Rapport de documentation de stage

Raoelisolonarivony - MISA M2

19 Février 2017

Table des matières

1 New generation crowd simulation algorithms (2014) of Julien Pettre and Ming Lin. (suite)

1.1 Interactive Modeling, Simulation and Control of Large-Scale Crowds and Traffic of Ming C. Lin

Cet article fait un recensement et une évaluation des travaux de Ming Lin et al. sur la simulation de larges foules (agents). Les groupes d'entités (individu ou véhicule) forment des systèmes complexes dévoilant des patterns observables dans la société. Les caractéristiques qui émergent de ces comportements sont encore à découvrir. Ils décrivent leur compréhension des principes des foules macroscopiques et identifient des modèles issus de leur comportement. Ils énoncent des méthodes de navigation globale d'entités virtuelles multiples tout en évitant les collisions au niveau local : une formulation modélisant la dynamique d'une large foule, une méthode de simulation d'un traffic continu, le contrôle d'une foule par des champs de navigation.

1.1.1 Clear path

"Clear path" est une approche qui généralise les méthodes basées sur la vélocité (VO : Velocity Obstacle) avec une simulation multi-agent. Les calculs pour les algorithmes d'évitement de collisions locales peuvent être réduit à l'optimisation d'un problème à complexité quadratique minimisant les variations de la vélocité de chaque agent soumis à des contraintes de non-collision. Un algorithme en temps polynomial pour les agents en ressort permettant de calculer le mouvement 2D sans collision d'une manière distribuée. En supposant que le problème de d'évitement de collision locale pour N agents est un problème d'optimisation combinatoire, ils étendent la formulation de la VO en imposant des contraintes additionnelles qui garantissent l'évitement de collision pour chaque agent pendant un interval discret. Le Fast Velocity Obstacle (FVO) utilise quatre contraintes. Les *deux* premières contraintes de frontières coniques de la FVO sont les mêmes que celles de la RVO :

$$FVO_{L}{}^{A}_{B}(\mathbf{v}) = \phi(\mathbf{v}, (\mathbf{v}_A + \mathbf{v}_B)/2, \mathbf{P}^{\perp}_{ABleft}) \geq 0 \tag{1}$$

$$FVO_{R}{}^{A}_{B}(\mathbf{v}) = \phi(\mathbf{v}, (\mathbf{v}_A + \mathbf{v}_B)/2, \mathbf{P}^{\perp}_{ABright}) \geq 0 \tag{2}$$

Deux contraintes supplémentaires sont introduites dans le modèle ClearPath.

Contrainte d'intervalle de temps finie. Les collisions sont évitées si elles surviennents dans un intervalle de temps défini ΔT. Le cône d'intervalle de la RVO est tronquée pour se limiter à un laps de temps défini.

Contrainte de l'orientation de vélocité. Les agents sont contraints de se déplacer sur leur côté droit par la formulation $FVO_{C}{}^{A}_{B}(\mathbf{v}) = \{v \,|\, \phi(\mathbf{v}, P_A, \mathbf{P}^{\perp}_{AB}) \leq 0\}$.

Toutes ses contraintes garantissent un évitement de collision. Pour chaque agent, le problème est résolu par l'optimisation d'une fonction avec des contraintes linéaires. Ce qui est possible en temps polynomial si le nombre de contraintes est limité (sinon c'est NP-difficile), deux dans le cas de ClearPath.

P-ClearPath est une extension de "ClearPath" qui parallélise l'algorithme. Des opérations de groupement d'agents pour le traitement de collision et de division des traitements permettent d'améliorer l'algorithme.

Résultats : P-ClearPath est un algorithme très efficace et ils l'ont prouvé en le testant sur des

machines à coeur multiples (Larabee avec 32-64 cores) et prends 2.5 millisecondes dans des scénarios composés de centaines d'agents.

1.1.2 Foules hybrides

Les foules denses sont caractérisées par une distance interpersonnelle faible et en conséquence une perte de liberté de mouvement. Ils ont choisi alors de modéliser le mouvement de ces encombrantes foules comme le flux effectué par un seul système. Pour cela, un modèle d'évitement de collision entre agents est développé pour alléger et séparer cette opération du guidage globale de la foule d'agents interactivement.

La méthode combine une représentation lagrangienne des individus avec un model de foule eulérien. Le but étant d'obtenir les comportements locaux entre agents et le mouvement global de la foule en général. Le défi est de simuler la variation observée de la structure de la foule en fonction de l'environnement qui devient plus étroit ou confiné : la foule initialement sous la forme d'un flux compressible avec une densité faible devient incompressible quand les agents se rapprochent de plus en plus les uns des autres. Ils proposent une nouvelle formulation mathématique de ce phénomène.

Résultats : Ils peuvent simuler une dense foule de plus de 100 000 agents compressés dans des environnements étroits (mosquées, marchés, conférences). Les agents peuvent donc avoir un objectif unique sans que l'algorithme ne produise d'effets indésirés.

