity.

LIST OF PUBLICATIONS

[1] A. Colas, W. van Toll, K. Zibrek, L. Hoyet, A.-H. Olivier, and J. Pettré, "Interaction Fields: Intuitive Sketch-based Steering Behaviors for Crowd Simulation," Computer Graphics Forum, pp. 1–14, Apr. 2022. [Online]. Available: https://hal.inria.fr/hal-03642462

[2] A. Colas, W. van Toll, L. Hoyet, C. Pacchierotti, M. Christie, K. Zibrek, A.-H. Olivier, and J. Pettré, "Interaction Fields: Sketching Collective Behaviours," MIG 2020: Motion, Interaction, and Games, Oct. 2020, poster. [Online]. Available: https://hal.inria.fr/hal-02969013

BIBLIOGRAPHY

Ahn, J., Wang, N., Thalmann, D., & Boulic, R. (2012). Within-crowd immersive evaluation of collision avoidance behaviors. *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, 231–238. https://doi.org/10.1145/2407516.2407573

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., & Savarese, S. (2016). Social LSTM: human trajectory prediction in crowded spaces. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 961–971.

Allain, P., Courty, N., & Corpetti, T. (2014). Optimal crowd editing. *Graphical Models*, *76*(1), 1–16.

Allbeck, J. (2010). CAROSA: A Tool for Authoring NPCs. *International Conference on Motion in Games*, 182–193. https://doi.org/10.1007/978-3-642-16958-8_18

Allen, T., Parvanov, A., Knight, S., & Maddock, S. (2015). Using Sketching to Control Heterogeneous Groups. In R. Borgo & C. Turkay (Eds.), *Computer graphics and visual computing (cgvc)*. The Eurographics Association. https://doi.org/10.2312/cgvc.20151249

Amirian, J., van Toll, W., Hayet, J.-B., & Pettré, J. (2019). Data-driven crowd simulation with generative adversarial networks. *Proc. 32nd Int. Conf. Computer Animation and Social Agents*, 7–10.

Animation Uprising. (2020). Motion Matching for Unity. https://assetstore.unity.com/packages/tools/animation/motion-matching-for-unity-145624

Arora, R., Kazi, R. H., Anderson, F., Grossman, T., Singh, K., & Fitzmaurice, G. (2017). Experimental evaluation of sketching on surfaces in vr. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 5643–5654. https://doi.org/10.1145/3025453.3025474

Arora, R., Kazi, R. H., Kaufman, D. M., Li, W., & Singh, K. (2019). Magicalhands: mid-air hand gestures for animating in vr. *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, 463–477. https://doi.org/10.1145/3332165.3347942

Assila, A., Ezzedine, H., et al. (2016). Standardized usability questionnaires: features and quality focus. *Electronic Journal of Computer Science and Information Technology: eJCIST*, *6*(1).

Badler, N. I., Bindiganavale, R., Allbeck, J. M., Schuler, W., Zhao, L., & Palmer, M. (1998). A Parameterized Action Representation for Virtual Human Agents. *EMBODIED CONVERSATIONAL AGENTS*.

Bangor, A., Kortum, P., & Miller, J. (2008). The system usability scale (SUS): an empirical evaluation. *International Journal of Human-Computer Interaction*, *24*(6), 574–594.

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual sus scores mean: adding an adjective rating scale. *J. Usability Stud.*, *4*, 114–123.

Barnett, A. (2014). Topology based global crowd control. *CISA 2013 Proceedings for the 26th International Conference on Computer Animation and Social Agents*.

Bassett, K., Baran, I., Schmid, J., Gross, M., & Sumner, R. W. (2013). Authoring and animating painterly characters. *ACM Trans. Graph.*, *32*(5). https://doi.org/10.1145/2484238

Bergig, O., Hagbi, N., El-Sana, J., & Billinghurst, M. (2009). In-place 3d sketching for authoring and augmenting mechanical systems. *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, 87–94. https://doi.org/10.1109/ISMAR.2009.5336490

Berton, F., Olivier, A., Bruneau, J., Hoyet, L., & Pettré, J. (2019). Studying gaze behaviour during collision avoidance with a virtual walker: influence of the virtual reality setup. *IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2019, Osaka, Japan, March 23-27, 2019*, 717–725. https://doi.org/10.1109/VR.2019.8798204

Bhattacharjee, S., & Chaudhuri, P. (2020). A survey on sketch based content creation: from the desktop to virtual and augmented reality. *Computer Graphics Forum*, *39*(2), 757–780. https://doi.org/https://doi.org/10.1111/cgf.14024

Boatright, C. D., Kapadia, M., Shapira, J. M., & Badler, N. I. (2015). Generating a multiplicity of policies for agent steering in crowd simulation. *Computer Animation and Virtual Worlds*, *26*(5), 483–494.

Bönsch, A., Barton, S. J., Ehret, J., & Kuhlen, T. W. (2020). Immersive sketching to author crowd movements in real-time. *Proc. ACM International Conference on Intelligent Virtual Agents*, 1–3.

Bönsch, A., Radke, S., Overath, H., Asché, L. M., Wendt, J., Vierjahn, T., Habel, U., & Kuhlen, T. W. (2018). Social vr: how personal space is affected by virtual agents' emotions. *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 199–206. https://doi.org/10.1109/VR.2018.8446480

Braun, A., Musse, S., de Oliveira, L., & Bodmann, B. (2003). Modeling individual behaviors in crowd simulation. *Proceedings 11th IEEE International Workshop on Program Comprehension*, 143–148. https://doi.org/10.1109/CASA.2003.1199317

Brooke, J. (1995). SUS: a quick and dirty usability scale. *Usability Eval. Ind.*, *189*.

Brooke, J. (1996). SUS: a quick and dirty usability scale. *Usability Evaluation in Industry*, *189*.

Brooke, J. (2013). SUS: a retrospective. *Journal of Usability Studies*, *8*, 29–40.

Bruneau, J., Dutra, T. B., & Pettré, J. (2014). Following behaviors: a model for computing following distances [The Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014), 22-24 October 2014, Delft, The Netherlands]. *Transportation Research Procedia*, *2*, 424–429. https://doi.org/https://doi.org/10.1016/j.trpro.2014.09.049

Bulbul, A., & Dahyot, R. (2017). Populating virtual cities using social media. *Computer Animation and Virtual Worlds*, *28*(5), e1742.

Buttnër, M., & Clavet, S. (2015). Motion matching - the road to next gen animation. https://www.youtube.com/watch?v=z_wpgHFSWss&t=658s

Cafaro, A., Ravenet, B., Ochs, M., Vilhjálmsson, H. H., & Pelachaud, C. (2016). The effects of interpersonal attitude of a group of agents on user's presence and proxemics behavior. *ACM Transactions on Interactive Intelligent Systems*, *6*(2).

Card, S. K., Newell, A., & Moran, T. P. (1983). The psychology of human-computer interaction. *L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1983*.

Casadiego, L., & Pelechano, N. (2015). From one to many: simulating groups of agents with reinforcement learning controllers. *Proc. Int. Conf. Intelligent Virtual Agents*, 119–123.

Charalambous, P., Karamouzas, I., Guy, S. J., & Chrysanthou, Y. (2014). A data-driven framework for visual crowd analysis. *Comput. Graph. Forum*, *33*(7), 41–50.

Chaudhuri, P., Kalra, P. K., & Banerjee, S. (2004). A system for view-dependent animation. *Computer Graphics Forum*, *23*.

Chen, C.-Y., Wong, S.-K., & Liu, W.-Y. (2020). Generation of small groups with rich behaviors from natural language interface. *Computer Animation and Virtual Worlds*, *31*(4-5), e1960.