1.1.3 Le traffic continu

Ils constatent que la simulation de traffic a été largement simulée soit par des approches microscopiques ou macroscopiques. Cependant, peu d'attention a été accordée à la production d'animations 3D basées sur les modèles macroscopiques. Ils ont alors simulé à grande échelle, les traffics sur des réseaux routiers réels avec ces modèles. Cependant, des informations au niveau microscopique sont utilisées pour la représentation visuelle de la simulation.

Leur approche est une extension des approches par file continue auxquelles ils procurent un nouveau modèle de changement de file utilisant une représentation discrète des véhicules. La méthode proposée a été validée par des observations du traffic réel.

Résultats : Ils ont testé leur technique sur des routes synthétiques (virtuelles) dans des villes à taille moyenne avec un grand volume de traffic sur un court segment d'autoroute. Des données sur les routes (routes secondaires, les feux tricolores, les grands axes) ont été obtenues depuis la base de données GIS américaines disponibles au public dénommée TIGER. La visualisation similaire au système de particules reflète des mouvements réalistes. Les calculs effectués sont directement proportionnels au nombre de cellules à mettre jour à chaque étape de la simulation, même une machine personnelle Intel Core2 produit des résultats satisfaisants (23 kilomètre-carrés, 180 kilomètres de routes, 2500 véhicules).

1.1.4 Diriger les foules avec des champs de navigation

Dans des systèmes basés agent, chaque agent est supposé être une entité prenant des décisions indépendantes. D'autres méthodes se concentrent sur les comportements de groupes pour prendre des décisions. Ces modèles produisent des effets indésirables pour les comportements macroscopiques. Le but étant de générer des mouvements de foules plausibles tout en respectant les règles locales de non-collision et d'interaction.

Ils développent une approche pour guider la simulation de flux d'agents, lequels possèdent des

connaissances de leur environnement et ont des objectifs à chaque étape de la simulation. Ils utilisent des champs de guidage discrétisés. Ces champs peuvent être modifié par l'utilisateur interactivement. Les comportements locaux sont le fruit des méthodes existantes pour l'évitement de la collision, l'espace personnel, la communication entre agents.

Résultats : Cette approche est applicable aux méthodes basées agent existantes. Ils démontrent l'utilité de cette approche par la simulation de scénarios basés sur des séquences vidéos du flux de personnes réelles. L'utiliseur peut interagir avec le champ de guidage.

1.1.5 Discussions

ClearPath est convenable pour la simulation d'interaction de foule très dense et a besoin des avancés en matière de parallélisation. Son implémentation reste à explorer.

La technique de modélisation 3D de traffic par la méthode macroscopique peut simuler des conditions variées d'interactions. Ils veulent creuser plus le couplage entre les méthodes macroscopiques qui permettent de générer une grande quantité de véhicules avec les modèles discrets qui décrivent les comportements locaux des agents. La perspective d'intégrer un traffic et des simulations de foule pour modéliser une scène urbaine est évoquée.

L'approche pour guider les foules avec des champs de navigation est général et peut intégrer tout algorithme d'évitement de collision. L'interaction de l'utilisateur peut se faire par dessin ou suivant des séquences vidéos réelles.

2 Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation (2008) of Jur van den Berg et al.

Dans cette article, les auteurs proposent un nouveau concept - le RVO (Reciprocal Velocity Obstacle) - pour la navigation multi-agent en temps réel. Ils considèrent le cas dans lequel chaque agent se déplace indépendamment sans communication explicite avec les autres agents.

D'abord, ils donnent un bref apercu des travaux antérieurs sur la question. Certaines méthodes ne prennent pas en compte le comportement des autres agents, ou les considèrent simplement comme statiques. Des risques de collision sont observés pour ces méthodes, par exemple pour les vitesses élevées. Ils citent également des méthodes antérieures basées sur le concept de "Velocity Obstacle" incorporant des comportements réactifs mais générant des oscillations dans les trajectoires. Ils notent des approches créées pour gérer ces oscillations mais qui ne sont pas clarifiées comme le "Common Velocity Obstacle".

Etant donné que leur formulation est une extension du concept "Velocity Obstacle", ils le reprennent.

Velocity Obstacle. Pour un agent A (position : p_A, vélocité : v_A) en mouvement dans un plan et un obstacle B (p_B, v_B), le "Velocity Obstacle" de l'obstacle B pour l'agent A noté $VO^A_B(v_B)$ est un *ensemble constitué des vélocités v_A de A qui conduisent à une collision avec l'obstacle B à un moment donné.* Pour la formulation mathématique, ils notent la somme de Minkowski $A \oplus B = \{a + b \,|\, a \in A, b \in B\}$ des objets A et B avec $-A = \{-a \,|\, a \in A\}$ la négation de l'objet A ainsi que $\lambda(p, v) = \{p + tv \,|\, t \geq 0\}$ représentant les rayons partant de p dans la direction de v. Si un rayon partant de p_A suivant la direction de la vélocité relative de A et B, $(v_A - v_B)$, intersecte la somme $B \oplus -A$ centré en p_B alors la vélocité v_A appartient au "Velocity Obstacle" de B. Il s'en suit que le "Velocity Obstacle" de B pour A est définie par :

$$VO^A_B(v_B) = \{v_A \text{ tel que } \lambda(p_A, v_A - v_B) \bigcap B \oplus -A \neq \emptyset\}$$