Chenney, S. (2004). Flow Tiles. In R. Boulic & D. K. Pai (Eds.), *Symposium on computer animation*. The Eurographics Association. https://doi.org/10.2312/SCA/SCA04/233-242

Choi, B., Blanco i Ribera, R., Lewis, J. P., Seol, Y., Hong, S., Eom, H., Jung, S., & Noh, J. (2016). SketchiMo: sketch-based motion editing for articulated characters. *ACM Transactions on Graphics*, *35*(4). https://doi.org/10.1145/2897824.2925970

Costa, M. (2010). Interpersonal distances in group walking. *Journal of Nonverbal Behavior*, *34*, 15–26. https://doi.org/10.1007/s10919-009-0077-y

Curtis, S., Best, A., & Manocha, D. (2016). Menge: a modular framework for simulating crowd movement. *Collective Dynamics*, *1*(A1), 1–40.

Davis, J., Agrawala, M., Chuang, E., Popovic, Z., & Salesin, D. (2003). A sketching interface for articulated figure animation. *SCA '03*.

Degond, P., Navoret, L., Bon, R., & Sanchez, D. (2010). Congestion in a macroscopic model of self-driven particles modeling gregariousness. *Journal of Statistical Physics*, *138*(1), 85–125. https://doi.org/10.1007/s10955-009-9879-x

Dickinson, P., Gerling, K., Hicks, K., Murray, J. C., Shearer, J., & Greenwood, J. (2019). Virtual reality crowd simulation: effects of agent density on user experience and behaviour. *Virtual Real.*, *23*(1), 19–32. https://doi.org/10.1007/s10055-018-0365-0

Dudley, J. J., Schuff, H., & Kristensson, P. O. (2018). Bare-handed 3d drawing in augmented reality. *Proceedings of the 2018 Designing Interactive Systems Conference*, 241–252. https://doi.org/10.1145/3196709.3196737

Dumas, J. S., & Redish, J. C. (1999). *A practical guide to usability testing* (1st). Intellect Books.

Durupinar, F., Pelechano, N., Allbeck, J., Güdükbay, U., & Badler, N. I. (2011). How the Ocean Personality Model Affects the Perception of Crowds. *IEEE Computer Graphics and Applications*, *31*(3), 22–31. https://doi.org/10.1109/MCG.2009.105

Durupınar, F., Güdükbay, U., Aman, A., & Badler, N. I. (2015). Psychological parameters for crowd simulation: From audiences to mobs. *IEEE transactions on visualization and computer graphics*, *22*(9), 2145–2159.

Dutra, T. B., Marques, R., Cavalcante-Neto, J. B., Vidal, C. A., & Pettré, J. (2017). Gradient-based steering for vision-based crowd simulation algorithms. *Computer Graphics Forum*, *36*(2), 337–348.

Eysenck, H. J. (1985). *Personality and individual differences : a natural science approach*. Plenum Press.

Federici, M. L., Gorrini, A., Manenti, L., & Vizzari, G. (2012). Data collection for modeling and simulation: case study at the university of milan-bicocca. In G. C. Sirakoulis & S. Bandini (Eds.), *Cellular automata* (pp. 699–708). Springer Berlin Heidelberg.

Funge, J. D., Tu, X., & Terzopoulos, D. (1999). Cognitive modeling: knowledge, reasoning and planning for intelligent characters. *SIGGRAPH '99*.

Gérin-Lajoie, M., Richards, C. L., Fung, J., & McFadyen, B. J. (2008). Characteristics of personal space during obstacle circumvention in physical and virtual environments. *Gait & posture*, *27*(2), 239–247.

Gonzalez, L. R. M., & Maddock, S. C. (2017). Sketching for Real-time Control of Crowd Simulations. *CGVC*.

Gu & Deng. (2013). Generating freestyle group formations in agent-based crowd simulations. *IEEE Computer Graphics and Applications*, *33*(1), 20–31. https://doi.org/10.1109/MCG.2011.87

Gu, Q., & Deng, Z. (2011). Formation sketching: an approach to stylize groups in crowd simulation. *Graphics Interface*.

Guay, M., Cani, M.-P., & Ronfard, R. (2013). The line of action: an intuitive interface for expressive character posing. *ACM Transactions on Graphics*, *32*(6). https://doi.org/10.1145/2508363.2508397

Guay, M., Ronfard, R., Gleicher, M., & Cani, M.-P. (2015). Space-time sketching of character animation. *ACM Transactions on Graphics*, *34*(4). https://doi.org/10.1145/2766893

Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., & Alahi, A. (2018). Social GAN: socially acceptable trajectories with generative adversarial networks. *2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2255–2264.

Gupta, H., & Chaudhuri, P. (2018). Sheetanim - from model sheets to 2d hand-drawn character animation -. *VISIGRAPP*.

Guy, S. J., Chhugani, J., Curtis, S., Dubey, P., Lin, M. C., & Manocha, D. (2010). PLEdestrians: a least-effort approach to crowd simulation. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 119–128.

Guy, S. J., Kim, S., Lin, M. C., & Manocha, D. (2011). Simulating heterogeneous crowd behaviors using personality trait theory. *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, 43–52.

Hahn, F., Mutzel, F., Coros, S., Thomaszewski, B., Nitti, M., Gross, M., & Sumner, R. W. (2015). Sketch abstractions for character posing. *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 185–191. https://doi.org/10.1145/2786784.2786785

Haworth, B., Berseth, G., Moon, S., Faloutsos, P., & Kapadia, M. (2020). Deep integration of physical humanoid control and crowd navigation. *Proc. 13th ACM SIGGRAPH Conf. Motion, Interaction and Games*.

He, L., Pan, J., Narang, S., Wang, W., & Manocha, D. (2016). Dynamic Group Behaviors for Interactive Crowd Simulation. *Symposium on Computer Animation*.

Heater, C. (1992). Being there: the subjective experience of presence. *Presence Teleoperators Virtual Environ.*, *1*(2), 262–271.

Helbing, D., Farkas, I., & Vicsek, T. (2000). Simulating dynamical features of escape panic. *Nature*, *407*, 487–490.

Helbing, D., & Molnár, P. (1995). Social force model for pedestrian dynamics. *Physical Review E*, *51*(5), 4282–4286.

Henry, J., Shum, H., & Komura, T. (2012). Environment-aware real-time crowd control. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 193–200.

Henry, J., Shum, H., & Komura, T. (2014). Interactive formation control in complex environments. *IEEE Transactions on Visualization and Computer Graphics*, *20*, 211–222. https://doi.org/10.1109/TVCG.2013.116

Holden, D., Kanoun, O., Perepichka, M., & Popa, T. (2020). Learned motion matching. *ACM Trans. Graph.*, *39*(4). https://doi.org/10.1145/3386569.3392440

Huang, L., Gong, J., Li, W., Xu, T., Shen, S., Liang, J., Feng, Q., Zhang, D., & Sun, J. (2018). Social force model-based group behavior simulation in virtual geographic environments. *ISPRS International Journal of Geo-Information*, *7*, 79. https://doi.org/10.3390/ijgi7020079

Huang, W. H., Fajen, B. R., Fink, J. R., & Warren, W. H. (2006). Visual navigation and obstacle avoidance using a steering potential function. *Robotics and Autonomous Systems*, *54*, 288–299.

Hughes, R., Ondřej, J., & Dingliana, J. (2015). Holonomic collision avoidance for virtual crowds. *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 103–111.

ISO/IEC 25010. (2011). *ISO/IEC 25010:2011, systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models*.

Jain, E., Sheikh, Y., & Hodgins, J. (2009). Leveraging the talent of hand animators to create three-dimensional animation. *Computer Animation, Conference Proceedings*, 93–102. https://doi.org/10.1145/1599470.1599483

Jain, E., Sheikh, Y., Mahler, M., & Hodgins, J. (2012). Three-dimensional proxies for hand-drawn characters. *ACM Trans. Graph.*, *31*(1), 8:1–8:16.

Jan, D., & Traum, D. R. (2007). Dynamic movement and positioning of embodied agents in multiparty conversations. *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*. https://doi.org/10.1145/1329125.1329142

Jin, X., Xu, J., Wang, C. C. L., Huang, S., & Zhang, J. (2008). Interactive Control of Large-Crowd Navigation in Virtual Environments Using Vector Fields. *IEEE Computer Graphics and Applications*, *28*.

Jordao, K., Charalambous, P., Christie, M., Pettré, J., & Cani, M. (2015). Crowd art: density and flow based crowd motion design. *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*.

Jordao, K., Pettré, J., Christie, M., & Cani, M.-P. (2014). Crowd sculpting: a space-time sculpting method for populating virtual environments. *Computer Graphics Forum*, *33*(2), 351–360.

Ju, E., Choi, M. G., Park, M., Lee, J., Lee, K. H., & Takahashi, S. (2010). Morphable crowds. *ACM Transactions on Graphics*, *29*(6), 1–10.

Kang, S. J., Kim, S.-K., et al. (2014). Crowd control with vector painting. *Journal of Research and Practice in Information Technology*, *46*(2-3), 119.

Kang, S., & Kim, S. (2014). Crowd control with vector painting. *J. Res. Pract. Inf. Technol.*, *46*.

Kapadia, M., Singh, S., Hewlett, W., & Faloutsos, P. Egocentric affordance fields in pedestrian steering. In: In *Proceedings of the 2009 symposium on interactive 3d graphics, si3d 2009, february 27 - march 1, 2009, boston, massachusetts, usa*. 2009, January, 215–223. https://doi.org/10.1145/1507149.1507185

Kapadia, M., Singh, S., Reinman, G., & Faloutsos, P. (2011). A behavior authoring framework for multiactor simulations. *IEEE Computer Graphics and Applications*, *31*. https://doi.org/10.1109/MCG.2011.68

Kara, L. B., Shimada, K., & Marmalefsky, S. D. (2007). An evaluation of user experience with a sketch-based 3d modeling system. *Computers & Graphics*, *31*(4), 580–597. https://doi.org/https://doi.org/10.1016/j.cag.2007.04.004

Karamouzas, I., Heil, P., Van Beek, P., & Overmars, M. H. (2009). A predictive collision avoidance model for pedestrian simulation. *Proc. International Workshop on Motion in Games*, 41–52.

Karamouzas, I., & Overmars, M. H. (2010). A velocity-based approach for simulating human collision avoidance. *Proc. International Conference on Intelligent Virtual Agents*, 180–186.

Karamouzas, I., & Overmars, M. H. (2012). Simulating and evaluating the local behavior of small pedestrian groups. *IEEE Trans. Vis. Comput. Graphics*, *13*, 394–406.

Karamouzas, I., Skinner, B., & Guy, S. J. (2014). Universal power law governing pedestrian interactions. *Physical Review Letters*, *113*, 238701:1–5.

Karmakharm, T., Richmond, P., & Romano, D. M. (2010). Agent-based Large Scale Simulation of Pedestrians With Adaptive Realistic Navigation Vector Fields. *TPCG*.

Kendon, A. (1990). *Conducting interaction: patterns of behavior in focused encounters*. Cambridge University Press.

Kielar, P. M., Biedermann, D. H., & Borrmann, A. (2016). *MomenTUMv2: a modular, extensible, and generic agent-based pedestrian behavior simulation framework* (tech. rep. No. TUM-I1643). Technische Universität München, Institut für Informatik.

Kim, J., & Lee, J. (2016). Interactive editing of crowd animation. In *Simulating heterogeneous crowds with interactive behaviors* (pp. 115–130). AK Peters/CRC Press.

Kim, J., Seol, Y., Kwon, T., & Lee, J. (2014). Interactive manipulation of large-scale crowd animation. *ACM Transactions on Graphics*, *33*(4), 1–10.

Kim, M., Hwang, Y., Hyun, K., & Lee, J. (2012). Tiling motion patches. *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 117–126.

Kim, M., Hyun, K., Kim, J., & Lee, J. (2009). Synchronized Multi-Character Motion Editing. *ACM Trans. Graph.*, *28*(3). https://doi.org/10.1145/1531326.1531385

Kim, S., Bera, A., Best, A., Chabra, R., & Manocha, D. (2016). Interactive and adaptive data-driven crowd simulation. *2016 IEEE Virtual Reality (VR)*, 29–38. https://doi.org/10.1109/VR.2016.7504685

Knapp, J., & Loomis, J. (2003). Visual perception of egocentric distance in real and virtual environments. https://doi.org/10.1201/9781410608888.pt1

Koilias, A., Nelson, M. G., Anagnostopoulos, C.-N., & Mousas, C. (2020). Immersive walking in a virtual crowd: the effects of the density, speed, and direction of a virtual crowd on human movement behavior. *Computer Animation and Virtual Worlds*, *31*(6), e1928. https://doi.org/https://doi.org/10.1002/cav.1928

Kraayenbrink, N., Kessing, J., Tutenel, T., Haan, G. D., & Bidarra, R. (2014). Semantic crowds. *Entertain. Comput.*, *5*, 297–312.

Krahmer, E., & Ummelen, N. (2004). Thinking about thinking aloud: a comparison of twoverbal protocols for usability testing. *Professional Communication, IEEE Transactions on*, *47*, 105–117. https://doi.org/10.1109/TPC.2004.828205

Kremyzas, A., Jaklin, N. S., & Geraerts, R. (2016). Towards social behavior in virtual-agent navigation. *Science China - Information Sciences*, *59*(11), 112102.

Krontiris, A., Bekris, K. E., & Kapadia, M. (2016). ACUMEN: Activity-centric crowd authoring using influence maps. *Proceedings of the 29th International Conference on Computer Animation and Social Agents*, 61–69.

Kwon, T., Lee, K. H., Lee, J., & Takahashi, S. (2008). Group motion editing. *ACM Transactions on Graphics*, *27*(3), 1–8.

Kyriakou, M., Pan, X., & Chrysanthou, Y. (2017). Interaction with virtual crowd in immersive and semi-immersive virtual reality systems. *Computer Animation and Virtual Worlds*, *28*(5), e1729.

Laval Virtual. (2022). Exhibitors laval virtual 2022. https://www.laval-virtual.com/exhibitors-2022/

Lavergne, F., Wendehenne, H., Bäuerle, T., & Bechinger, C. (2019). Group formation and cohesion of active particles with visual perception–dependent motility. *Science*, *364*, 70–74. https://doi.org/10.1126/science.aau5347

Lecon, C., Engel, B., & Schneider, L. (2021). Vr live motion capture. *2021 16th International Conference on Computer Science & Education (ICCSE)*, 144–149. https://doi.org/10.1109/ICCSE51940.2021.9569641

Ledo, D., Houben, S., Vermeulen, J., Marquardt, N., Oehlberg, L., & Greenberg, S. (2018). Evaluation strategies for hci toolkit research. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–17. https://doi.org/10.1145/3173574.3173610

Lee, J., Won, J., & Lee, J. (2018). Crowd simulation by deep reinforcement learning. *Proc. 11th ACM SIGGRAPH Conf. Motion, Interaction and Games*.

Lee, K. H., Choi, M. G., Hong, Q., & Lee, J. (2007a). Group behavior from video: a data-driven approach to crowd simulation. *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, 109–118.

Lee, K. H., Choi, M. G., Hong, Q., & Lee, J. (2007b). Group behavior from video: a data-driven approach to crowd simulation. *SCA '07*.