Si v_A appartient à $VO_B^A(v_B)$, A et B entreront en collision. Si v_A se trouve en dehors de cet ensemble, les deux agents ne seront jamais en collision. Enfin, si v_A se trouve à la frontière de $VO_B^A(v_B)$, A va toucher B. Le Velocity Obstacle de B pour A forme une zone conique dont le sommet est pointé par V_B. Pour plannifier le déplacement d'un agent en suivant ce concept, un choix de vélocité est effectué parmi celles qui sont à l'extérieur du "Velocity Obstacle" à chaque cycle. Si la vélocité choisie figure parmi celles qui conduisent à l'objectif de l'agent, ce dernier peut suivre son chemin sans encombre. Ils ajoutent des lemmes pour définir les propriétés des "Velocity Obstacles" comme la symmétricité, l'invariance par translation et la convexité. Ils introduisent notamment le concept de régions admissibles à droite et à gauche définie par les demi-plans à gauche et à droite du domaine $VO_B^A(v_B)$.

Ensuite, ils montrent que le "Velocity Obstacle" génère des oscillations quand il est utilisé pour la navigation multi-agent. Soient deux agents (A, v_A) et (B, v_B) en mouvement tels que $v_A \in VO_B^A(v_B)$ et $v_B \in VO_A^B(v_A)$, ils vont entrer en collision donc ils choisissent chacun de nouvelles vélocités v'_A et v'_B en dehors des obstacles de vélocité respectifs. Dans cette nouvelle situation, les vélocités antérieures se trouvent à l'extérieur des nouveaux obstacles de vélocité, en plus d'être des candidates idéales vu qu'elles conduisent aux objectifs. Donc, elles sont resélectionnées en vertu de la symmétricité des Velocity Obstacles. De nouveau, les agents doivent éviter la collision entre eux et modifier les vélocités et ainsi de suite. D'où les oscillations dans les trajectoires des agents.

Reciprocal Velocity Obstacles (RVO). Ils présentent ensuite leur nouveau concept, le RVO et montrent qu'il génère des déplacements sûrs et sans oscillations pour chacun des agents. Leur approche prend en compte le comportement de réaction des autres agents en supposant implicitement que les autres agents effectuent un raisonnement d'évitement de collision similaire. Leur idée est qu'au lieu de choisir une vélocité en dehors de l'obstacle de vélocité, il faut en choisir une qui est la moyenne entre la vélocité actuelle et celle à choisir. Le Reciprocal Velocity Obstacle d'un agent B pour un agent A est formalisé par la définition suivante :

$$RVO_B^A(v_B, v_A) = \{v'_A \text{ tel que } 2\,v'_A - v_A \in VO_B^A(v_B)\}$$

Géométriquement, le sommet du cône du domaine de ce RVO est indiqué par la vélocité $\frac{v_A+v_B}{2}$. Ils démontrent en se basant sur les définitions et les lemmes la propriété de non-collision de cette méthode. En outre, quand les agents à risque de collision choisissent des vélocités en dehors de leurs RVO respectifs, leur chemin reste sur le même côté. De plus pour l'agent A, la vélocité la plus proche de v_A en dehors de la RVO de l'agent B est $v_A + u$, réciproquement pour l'agent B la vélocité la plus proche de v_B en dehors de la RVO de A est égale à $v_B - u$. Ils prouvent ensuite le caractère non oscillatoire de la méthode de la RVO par le théorème suivant :

$$v_A \in RVO_B^A(v_B, v_A) \Leftrightarrow v_A \in RVO_B^A(v_B - u, v_A + u)$$

Ce théorème montre que les vélocités initiales v_A et v_B se trouvent dans les RVO correspondants aux nouvelles vélocités $v_A + u$ et $v_B - u$. Elles ne sont pas susceptibles d'être resélectionnées pour cette méthode étant donné que les vélocités les plus proches restent les nouvelles vélocités et ainsi de suite. De par la convexité du domaine de la RVO, ils proposent une généralisation en ne se basant pas sur la moyenne mais sur un paramètre α_i^j définissant la proportion de participation de chaque agent pour l'évitement de la collision. La définition générale est la suivante :

$$RVO_B^A(v_B, v_A, \alpha_B^A) = \{v'_A \text{ tel que } \frac{1}{\alpha_B^A}\,v'_A + (1 - \frac{1}{\alpha_B^A})\,v_A \in VO_B^A(v_B)\}$$