Lee, K. H., Choi, M. G., & Lee, J. (2006). Motion Patches: Building Blocks for Virtual Environments Annotated with Motion Data. *ACM SIGGRAPH 2006 Papers*, 898–906. https://doi.org/10.1145/1179352.1141972

Lemercier, S., & Auberlet, J.-M. (2015). Towards more behaviours in crowd simulation. *Computer Animation and Virtual Worlds*, *27*. https://doi.org/10.1002/cav.1629

Lemercier, S., Jelic, A., Kulpa, R., Hua, J., Fehrenbach, J., Degond, P., Appert-Rolland, C., Donikian, S., & Pettré, J. (2012). Realistic following behaviors for crowd simulation. *Computer Graphics Forum*, *31*(2), 489–498. https://doi.org/10.1111/j.1467-8659.2012.03028.x

Lemonari, M., Blanco, R., Charalambous, P., Pelechano, N., Avraamides, M., Pettré, J., & Chrysanthou, Y. (2022). Authoring virtual crowds: a survey. *Computer Graphics Forum*, *41*(2), 677–701. https://doi.org/https://doi.org/10.1111/cgf.14506

Lerner, A., Chrysanthou, Y., & Lischinski, D. (2007). Crowds by example. *Comput. Graph. Forum*, *26*(3), 655–664.

Lewis, J. R. (2018). Measuring perceived usability: the csuq, sus, and umux. *International Journal of Human–Computer Interaction*, *34*(12), 1148–1156. https://doi.org/10.1080/10447318.2017.1418805

Lewis, J. R., & Sauro, J. (2018). Item benchmarks for the system usability scale. *Journal of Usability Studies*, *13*(3).

Li, W., & Allbeck, J. M. (2011). Populations with purpose. *International Conference on Motion in Games*, 132–143.

Li, Y., Liu, H., Liu, G.-p., Li, L., Moore, P., & Hu, B. (2017). A grouping method based on grid density and relationship for crowd evacuation simulation. *Physica A: Statistical Mechanics and its Applications*, *473*, 319–336. https://doi.org/https://doi.org/10.1016/j.physa.2017.01.008

Liu, B., Liu, H., Zhang, H., & Qin, X. (2018). A social force evacuation model driven by video data. *Simulation Modelling Practice and Theory*, *84*, 190–203. https://doi.org/https://doi.org/10.1016/j.simpat.2018.02.007

Liu, W.-Y., Wong, S.-K., & Chen, C.-Y. (2020). A natural language interface with casual users for crowd animation. *Computer Animation and Virtual Worlds*, *31*(4-5), e1965.

Llobera, J., Spanlang, B., Ruffini, G., & Slater, M. (2010). Proxemics with multiple dynamic characters in an immersive virtual environment. *ACM Trans. Appl. Percept.*, *8*(1), 3:1–3:12. https://doi.org/10.1145/1857893.1857896

López, A., Chaumette, F., Marchand, E., & Pettré, J. (2019). Character navigation in dynamic environments based on optical flow. *Computer Graphics Forum*, *38*(2), 181–192.

Machado, T., Gomes, A., & Walter, M. (2009). A comparison study: sketch-based interfaces versus wimp interfaces in three dimensional modeling tasks, 29–35. https://doi.org/10.1109/LA-WEB.2009.22

Mason, I., Starke, S., & Komura, T. (2022). Real-time style modelling of human locomotion via feature-wise transformations and local motion phases. *Proc. ACM Comput. Graph. Interact. Tech.*, *5*(1). https://doi.org/10.1145/3522618

Mathew, T., Benes, B., & Aliaga, D. (2020). Interactive inverse spatio-temporal crowd motion design. *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 1–9.

Maury, B., Roudneff-Chupin, A., & Santambrogio, F. (2010). A macroscopic crowd motion model of gradient flow type. https://doi.org/10.48550/ARXIV.1002.0686

McIlveen, J., Maddock, S. C., Heywood, P., & Richmond, P. (2016). Ped: Pedestrian Environment Designer. *Proc. Conference on Computer Graphics & Visual Computing*, 105–112.

Meerhoff, L. A., Pettré, J., Lynch, S. D., Crétual, A., & Olivier, A.-H. (2018). Collision avoidance with multiple walkers: sequential or simultaneous interactions? *Frontiers in psychology*, *9*, 2354.

Millán, E., & Rudomin, I. (2005). Agent paint: intuitive specification and control of multiagent animations. *Proc. International Conference on Computer Animation and Social Agents*, *2*.

Montana Gonzalez, L., & Maddock, S. (2019). A sketch-based interface for real-time control of crowd simulations that use navigation meshes. *Proc. International Conference on Computer Graphics Theory and Applications*, *1*, 41–52.

Moussaïd, M., Helbing, D., & Theraulaz, G. (2011). How simple rules determine pedestrian behavior and crowd disasters. *Proc. National Academy of Sciences*, *108*, 6884–6888.

Moussaïd, M., Kapadia, M., Thrash, T., Sumner, R. W., Gross, M., Helbing, D., & Hölscher, C. (2016). Crowd behaviour during high-stress evacuations in an immersive virtual environment. *Journal of The Royal Society Interface*, *13*(122), 20160414.

Moussaïd, M., Perozo, N., Garnier, S., Helbing, D., & Theraulaz, G. (2010). The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PloS one*, *5*(4), e10047.

Musse, S. R., & Thalmann, D. (2001). Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics*, *7*(2), 152–164. https://doi.org/10.1109/2945.928167

Nelson, M., Koilias, A., Gubbi, S., & Mousas, C. (2019). Within a virtual crowd: exploring human movement behavior during immersive virtual crowd interaction. *The 17th International Conference on Virtual-Reality Continuum and Its Applications in Industry*. https://doi.org/10.1145/3359997.3365709

Nicolas, A., & Hafinaz, F. (2021). Social groups in pedestrian crowds: review of their influence on the dynamics and their modelling. https://doi.org/10.48550/ARXIV.2107.13293

Norgaard, M., & Hornbæk, K. (2006). What do usability evaluators do in practice? an explorative study of think-aloud testing. *Proceedings of the 6th Conference on Designing Interactive Systems*, 209–218. https://doi.org/10.1145/1142405.1142439

Normoyle, A., Likhachev, M., & Safonova, A. (2014). Stochastic activity authoring with direct user control. *ACM SIGGRAPH Symposium on Interactive 3D Graphics*, 31–38. https://doi.org/10.1145/2556700.2556714

Olivier, A.-H., Bruneau, J., Cirio, G., & Pettré, J. (2014). A virtual reality platform to study crowd behaviors. *Transportation research procedia*, *2*, 114–122.

Olivier, A., Bruneau, J., Kulpa, R., & Pettré, J. (2018). Walking with virtual people: evaluation of locomotion interfaces in dynamic environments. *IEEE Trans. Vis. Comput. Graph.*, *24*(7), 2251–2263. https://doi.org/10.1109/TVCG.2017.2714665

Ondřej, J., Pettré, J., Olivier, A.-H., & Donikian, S. (2010). A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics*, *29*(4), 123.

Oshita, M., & Ogiwara, Y. (2009a). Sketch-Based Interface for Crowd Animation. In A. Butz, B. Fisher, M. Christie, A. Krüger, P. Olivier, & R. Therón (Eds.), *Smart graphics* (pp. 253–262). Springer Berlin Heidelberg.

Oshita, M., & Ogiwara, Y. (2009b). Sketch-based interface for crowd animation. *International Symposium on Smart Graphics*, 253–262.

Ozgur, O. (2010). Local interactions. *Handbook of Social Economics*, *1*. https://doi.org/10.2139/ssrn.1682067

Paravisi, M., Werhli, A., Junior, J. J., Rodrigues, R., Jung, C. R., & Musse, S. R. (2008). Continuum crowds with local control. *Proc. Computer Graphics International*, 108–115. https://doi.org/10.1145/800186.810616

Paris, S., Pettré, J., & Donikian, S. (2007). Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum*, *26*(3), 665–674.