Navigation multi-agent. Ils décrivent comment ils utilisent cette méthode pour diriger plusieurs agents dans des environnements complexes contenant des obstacles statiques et en mouvement. Ils combinent les RVO de tous les agents en faisant l'union avec les "Velocity Obstacles" des obstacles dans l'ensemble $\mathcal{O}$:

$$RVO^i = \bigcup_{i \neq j} RVO_j^i(v_j, v_i, \alpha_j^i) \;\cup\; \bigcup_{o \in \mathcal{O}} VO_o^i(v_o)$$

Ils définissent les contraintes de vitesse pour chaque agent par un ensemble de vélocités admissibles, en considérant les vitesses maximales v_i^{max} et les accélérations maximales a_i^{max} et le domaine des vélocités admissibles :

$$AV^i(v_i) = \{v_i' \text{ tel que } \|v_i'\| < v_i^{max} \text{ et } \|v_i' - v_i\| < a_i^{max}\,\Delta t\}$$

Concernant la sélection des vélocités, pour chaque agent A_i, une vélocité de préférence v_i^{pref} : la vélocité dont la norme est égale à la vitesse convenable à l'agent et sa direction pointe vers l'objectif. Si l'agent est proche de l'objectif, le vecteur nul est assigné à v_i^{pref}. Consécutivement, une nouvelle vélocité v_i' est choisie comme la plus proche de v_i^{pref} et se trouvant en dehors du RVO^i tout en appartenant au domaine AV^i. Il arrive que, dans un environnement encombré, les RVO combinés remplissent les vélocités admissibles. Dans ce cas, l'algorithme choisit une vélocité à l'intérieur de la RVO mais cette vélocité est pénalisé (sous forme d'une fonction de pénalité) par rapport à la vélocité de préférence et du time-to-collision. Les vélocités avec les pénalités minimales sont plus susceptibles d'être choisies :

$$v_i' = arg\min_{v_i'' \in\, AV^i} penalty_{\,i}(v_i'')$$

Un échantillon au nombre de N des vélocités du domaine AV^i est utilisé pour ce choix.
Enfin ils définissent une région de voisinage pour séléctionner les agents à prendre en compte pour la définition de la RVO^i. Cette région NR^i est située par rapport à la position de l'agent A_i et sa taille est proportionelle à la vitesse moyenne des agents et des obstacles.

Résultats et expériences. Ils appliquent leur concept de navigation sur des centaines d'agents dans des environnements densément peuplés d'obstacles statiques et en mouvements. Parmi les expériences, des agents forment la circonférence d'un cercle et se déplacent vers la position opposée comme objectif. Une autre forme consiste à faire interagir des agents dans un espace étroit en plaçant des obstacles au milieu d'une pièce. Une dernière catégorie d'expérience consiste à mettre une file d'agent en mouvement en face d'un obstacle (auto) avançant vers eux passivement sans prendre en compte leurs actions.
Pour finir, ils montrent que la performance en temps réel est achevée dans ces sénarios complexes par rapport au nombre d'agents de la simulation.

3 Mécanismes évolutionnistes pour la simulation comportementale d'acteurs virtuels (2004) de Stéphane Sanchez

Dans cette article, Stéphane Sanchez propose une méthode de génération de comportements pour les personnages virtuels à l'aide de systèmes de classeurs (une méthode d'apprentissage évolutionniste). Les générateurs de comportements servent à rendre les personnages plus autonomes : effectuer une tâche définie en fonction de son environnement sans la participation de l'utilisateur.

Les méthodes existantes reposent sur des scripts (scénarii) ou sur des automates. Associées à des systèmes d'animation, les personnages virtuels résultants de ces méthodes effectuent des séquences d'actions complexes convaincantes. Les limites de ces méthodes sont observées quand le personnage se trouve face à des situations imprévues par les scripts ou les automates : il ne peut pas accomplir sa tâche.

Sanchez propose de donner au personnage virtuel la capacité d'apprendre à résoudre ces situations indéfinies en se basant sur ses connaissances et la perception de son environnement. Généralement, les humains virtuels disposent déjà de comportements élémentaires (déplacement, saisir un objet, ...), l'objectif est de mettre à profit ses comportements en les combinant.
Etant donné que l'environnement dans lequel évolue le personnage est dynamique, le choix des systèmes de classeur est justifié car ce sont des systèmes évolutifs. Pour cela, l'utilisateur contribue à l'initialisation du système en fournissant les tâches à résoudre et les situations clés pour définir des récompenses utiles

à l'apprentissage du personnage. Les actions effectuées par ce personnage résultent d'une déduction à partir des informations reçues et des récompenses perçues.

Les systèmes de classeurs sont à bases de règles, introduits par John Holland, mettant en jeu les algorithmes génétiques pour identifier et classer des règles par ordre de force. Les règles les plus fortes pour une situation donnée (théorie de Darwin) permettent au système de survivre dans son environnement. L'apprentissage des règles ne requiert pas une connaissance totale de l'environnement, le système a pour but de maximiser ses récompenses. Les règles sont représentés par une série de caractères respectant le masque suivant :

$$\underbrace{011}_{condition} :: \underbrace{1011}_{acion}\ \underbrace{0.5}_{force}$$

Le terme *condition* représente l'état actuel de l'environnement (une chaîne de caractères pris dans l'alphabet $\{0, 1, \#\}$ de longueur fixe, # peut être 0 ou 1). Le terme *action* est l'action à entreprendre (les caractères sont pris dans $\{0, 1\}^2$). Le dernier terme "*force*" est un scalaire reflétant la performance du système.