Park, J., Rojas, F., & Yang, H. (2013). A collision avoidance behavior model for crowd simulation based on psychological findings. *Computer Animation and Virtual Worlds*, *24*, 173–183. https://doi.org/10.1002/cav.1504

Park, M. J. (2010). Guiding Flows for Controlling Crowds. *Vis. Comput.*, *26*(11), 1383–1391. https://doi.org/10.1007/s00371-009-0415-4

Park, R. E. (1967). *On social control and collective behavior: selected papers* (Vol. 275). Chicago: University of Chicago Press.

Patel, P., Gupta, H., & Chaudhuri, P. (2016). Tracemove: a data-assisted interface for sketching 2d character animation. *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications: Volume 1: GRAPP*, 191–199. https://doi.org/10.5220/0005672501890197

Patil, S., van den Berg, J. P., Curtis, S., Lin, M. C., & Manocha, D. (2011). Directing crowd simulations using navigation fields. *IEEE Trans. Vis. Comput. Graph.*, *17*(2), 244–254.

Pedica, C., & Vilhjálmsson, H. H. (2008). Social perception and steering for online avatars. *IVA*.

Pelechano, N., & Allbeck, J. (2016). Feeling crowded yet?: crowd simulations for vr, 17–21. https://doi.org/10.1109/VHCIE.2016.7563568

Pelechano, N., Allbeck, J. M., & Badler, N. I. (2007). Controlling individual agents in high-density crowd simulation. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 99–108.

Pelechano, N., Allbeck, J. M., & Badler, N. I. (2008a). *Virtual crowds: methods, simulation, and control*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00123ED1V01Y200808CGR008

Pelechano, N., Stocker, C., Allbeck, J. M., & Badler, N. I. (2008b). Being a part of the crowd: towards validating VR crowds using presence. In L. Padgham, D. C. Parkes, J. P. Müller, & S. Parsons (Eds.), *7th international joint conference on autonomous agents and multiagent systems (AAMAS 2008), estoril, portugal, may 12-16, 2008, volume 1* (pp. 136–142). IFAAMAS. https://dl.acm.org/citation.cfm?id=1402407

Peters, C., & Ennis, C. (2009). Modeling groups of plausible virtual pedestrians. *IEEE Computer Graphics and Applications*, *29*(4), 54–63. https://doi.org/10.1109/MCG.2009.69

Qiu, F., & Hu, X. (2010). Modeling group structures in pedestrian crowd simulation. *Simulation Modelling Practice and Theory*, *18*(2), 190–205. https://doi.org/https://doi.org/10.1016/j.simpat.2009.10.005

Ren, J., Xiang, W., Xiao, Y., Yang, R., Manocha, D., & Jin, X. (2021). Heter-sim: heterogeneous multi-agent systems simulation by interactive data-driven optimization [Pre-print available online since 2018]. *IEEE Trans. Vis. Comput. Graphics*, *27*(3), 1953–1966.

Ren, Z., Charalambous, P., Bruneau, J., Peng, Q., & Pettré, J. (2017). Group Modeling: A Unified Velocity-Based Approach. *Computer Graphics Forum*, *36*(8), 45–56. https://doi.org/10.1111/cgf.12993

Reynolds, C. W. (1987). Flocks, herds and schools: a distributed behavioral model. *Proc. 14th Conf. Computer graphics and interactive techniques*, 25–34.

Reynolds, C. W. (1999). Steering behaviors for autonomous characters. *Game developers conference, 1999*, 763–782.

Rio, K., Rhea, C. K., & Warren, W. H. (2014). Follow the leader: visual control of speed in pedestrian following. *Journal of vision*, *14 2*.

Rogla, O., Pelechano, N., & Patow, G. A. (2021). Procedural crowd generation for semantically augmented virtual cities. *Computers & Graphics*, *99*, 83–99.

Rojas, F. A., & Yang, H. S. (2013). Immersive human-in-the-loop hmd evaluation of dynamic group behavior in a pedestrian crowd simulation that uses group agent-based steering. *Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, 31–40. https://doi.org/10.1145/2534329.2534336

Rojas, F. A., Yang, H. S., & Tarnogol, F. M. (2014a). Safe navigation of pedestrians in social groups in a virtual urban environment. *2014 International Conference on Cyberworlds*, 31–38. https://doi.org/10.1109/CW.2014.13

Rojas, F. A., Yang, H. S., & Tarnogol, F. M. (2014b). Safe navigation of pedestrians in social groups in a virtual urban environment. *2014 International Conference on Cyberworlds*, 31–38. https://doi.org/10.1109/CW.2014.13

Rudenko, A., Palmieri, L., Herman, M., Kitani, K. M., Gavrila, D. M., & Arras, K. O. (2020). Human motion trajectory prediction: a survey. *International Journal of Robotics Research*, *39*(8), 895–935.

Saeed, R. A., Recupero, D. R., & Remagnino, P. (2022). Modelling group dynamics for crowd simulations. *Personal and Ubiquitous Computing*. https://doi.org/10.1007/s00779-022-01687-9

Sauro, J., & Lewis, J. (2016). Standardized usability questionnaires. https://doi.org/10.1016/B978-0-12-802308-2.00008-4

Savenije, N., Geraerts, R., & Hürst, W. (2020). CrowdAR table : an AR system for real-time interactive crowd simulation. *Proc. IEEE International Conference on Artificial Intelligence and Virtual Reality*, 57–59.

Schuerman, M., Singh, S., Kapadia, M., & Faloutsos, P. (2010). Situation agents: agent-based externalized steering logic. *Computer Animation and Virtual Worlds*, *21*(3-4), 267–276.

Seffah, A., Donyaee, M., Kline, R., & Padda, H. (2006). Usability measurement and metrics: a consolidated model. *Software Quality Journal*, *14*, 159–178. https://doi.org/10.1007/s11219-006-7600-8

Shao, W., & Terzopoulos, D. (2007). Autonomous Pedestrians. *Graphical Models*, *69*, 246–274. https://doi.org/10.1016/j.gmod.2007.09.001

Shen, Y., Henry, J., Wang, H., Ho, E. S., Komura, T., & Shum, H. (2018). Data-driven crowd motion control with multi-touch gestures. *Computer Graphics Forum*, *37*(6), 382–394.

Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. *Proc. ACM National Conference*, 517–524. https://doi.org/10.1145/800186.810616

Shum, H. P. H., Komura, T., & Yamazaki, S. (2007). Simulating competitive interactions using singly captured motions. *VRST '07*.

Shum, H. P. H., Komura, T., Shiraishi, M., & Yamazaki, S. (2008). Interaction patches for multi-character animation. *ACM Trans. Graph.*, *27*(5). https://doi.org/10.1145/1409060.1409067

Sinclair, J., & Lui, C. S. M. (2015). Integrating personality and emotion for human crowd simulation. *Fifteenth International Conference on Electronic Business*.

Slater, M., Pertaub, D.-P., & Steed, A. (1999). Public speaking in virtual reality: facing an audience of avatars. *IEEE Computer Graphics and Applications*, *19*, 6–9.

Sohre, N., Mackin, C., Interrante, V., & Guy, S. J. (2017). Evaluating collision avoidance effects on discomfort in virtual environments. *2017 IEEE Virtual Humans and Crowds for Immersive Environments (VHCIE)*, 1–5. https://doi.org/10.1109/VHCIE.2017.7935624

Spirops. (2010). Vranimationspirops. https://animation.spirops.com/

Sung, M., Gleicher, M., & Chenney, S. (2004). Scalable behaviors for crowd simulation. *Computer Graphics Forum*, *23*.

Takahashi, S., Yoshida, K., Kwon, T., Lee, K. H., Lee, J., & Shin, S. Y. (2009). Spectral-Based Group Formation Control. *Computer Graphics Forum*. https://doi.org/10.1111/j.1467-8659.2009.01404.x

Thorne, M., Burke, D., & van de Panne, M. (2004). Motion doodles: an interface for sketching character motion. *ACM Trans. Graph.*, *23*(3), 424–431. https://doi.org/10.1145/1015706.1015740

Treuille, A., Cooper, S., & Popović, Z. (2006). Continuum crowds. *ACM Transactions on Graphics*, *25*(3), 1160–1168.

Tsiros, A., & Leplâtre, G. (2016). Evaluation of a sketching interface to control a concatenative synthesiser.

Ubisoft. (2016). Motion-matching in ubisoft's for honor. https://www.gameanim.com/2016/05/03/motion-matching-ubisofts-honor/

Ubisoft Toronto. (2016). Motion matching - 'dance card' breakdown. https://www.youtube.com/watch?v=_Bd2T7uP9VA

Ulicny, B., de Heras Ciechomski, P., & Thalmann, D. (2004). CrowdBrush: interactive authoring of real-time crowd scenes. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 243–252.

Unity. (2016). State machine unity. https://docs.unity3d.com/Manual/StateMachineBasics.html

Unity. (2021). Unity Documentation Navigation and Pathfinding. https://docs.unity3d.com/Manual/Navigation.html

Unity. (2022). Unityxrdoc. https://docs.unity3d.com/Manual/XR.html

van den Berg, J., Lin, M., & Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. *Proc. IEEE International Conference on Robotics and Automation*, 1928–1935.

van den Berg, J. P., Guy, S. J., Lin, M. C., & Manocha, D. (2011). Reciprocal n-body collision avoidance. *Proc. International Symposium of Robotics Research*, 3–19.

van Toll, W., Grzeskowiak, F., Gandía, A. L., Amirian, J., Berton, F., Bruneau, J., Daniel, B. C., Jovane, A., & Pettré, J. (2020). Generalized microscropic crowd simulation using costs in velocity space. *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. https://doi.org/10.1145/3384382.3384532

van Toll, W., Jaklin, N., & Geraerts, R. (2015). Towards believable crowds: a generic multi-level framework for agent navigation. *ASCI.OPEN*.

van Toll, W., & Pettré, J. (2019). Connecting global and local agent navigation via topology. In *Motion, Interaction and Games* (pp. 1–10). Association for Computing Machinery.

van Toll, W., & Pettré, J. (2021). Algorithms for microscopic crowd simulation: advancements in the 2010s. *Computer Graphics Forum*, *40*. https://doi.org/10.1111/cgf.142664

Villamil, M. B., Musse, S. R., & de Oliveira, L. P. L. (2003). A model for generating and animating groups of virtual agents. *IVA*.

Walther-Franks, B., Herrlich, M., & Malaka, R. (2011). A multi-touch system for 3d modelling and animation, 48–59. https://doi.org/10.1007/978-3-642-22571-0_5

Warren, W. H. (2018). Collective motion in human crowds. *Current Directions in Psychological Science*, *27*(4), 232–240.

Wiese, E., Israel, J. H., Meyer, A., & Bongartz, S. (2010). Investigating the learnability of immersive free-hand sketching. *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, 135–142.

Wiggins, J. S. (1996). The five-factor model of personality : theoretical perspectives. *The five-factor model of personality : theoretical perspectives*.

Wilder, D. A. (1986). Social categorization: implications for creation and reduction of intergroup bias. In L. Berkowitz (Ed.), *Advances in experimental social psychology* (pp. 291–355, Vol. 19). Academic Press. https://doi.org/https://doi.org/10.1016/S0065-2601(08)60217-8

Xi, J.-a., Zou, X.-l., Chen, Z., & Huang, J.-j. (2014). Multi-pattern of complex social pedestrian groups [The Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014), 22-24 October 2014, Delft, The Netherlands]. *Transportation Research Procedia*, *2*, 60–68. https://doi.org/https://doi.org/10.1016/j.trpro.2014.09.009

X-sens. (2000). Motion matching - 'dance card' breakdown. https://www.xsens.com/motion-capture

Xu, M., Wu, Y., Ye, Y., Farkas, I., & Jiang, H. (2014). Collective Crowd Formation Transform with Mutual Information-Based Runtime Feedback. *Computer Graphics Forum*, *34*. https://doi.org/10.1111/cgf.12459

Xu, X., Liu, W., Jin, X., & Sun, Z. (2002). Sketch-based user interface for creative tasks. *Proceedings of the 5th Asia Pacific conference on computer human interaction, Beijing*, 560–570.

Yang, F., & Peters, C. (2019). Social-aware navigation in crowds with static and dynamic groups. *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*, 1–4.

Yeh, H., Curtis, S., Patil, S., van den Berg, J., Manocha, D., & Lin, M. (2008). Composite agents. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

Yersin, B., Maïm, J., Ciechomski, P. D. H., Schertenleib, S., & Thalmann, D. (2005). Steering a virtual crowd based on a semantically augmented navigation graph. *In Proceedings of the First International Workshop on Crowd Simulation*, 169–178.

Yersin, B., Maïm, J., Pettré, J., & Thalmann, D. (2009). Crowd Patches: Populating Large-Scale Virtual Environments for Real-Time Applications. *Proceedings of the 2009 symposium on Interactive 3D graphics and games*. https://doi.org/10.1145/1507149.1507184

Yu, Q., & Terzopoulos, D. (2007). A decision network framework for the behavioral animation of virtual humans. *SCA '07*.

Yumer, M. E., & Mitra, N. J. (2016). Spectral style transfer for human motion between independent actions. *ACM Trans. Graph.*, *35*(4). https://doi.org/10.1145/2897824.2925955

Zanlungo, F., Ikeda, T., & Kanda, T. (2011). Social force model with explicit collision prediction. *EPL (Europhysics Letters)*, *93*(6), 68005.

Zhang, Y., Zhang, X., Zhang, T., & Yin, B. (2020). Crowd motion editing based on mesh deformation. *International Journal of Digital Multimedia Broadcasting*, *2020*.

Zhao, M., Turner, S. J., & Cai, W. (2013). A data-driven crowd simulation model based on clustering and classification. *Proc. IEEE/ACM 17th Int. Symp. Distributed Simulation and Real Time Applications*, 125–134.

List of Figures

1.1 Crowd crossing Shibuya Street, photo by Sei F – Wikimedia. (a) Macroscopic model: The crowd is seen as a stream in different directions (blue and orange arrows). (b) Microscopic model: same image zoomed in, each individual is simulated individually with local interactions such as collision avoidance in green and group formation in blue. . . . 24

2.1 Microscopic model processes by levels: High level is global path planning and low level is local interaction. For each frame, the global path gives the preferred velocity $\mathbf{v}_{pref}$. After searching for neighbors, local interactions are defined and applied in the form of unbreakable rules. $\mathbf{v}_{new}$ is the closest match to $\mathbf{v}_{pref}$ that complies with these rules. Applying $\mathbf{v}_{new}$ to each agent yields the new position. 32

2.2 Overview of collision-prediction concepts between two agents A_j (in orange) and A_k (in purple). $\mathbf{v}'$ is a hypothetical velocity for A_j, and $\mathbf{v}_k$ is the current observed velocity of A_k. (a) The time and distance to collision. In this example, the agents collide. (b) The time and distance to closest approach. In this example, the agents do not collide. (c) α, bearing angle that decreases over time. (d) α, bearing angle that increases over time. Based on the current speed and respective position, time to closest approach and distance of closest approach can be computed to identify whether there will be a collision (a) or not (b). Image taken from van Toll and Pettré's survey [van Toll and Pettré, 2021]. 35

2.3 Typical patterns of walking groups (from the left to the right: line-abreast, V-like, river-like pattern) [Federici et al., 2012]. 39

2.4 (a) Example of an intra-group matrix and the resulting simulated formation in [Qiu and Hu, 2010]. (b) Example scenario of groups crossing a street in [Ren et al., 2017]. Groups can split and merge back following link connection between subgroups. (c) In [Kremyzas et al., 2016], groups separate into subgroups, in this case a subleader is defined to wait for other sub-group(s). 40