Le système de classeurs fonctionne avec trois composantes : le capteur, le classifieur (muni d'un algorithme génétique) et l'effecteur. A partir des données de l'environnement, le capteur transcrit le message (appelé *situation*) et le stocke dans une liste. Ce message est transmis au système de classeurs qui recherche les classeurs à activer tels que les conditions correspondent au message transmis. L'action avec la plus grande force est séléctionnée et transmise aux effecteurs. L'environnement assigne une note à cette action en fonction de sa performance. L'algorithme génétique agit sur la liste des classeurs en produisant de nouvelles règles et en remplaçant les règles moins performantes.
Les systèmes de classeurs LCS [1] ont été améliorés par des variantes comme le ZCS (Zeroth Level Classifiers) ou le XCS (eXtended Classifier Systems) : le premier en éliminant la boucle interne sur la liste des messages et en introduisant le "covering" pour générer des règles en fonction des situations, le deuxième en utilisant la présicion de la prédiction de récompense pour limiter l'exploration de nouvelles solutions en faisant dominer les classeurs "forts".

Sanchez applique ensuite le système de classeur dans la simulation comportementale. En premier lieu, Pour justifier cette stratégie, il affirme que ce système satisfait aux contraintes de conception de systèmes d'agents cognitifs (en autres, les connaissances de l'agent sont représentées par les règles des classeurs, le système implémente une boucle perception-décision-action, un stimulus de l'environnement déclenche le mécanisme, etc.). En second lieu, il indique que le système est utilisable dans différentes environnements tout en réduisant la taille de la base de règles, prennant avantage de l'adaptibilité des LCS. En troisième lieu, les systèmes de classeurs peuvent être installés dans des environnements complexes (dynamiques, requérant une génération continue d'actions nouvelles, avec des buts, des informations partiels, etc.). Enfin, l'intégration des systèmes de classeurs dans des architectures de simulation est facilitée par sa structure compacte (capteurs, base de connaissance, effecteurs). Il fait remarquer que l'utilisation des systèmes de classeurs dans la simulation de personnage virtuelle a été orientée sur la génération de comportements collectifs (ex. simulation de partie de basketball).

Sanchez décrit ensuite une génération de comportements individuels à travers le système ViBes qui simule le comportement d'un agent en combinant des actions élémentaires formant des actions complexes afin d'atteindre un objectif. Ce système sert de base de simulation pour introduire les systèmes de classeurs dans l'animation comportementale. Le système ViBes est composé principalement d'un système de perception, d'un gestionnaire de connaissances, d'un séquenceur d'instruction (il assure le dialogue entre l'interface entrante et le système comportemental), d'un système décisionnel (un réseau hierarchisé de modules comportementaux, à un module correspond une tâche utilisateur). Le système de perception capte les informations de l'environnement avec les instructions de l'utilisateur et les transmet au séquenceur qui décompose la tâche en directives élémentaires moins complexes

1. Learning Classifier Systems

(sous-modules). Ces sous-modules sont combinés et mis en séquence pour définir l'action à transmettre aux effecteurs.

Sanchez détaille ensuite son idée de faire l'apprentissage du personnage grâce à un système de classeurs. Le système permet de combiner et sélectionner l'action la plus cohérente pour effectuer une tâche. Il décrit le paramétrage nécessaire pour le système : la forme de la condition en tant que vecteur *situation*, la forme d'une *action* comme un vecteur indexé d'actions, la politique de compensation (le classeur activant une instruction irréalisable est pénalisé, tandis que le classeur activant la dernière instruction conduisant à l'objectif obtient la récompense maximale). Il indique que le système de classeur est intégré dans le système ViBes sous forme de module (les filtres perceptifs sont les capteurs, le système de classeur se trouve dans le solveur du module en évaluant l'action trouvée, le gestionnaire de comportement gère les récompenses).

Comme résultat, il relate la mise en place du module dans le système ViBes. Le système de classeur du module choisi est GHXCS (Generic Heterogeneous Classifier Systems) qui permet d'éviter la transcription des messages dans l'alphabet ternaire et homogénéise l'ensemble du système par des vecteurs typés (bits, trits, entiers, ...). Il illustre ces résultats par le succès de l'apprentissage d'un personnage virtuel affamé qui interagit avec les appareils éléctroménagers et la cuisine pour manger une pomme.

4 Reconstruction de scènes urbaines à l'aide de fusion de données de type GPS SIG et Vidéo (2007) de Gael Sourimant

Cette thèse de doctorat concerne la reconstruction 3D de scènes urbaines en combinant des données SIG et des images vidéos prises au sol conjointement avec des informations de positionnement GPS.