2.5 (a) Example of queuing along a winding path (top) and corridor traffic that combines avoidance and following behaviors (bottom) from Lemercier et al. [2012] (b) In Warren's work [2018], a follower is attracted to either a leader's speed by a spring with stiffness k_s (left) or a leader's heading direction by spring with stiffness -k_h (right). (c) Example of an aggression proxy from [Yeh et al., 2008], as A's urgency increases, its aggression proxy, P, grows and the other agents move to avoid it, leaving a space for A to move into. 42

2.6 Reynolds [1999] models example of local interactions 43

2.7 Figure courtesy of van Toll et al. [van Toll et al., 2020]. (a) Translating a force-based navigation method to the domain. An agent experiences forces from other agents and from the goal (left). The cost $C(\mathbf{v}')$ depends on the distance between $\mathbf{v}'$ and the velocity $\mathbf{v}^*$ suggested by the forces (right). (b) Translating a typical sampling-based navigation method to the domain. Values and gradient of the cost function are visualized in light blue. 44

2.8 Overview of crowd simulation components and each component's authorable aspects. Image courtesy of Lemornari et al. [2022]. 46

2.9 Example of patches and their combination respectively. (a) [Kim et al., 2012], (b) [Yersin et al., 2009] and (c) [Jordao et al., 2015]. 51

2.10 Example of flow fields. (a) [Barnett, 2014], (b) [Treuille et al., 2006] and (c) [McIlveen et al., 2016]. 53

2.11 Example of sketched fields. (a) [Gonzalez and Maddock, 2017], (b) [Jin et al., 2008] and (c) [Patil et al., 2011]. 54

2.12 Sketch-based interface in [Ulicny et al., 2004] (a), that enables to free-handly define the positions of the agents as well as there global path. Interface in [Mathew et al., 2020] (b) and [Allen et al., 2015] (c) that enables to sketch groups motion and formation of agents. 56

2.13 Diagram of a sketch-based content creation framework. Image courtesy of Bhattecharjee and Chaudhuri [Bhattacharjee and Chaudhuri, 2020]. 58

2.14 Example of 3D pose reconstruction from 2D strokes. (a) [Davis et al., 2003], (b) [Gupta and Chaudhuri, 2018] and (c) [Hahn et al., 2015]. 59

2.15 Motion path sketch interfaces. (a) Curves define the motion of the hand and upper body of the mode [Choi et al., 2016]. (b) Red line of action defines the local motion (flap its wings and tail) and the blue one, the global path of the model (fly up and down) [Guay et al., 2015]. (c) Animation adapts to the sketch to follow the trajectory [Thorne et al., 2004]. 60

2.16 (a) shows an example of mid-air gesture to animate a steam [Arora et al., 2019], (b) and (c) illustrate the interface of Bönsh et al. [Bönsch et al., 2018], users can sketch flows using a ray interactor (b) and virtual characters follow the flow (c). 64

2.17 Screen shot of experiment around crowd in VR. (a) Study of the impact of crowd density [Dickinson et al., 2019]. (b) Comparison between virtual and real world collision avoidance [Olivier et al., 2014]. (c) Cross road experiment comparing various settings [Koilias et al., 2020]. 66

3.1 Outline of a complete simulation system with IFs. 72

3.2 (a) An IF is a vector field (shown here in blue) that prescribes velocity or orientation vectors in a domain D around a source object s (here: the red agent). (b) During the simulation, the IF is mapped onto the environment to match the current position and orientation of s. Other agents (in orange), if they are receivers, use this mapped IF to compute a velocity or orientation (in green), which they can apply directly or combine with other IF prescriptions or other navigation algorithms. Agents outside the domain D (in yellow) are not affected. 74

3.3 Parametric interaction fields based on the source speed. Example of an IF that depends on the speed of its source agent s. 75

3.4 Parametric interaction fields based on angular relation. Example of an IF that depends on the angular relation between the source s and an obstacle o. 76

4.1 IF editor overall interface, where the user can sketch an orientation field or a velocity field. The red rectangle contains the tools to sketch on canvas (in orange) a field. The user can: select guide curves to edit them (mouse icon), erase vectors (eraser icon) of the grid (blue), delete guide curves (red cross), add straight guide lines (black arrow) or free handed shaped guide curves (curved arrow). It can also link objects together (red link). Shortcuts are available so that users can more quickly draw the field by duplicating guide curves according to the desired symmetry (purple). In green, the keyframe slider enables users to navigate through all the frames of a parametric fields. 78

4.2 Concepts of the IF editor. (a) The user can draw guide curves (blue) and zero areas (red) to specify IF vectors; example vectors are shown in black. IF vectors for points in-between will be interpolated (green). (b) For any point p outside all zero areas, the IF vector is a weighted average of all vectors along all guide curves, where weights depend on the distance to p. 80

4.3 (a) Once the guide curve tool had been selected, a user can freely hand draw a guide curve of any shape: the gray line is the scribble hand free sketch and the blue lines are the corresponding guide curve. (b) Then the user modifies the shape of the curves, dragging the guide curve's handles. (c) To create a zero area, the user must select the eraser and erase the vectors which should have a amplitude of zero. 81

4.4 Examples of guide curves (shown in blue) and their resulting IFs. The gray arrows are the IF vectors (following from the interpolation scheme of Section 2) on a 20×20 sample grid. 81

4.5 To decide on the speed an agent a should have inside a field, a user can paint over the field's vectors attributing them colors. Each color corresponds to a percentage of a's maximum speed. 82

4.6 Examples of guide curves key IF sketched for different key values of a selected parameter. 83

5.1 IF framework for 3D simulation. Each block of a different color is a component of the final framework. In green, the 2D Editor to sketch IF in 2D. In purple, the 2D simulation software *UMANS* to apply IFs. The xml describing the scene is in blue. Finally, the gray block includes components built in Unity, the 3D simulator in orange, including *CrowdMP* and the Motion Matching animator. The inputs of the user are in red. 89

5.2 Motion matching compares poses according to the position (yellow sphere) and velocity (yellow arrows) of key joints, the future (red line) and past (green line) trajectories. The white lines represent the overall data set trajectories. 90

5.3 *MxM* integration into our framework. First, motion must be captured (using X-sens or others) to be processed and build up a animation library. From this library, *MxM* will choose the animation matching the positions and orientations predicted by *UMANS* dll. . 92

5.4 To build the prediction required from Motion Matching, several steps forwards are computed over Interaction Fields outputs. The future positions and orientations (transparent agents in red and green) are stored and send to *MxM* 93

5.5 Example of dance cards for *MxM* motion capture requirement. 95

6.1 Results for the *Hide and Seek* scenario (Section 1.1). (a) A velocity IF with a rotation link (red dashed segment) between the source (square in red) and a second object (orange). Guide curves are shown in grey. The red vectors indicate that the impacted agents will go at their full speed.(b) A simulation where the blue agent uses this IF to hide from the user-controlled red agent. (c) A simulation where the blue agent can hide behind all obstacles and orange agents, each emitting the same IF. (d) A 3D impression with the two main agents on the left. 98

6.2 Results for the *Several Hiders* scenario (Section 1.1). (a) A velocity IF with guide curves in grey and the field define a local minimum behind the source (in red). The vectors were colored in green so that the impacted agent would go at their medium speed. (b) A simulation where the blue agents can hide behind all obstacles and orange agents, each emitting the same IF. (c) When the blue agents are less than 1 meter away from the red agent, they are impacted by the red IF and follow the red agent. 99

6.3 Results for the *Giant* scenario (Section 1.1). (a) A velocity IF around the source (in red), the color of the vector show the relative speed the impacted agent should take (from fast in red to slow in blue). (b) A simulation where the blue agents are going slowly toward the red agent, impacted by the red IF (sketch in (a)) (c) When the red agent moves, it does not emit a IF any more and the agents are assigned to hide behind one obstacle each. 100

6.4 Results for the *VIP in a Crowd* scenario (Section 1.2). (a) Keyframes of the velocity IF used by the crowd. (b) The orientation IF used by the crowd. (c) Keyframes of the velocity IF used by the bodyguards. (d–e) Simulation examples with different speeds for the VIP (in red). The interpolated IF is shown as well. (f) Simulation example with bodyguards (in dark blue). Here, all IFs are omitted for clarity. 101

6.5 Results for the *Ambulance in a Crowd* scenario (Section 1.2).(a–c) Simulation examples with different speeds for the emergency vehicle (in red). The interpolated IF is shown as well. 102

6.6 Results for the *Cross-Road* scenario (Section 1.3). (a) Velocity Fields to make one group of agents navigate counter clockwise (green) on the crossroad. (b) Traffic velocity and orientation IF for the cars. (c) Example of a stop when a pedestrian cross, in yellow the velocity IF emitted by pedestrians and in blue the one emitted by cars. 103

6.7 Results for the *Museum* scenario (Section 1.4). (a) One of the velocity IF for walking around the central pillar. (b) The velocity IFs for all five paintings. (c) Screenshot of the simulation, also showing the parametric IFs around standing and moving agents. 104

6.8 Results for the *Moose* scenario (Section 1.5). (a–c) The simulation in 2D of the *Moose* with the previously sketched velocity parametric IF. 105

6.9 Photo of a user conducting our user study. 107

6.10 (a) Boxplots showing the median ratings, interquartile ranges, and maximum/minimum ratings (outliers excluded) of the understanding of the training (green boxplots) and the ease of completing the training (yellow boxplots) for the 5 training tasks. (b) Average ratings and ranges for the 7 tasks participants completed independently. The blue boxplots represent participants' ease of completing the task and orange their satisfaction with the final result. (c) Boxplots representing the time to complete each task. (d) The final usability scores (in percentiles) for each participant. 110

6.11 Summary of the velocity IFs that the participants drew for all 7 tasks. The source of the fields is always the red object. The black arrow in each grid cell denotes the average IF vector for that cell among the IFs of all participants. The blue intensity of a cell indicates the variety among participants: it is the standard deviation of the Euclidean distance to the average IF vector. The green curves are the trajectories induced by the participants' fields for various starting positions. The purple curves are the trajectories induced by a 'ground truth' IF drawn by the authors before the user study. This trajectory corresponds to the video instructions given to the participants. 112

7.1 The 'Scary Giant' scenario presented in Section 1.1 in VR: view of the immersed user on the left and picture of a participant testing the demo at Laval Virtual 2022. 117

7.2 Integration of VR in the IF framework. In “simulation mode”, represented in the orange block, the user can now be immersed in VR and interact directly with agents. In “sketch mode” (yellow block), the simulation loop stops and the user can sketch or edit IFs in VR. The user alternates between the two modes. 120
7.3 3D models of the controllers in VR. Those are HTC Vive models but our framework could work with many controllers (even if we recommend using Vive for consistency). . 121
7.4 An immersed user can hover over sources to display their IFs. Once an IF is selected, the user clicks on ‘sketch mode’ to start editing the IF. ‘Sketch’ mode enables to add new guiding curves by sketching them or acting them out, to modify guide curves by dragging and dropping handles, and to create zero areas by hovering over vectors when ‘eraser’ is toggled. In the top-down view, the user is represented by the white camera, the blue circles are the other agents, and the vector field of the IF being edited is in dark blue. Handles are in orange, and in yellow when selected, the preview guideline is in green, and the already sketched guide curve in light blue. 122
7.5 Selecting an IF to sketch on it. The vector field is in dark blue. 122
7.6 Example of the view appearing at any time with a grip press. 123
7.7 An example of zero area designed in VR. First the user activates the eraser function, then the user can erase vectors by pressing the main grip and hovering over the vectors (in blue). 124

LIST OF TABLES

2.1 Personality trait mapping to low level crowd simulation parameters [Sinclair and Lui, 2015]. 48

6.1 Video simulations of the scenario Section 1 106
6.2 The average ratings and standard deviations for each item of the usability questionnaire in our study. † indicates negative questions, whose score were inverted for computing the final overall score. All questions were answered on a 7-point Likert scale (from 1: Completely Disagree to 7: Completely Agree). 113
6.3 Descriptions of the five scenarios and seven evaluation tasks in the user study. Columns show the overall goal of a task (which the participants received as instructions), the agent behavior criteria used during the expert evaluation, and the resulting expert grade (averaged over all users and experts combined). 114
6.4 Videos related to the User Study and its evaluation Section 2. 115

7.1 Demonstration videos of IF in VR, presented Chapter 7 125

DOCTORAT BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Champ d'Interaction : Une Nouvelle Méthode Intuitive pour Esquisser des Comportements Collectifs.

Mot clés : Foule, RV, esquisse, intuitif, simulation de comportement, humain virtuel

Résumé : La simulation en temps réel de foules humaines a de nombreuses applications. Dans une simulation de foule typique, chaque personne ("agent") dans la foule se déplace vers un but tout en adhérant à des contraintes locales. De nombreux algorithmes existent pour des tâches locales spécifiques de "pilotage" telles que l'évitement des collisions ou le comportement de groupe. Cependant, ils ne s'étendent pas facilement à des types de comportement complètement nouveaux et ont également tendance à se concentrer uniquement sur la vitesse d'un agent sans contrôler explicitement son orientation. Cette thèse présente une nouvelle méthode basée sur des esquisses pour modéliser et simuler de nombreuses interactions pour les agents dans une foule. Le concept de champ d'interaction (CI) est central : un champ vectoriel qui décrit les vitesses ou les orientations que les agents doivent utiliser autour d'un agent ou d'un obstacle "source" donné. Un champ d'interaction peut aussi changer dynamiquement en fonction de paramètres, comme la vitesse de marche de l'agent source. Les CIs peuvent être facilement combinés avec d'autres aspects de la simulation de foule, tels que l'évitement des collisions. Après l'état de l'art, ce manuscrit détaille le concept théorique et pratique associé aux CIs. Nous présentons également un outil intuitif qui calcule un CI à partir de croquis d'entrée. Nous illustrons ensuite les capacités des CIs avec plusieurs scénarios interactifs et nous évaluation notre interface graphique avec une étude utilisateur. Un dernier chapitre explore ensuite les possibilités d'étendre l'approche à la Réalité Virtuelle et présente une une preuve de concept. Globalement, notre nouvelle méthode permet d'enrichir facilement toute simulation de foule basée sur des agents avec de nouvelles interactions entre agents.

Title: Interaction Field: a New Intuitive Method to Sketch Collective Behaviors.

Keywords: Crowd, VR, sketch-based, intuitive, simulated behaviors, virtual human

Abstract: The real-time simulation of human crowds has many applications. In a typical crowd simulation, each person ('agent') in the crowd moves towards a goal while adhering to local constraints. Many algorithms exist for specific local 'steering' tasks such as collision avoidance or group behavior. However, these do not easily extend to completely new types of behaviors, such as circling around another agent or hiding behind an obstacle. They also tend to focus purely on an agent's velocity without explicitly controlling its orientation. This thesis presents a novel sketch-based method for modelling and simulating many steering behaviors for agents in a crowd. Central to this is the concept of an *Interaction Field* (IF): a vector field that describes the velocities or orientations that agents should use around a given 'source' agent or obstacle. An IF can also change dynamically according to parameters, such as the walking speed of the source agent. IFs can be easily combined with other aspects of crowd simulation, such as collision avoidance. Following a review of the state of the art, this manuscript details the theoretical and practical concept associated with IF. We then demonstrate the capabilities of IF by illustrating several scenarios, as well as through a user evaluation of the principle. A final chapter then explores the possibilities of extending the approach in VR and present a proof of concept. Overall, our novel method enables non-expert users to easily enrich any agent-based crowd simulation with new agent interactions.