Gael Sourimant signale l'importance des plates-formes tels que Google Earth actuellement. Il pointe également les lacunes de ces systèmes au niveau des textures et de la précision de modélisation des bâtiments urbains, parfois dénués de fenêtres et de portes (les micro-structures).

4.1 Reconstruction 3D

Une première partie développe les méthodes de reconstruction 3D et les données utilisées.
Définir le type de reconstruction 3D permet de préciser les données nécessaires.

Reconstruction à partir de données aériennes. Un avion ou des satellites permettent d'acquérir des images via le radar par exemple. La zone de couverture de ces méthodes est large et ils ont l'avantage d'avoir des coûts faibles. En revanche, ils manquent de précision quant aux structures et formes des bâtiments.

La reconstruction nécessite deux étapes : déterminer les zones occupées par les bâtiments et calculer leur hauteur.

Pour *distinguer les bâtiments*, un premier modèle repose sur la segmentation de la zone en plus petites régions et d'identifier une similarité de ces régions à partir de critères d'homogénéité. Des données d'élévation DEM peuvent être utilisées pour la détermination des régions. Par les réseaux de neurones, ils peuvent être classifiés.
Un deuxième modèle projette les empreintes des bâtiments sous forme de SIG dans des photos aériennes obliques. Il faut ensuite une mise en correspondance entre les éléments de deux images d'une même zone pour trouver le contour des bâtiments.
Un dernier modèle consiste à détecter des formes rectangulaires des bâtiments à partir de la projection

de lignes extraites d'images radar sur celles d'une image optique classique, la détection de Canny-Deriche est cité comme exemple. Une autre alternative est de considérer les bâtiments comme un ensemble de coins à déterminer. Des segments extraits de l'image radar sont utilisés car les points d'intersections de ces segments servent à déterminer les coins composants les bâtiments.

Pour *calculer la hauteur* des bâtiments, les régions des images segmentées antérieurement permettent de déterminer les polygones des toits. En appliquant la stéréovision, les hauteurs du modèle 3D sont déterminées. Une autre méthode implique la mise en correspondance des données extraites de l'image avec des modèles synthétiques (sous forme CSG : Constructive Solid Geometry) et permet d'avoir des formes de bâtiment complexes.

Reconstruction 3D à partir de données au sol. Les coordonnées, la texture, les formes et éléments des bâtiments sont acquises par des photographies. L'acquisition au sol est précise mais nécessite des moyens considérables.
Il y a deux méthodes de reconstruction : celle reposant uniquement sur les images, ensuite celle qui utilise des modèles pour faciliter la reconstruction.
Une caméra fournit les données sur les images capturées pour l'*approche sans à priori*. La pose, le calibrage et les prises de vue de la caméra sont enregistrés. Un algorithme distingue les façades des fausses facades et détermine les textures des bâtiments. Les structures des villes loin d'être polyédriques sont plus complexes. Tsy modélise la surface des villes par des ondelettes.
Pour la *reconstruction basée modèles*, le logiciel *Façade* qui fait intervernir une personne au processus de placement des primitives, de définition de contraintes du modèle sur la scène. C'est une combinaison de conception assisté par ordinateur et de stéréovision. Un algorithme rassemble les bords du modèle. Les images de la caméra étant projeté sur le modèle choisi par l'utilisateur.
Une autre méthode plus incrémentale déduit un nuage de point de la scène à partir de plusieurs images. Ces points déterminent les plans principaux du bâtiment. Les portes et fenêtres sont ensuite détectées comme des régions d'intérêt par l'observation de la variation de gradient par rapport à la façade. Ces nuages de points permettent aussi de raffiner les zones de micro-structure en donnant des informations sur leur profondeur.
Il est également possible d'établir les plans principaux des bâtiments par l'estimation des points de fuite. Le raffinage du modèle est le fruit le résultat d'un appariement avec des modèles 3D prédéfinis de portes et fenêtres.
La méthode de Torr et al. représente la scène par des couches, détectées en estimant les nuages de points de la scène en 3D. Le plan principale de la scène en est une, les micro-structures sont des couches superposées. Une couche est constituée par les propriétés de positionnement, la taille, la texture, etc. Elle est associée à un modèle prédéfinies (arche, fenêtre rectangulaire) et regroupée par des hyperparamètres (propriété d'alignement par exemple).

Reconstruction à partir de données hybrides.

Pour profiter des forces des deux approches de reconstruction et avoir des modèles 3D précis à grande échelle. Des méthodes hybrides combinent à la fois les données aériennes et les données au sol. Un premier modèle se base sur les images aériennes pour en dégager les empreintes au sol des bâtiments, avec une intervention manuelle. Des données Lidar permettent ensuite de reconstruire les bâtiments. Des images au sol servent ensuite à dfinir les textures. La fusion d'images aériennes avec les données au sol et Lidar ne donnent pas de géométries précises pour les façades.
Une autre méthode modélise les scènes urbaines à partir de données acquises au sol par deux lasers et une caméra. Les données brutes sont acquises et une estimation de pose de la caméra est effectuée, les textures des façades sont plaquées automatiquement.

4.2 Données utilisées

Sourimant introduit son idée de combiner des données aériennes fournies par l'Institut Géographique National (IGN) et des données acquises au sol avec des caméras. Pour cela, il utilise les mesures GPS conjointement à l'acquisition des images au sol. Il présente ces trois types de données en détails pour en dégager les spécificités et limitations.

Un **Sig** (Système d'Information Géographique) est un système donnant une représentation d'un environnement géographique à partir de primitives simples. Les informations fournies par le SIG dépendent de la nature (hydrographie, routes, bâtiments) de l'environnement et sont regroupées par couches. Evidemment, la couche qui nous intéresse est celle des bâtiments. Elle est composée par une liste de bâtiments sous forme de points et représentée par le triplet [empreinte au sol, altitute, hauteur]. Dans le repère spécifique UTM (Universal Transverse Mercator), le bâtiment est décrit par le triplet [liste de points 2D dans le plan $(X, Y)_{utm}$, altitude a_{utm}, hauteur h_{utm}]. La représentation GIS a des limites : aucun détail géométrique pour les micro-structures, une représentation strictement plane, pas d'information sur la texture. L'esthétique des bâtiments n'est pas modélisée. En conséquence, il est nécessaire d'introduire les images de la vidéo. Elles servent à extraire les textures et les micro-structures par recalage, ces images étant projetées sur le modèle choisi.

Vidéos Les images vidéos sont acquises au sol grâce à une caméra numérique. Sourimant suppose le pire des cas où la caméra n'est pas calibrée et que seules les propriétés données par le constructeur de l'appareil sont disponibles, et ainsi rendre robuste la reconstruction.

GPS Pour mettre en relation les données SIG et les images des vidéos, une information de GPS est ajoutée aux images au sol qui sont acquises conjointement. Le GPS est un système de navigation américain par satellite consistué de 24 satellites gravitant autour de la planète sur 6 orbites circulaires. Les informations requises pendant leur passage permettent de repérer tout point du globe. Cinq stations au sol inséminées sur la terre sont chargées de le piloter. Ces satellites émettent sur deux fréquences pour minimiser les erreurs de localisation dûes aux interférences électriques sur le signal envoyé depuis les satellites lors de la traversée des couches protectrices terrestres.
L'auteur détaille ensuite la manière dont le système GPS opère pour repérer un point sur la planète. Notamment, il note l'intersection des sphères satellitaires qui doivent être au nombre de quatre pour fournir des données d'une grande précision en 3D. Il indique que les coordonnées doivent être polaire (latitude, longitude, altitude) en rappelant les règles de transformation entre les coordonnées. Les sources d'erreur du GPS sont les perturbations atmosphériques, les trajets multiples occasionnés par les environnements urbains par réflexion des signaux et la configuration des satellites (évaluée par la DOP : Dilution of Precision).
Pour corriger ces erreurs, les USA introduisent les GPS différentiel qui fait appel à des stations au sol pour réduire les erreurs en effectuant des mesures à partir de points éloignés les uns des autres. L'autre système proposé EGNOS (European Geostationary Navigation Overlay System) joue le même rôle que le GPS différentiel à la différence que EGNOS utilise trois satellites géo-stationaires pour relayer les informations de corrections au lieu de stations au sol. Les récepteurs GPS récents sont munis d'un canal supplémentaire pour les corrections transmises par EGNOS, en plus du canal captant le GPS normal.

5 Procedural city layout generation based on urban land use models (2009) of S. A. Groenewegen, R. M. Smelik , K. J. de Kraker and R. Bidarra

L'article décrit une méthode procédurale pour créer les tracés (layouts) en 2D de villes accessible à des utilisateurs non techniques qui fournissent des paramètres simples (taille de la ville, situation

géographique et information historique). Le résultat est représenté par des districts de la ville disposés suivant des contraintes issues de l'aménagement du territoire (urban land use). C'est une génération rapide.

Ce qui manque aux villes générées procéduralement est le réalisme des structures. Ces effets nécessitent une expertise de la part de l'utiliateur et des données externes comme la carte de la densité de la population ou la structure d'une ville existante (cas du logiciel CityEngine de Parish et Muller). Les villes ne sont alors pas générées par la méthode de CityEngine ou celle pour CityGen (Kelly et McCabe) mais utilise des données statistiques. L'approche basée sur l'agent comme pour City-Builder se remet à plusieurs agents (spécialistes en urbanisme) qui génèrent le tracé d'une petite ville d'une surface limitée en 15 minutes. L'objectif de leur méthode est de permettre aux non techniciens de générer rapidement des villes réalistes sans besoin de données externes.

5.1 Les modèles d'aménagement de territoire urbain

Retrouver des patterns dans les structures urbaines est essentiel en géographie urbaine. Dans les villes modernes, les districts sont distingués par des caractéristiques résidentielles et sociales (industrie, commerce, zones résidentielles "high-class" et "low-class"). La distribution de ces districts suit des principes de l'aménagement du territoire sans oublier les influences locales et historiques. Ils utilisent deux modèles principales pour leur algorithme : celui d'une ville en Europe de l'Ouest (caractérisée par de grands édifices comme les cathédrales, un dégradé de bâtiments ordonnés plaçant les personnes aisées plus au centre, des subdivisions marquées par les guerres) et celui d'une ville nord américaine (avec les gratte-ciels et une aversion de la vie en centre-ville caractérisée par une dispersion vers les zones éloignées le long des grands axes routiers, se caractérisant aussi par les CBD ou Central Business District réunissant les hubs économiques, sociaux et politiques très peuplés et une couche moins dense composant les habitations, les hôtels et les services médicales, éducatives et les entreprises).

5.2 Méthode

La méthode ne simule pas la croissance de la ville mais un cliché de la ville en trois étapes : mis en place des paramètres basiques entrés par l'utilisateur (le diamètre de la ville, le continent pour définir la distribution des districts, les influences historiques, le nombre d'autoroutes traversant la ville), ensuite le deuxième étape est la génération des districts et enfin le placement de ces districts dans la ville (leurs positions). Ils disposent de 18 types de districts selon les paramètres choisis : 3 districts résidentiels, 2 pour l'industrie (lourde ou légère), 2 districts commerciaux, des noeuds de transport pour les personnes et les biens, les espaces verts et 8 districts spéciaux (cathédrales, squares). Le placement des districts est contraint par cinq facteurs : le type des districts voisins (les zones résidentielles ne sont pas enclin à cohabiter avec les industries), le type de terrain (les industries préfèrent s'implanter près des eaux et les quartiers résidentiels en hauteur de colline), la surface de la ville (pour représenter le dégradé social et économique depuis le centre vers les zones suburbaines), la localisation des rivières et celle des grandes axes routiers (autoroutes). Suivant les modèles d'aménagement de territoire, à chaque district est assigné à un ensemble de valeurs de ces facteurs pour désigner un degré d'attraction/répulsion entre les districts (les industries lourdes sont fortement attirées par les terrains aux bordures de l'eau et ont une forte répulsion vis-à-vis des zones résidentielles de premier plan, et à la fois sont modérément attirées par les autres industries.

L'algorithme de placement des districts est basés sur ces coefficients d'attraction/répulsion :
- la taille de la ville et sa localisation est définie par l'utilisateur
- les autoroutes sont générées autour des centres et se dirigent en dehors de la ville (vers d'autres villes)
- des localisations de districts aléatoires sont générées (environ le double du nombre de districts)

— les meilleures localisations sont choisies
— un diagramme de Voronoi est généré à partir des localisations trouvées définissant des polygones autour de ces localisations et partageant la villes en districts
— Du bruit est ajouté au diagramme pour plus de réalisme
— un réseau routier est généré avec des méthodes connus

La convenabilité notée S de la location l pour contenir un district donné d indique leur degré mutuel de symbiose, en tenant compte des paramètres d'aménagement du territoire choisi. En calculant S pour chaque emplacement généré l et ils assignent à d la location avec une convenabilité S la plus élevée. S est une fonction à cinq paramètres :

1. S_d : placement de d par rapport aux autres n districts
2. S_t : type de terrain
3. S_a : surface à l'intérieur de la ville
4. S_r : distance par rapport aux rivières
5. S_h : distance par rapport aux autoroutes

Example :

S_d : convenabilité d'une localisation par rapport aux districts (au nombre de n) environnants déjàs placés

$$S_d = \sum_{i=1}^{n} S_{d_i} = \sum_{i=1}^{n} A_{d_i} * \frac{\Delta_{d_i}}{\Delta_{min}} \tag{3}$$

avec :

— A_{d_i} : attraction par rapport à un district de type d_i
— Δ_{d_i} : distance par rapport à d_i
— Δ_{min} : distance minimum possible entre les centres de districts

Les coefficients d'attraction (entier entre -100 et 100, représentant respectivement une forte répulsion et une forte attraction) dépendent du type de ville (ex. Européenne avec un cor mercantile). Les distances sont mesurées en pixels.

Les valeurs de chaque paramètre sont alors valuées avec des poids (w) suivant leur importance dans la ville (ex. la distance par rapport aux autoroutes est importante pour une ville américaine) d'où la valeur finale de la convenabilité S suivante :

$$S = w_d * S_d + w_t * S_t + w_a * S_a + w_r * S_r + w_h * S_h \tag{4}$$

La location l avec la valeur de S la plus élévée est choisie pour le district d. Cette algorithme continue jusqu'à ce que tous les districts aient des emplacements, parcourant les districts centraux puis seulement après les districts du reste de la ville. Dans ces deux ensembles (central et périphérique) le placement des districts est aléatoire pour avoir des tracés variés.