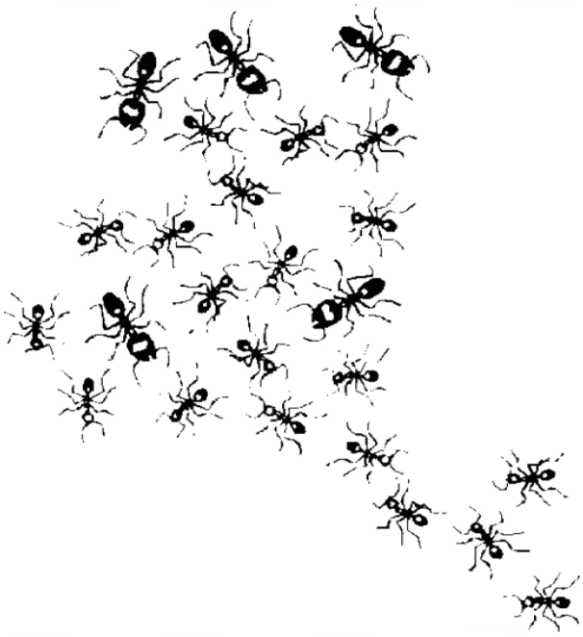
IN52-IN54

A2008

Algorithme des fourmis appliqué à la détection et au suivi de contours dans une image



Etudiants :

Nicolas MONNERET

Alexandre HAFFNER

Sébastien DE MELO

Responsable :

Franck GECHTER

Sommaire

I. Introduction

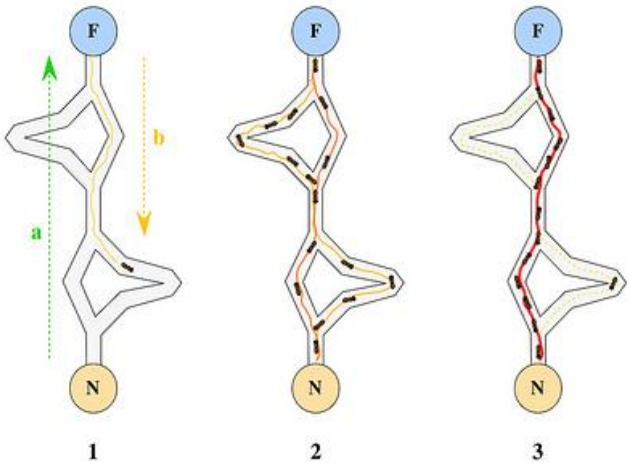
L'extraction de contours joue un rôle primordial dans tout système de vision par ordinateur. De nombreuses méthodes existent tels que l'algorithme avec opérateur de Sobel ou de Prewitt. Le but est d'extraire les zones où les variations locales de l'image sont maximales.
Le problème posé ici est d'extraire les contours dans une image en utilisant l'algorithme des fourmis.

Principe général

L'algorithme des fourmis consiste à faire évoluer des entités (fourmis) dans un environnement et à leur faire déposer de la phéromone en fonction de certaines données de l'environnement. Les fourmis réagissent à la phéromone déposée sur l'environnement et sont attirées par elle. La phéromone déposée sur l'environnement s'évapore au bout d'un certain temps.

Exemple d'algorithme de fourmis

Un algorithme classique utilisant les fourmis est la recherche de plus court chemin entre deux points. Les fourmis se déplacent aléatoirement. Si elles rencontrent de la nourriture (le point d'arrivée), elles retournent au nid (point de départ) en déposant des phéromones.
Toutes les fourmis de la colonie étant attirées par les phéromones, elles vont donc se rendre à la nourriture en suivant celles-ci. En revenant au nid, ces fourmis vont renforcer la piste de phéromones.
Si deux pistes sont possibles pour atteindre la même source, la plus courte sera parcourue par plus de fourmis, dans le même temps, que la longue. La piste courte sera donc de plus en plus renforcée et la piste longue va disparaître, les phéromones étant volatiles.
Après plusieurs itérations, la piste va tendre à devenir le plus court chemin du nid jusqu'à la nourriture.



Application à la détection de contours

Pour appliquer ce type d'algorithme sur une image dans le but de détecter les contours, l'idée est de déposer de la phéromone en fonction du gradient. L'image en niveaux de gris représentant l'environnement sur lequel les fourmis vont évoluer. Si, entre deux pixels consécutifs, les niveaux de gris sont très différents, la fourmi va déposer beaucoup de phéromones. Le maximum de phéromones se situera donc sur les contours, là ou la variation est maximale, ce qui aura pour effet d'attirer les autres fourmis qui vont déposer à leur tour des phéromones.
A la fin des itérations, il nous suffira de dessiner une carte des phéromones déposées pour obtenir les contours de l'image.

L'algorithme haut niveau est le suivant :

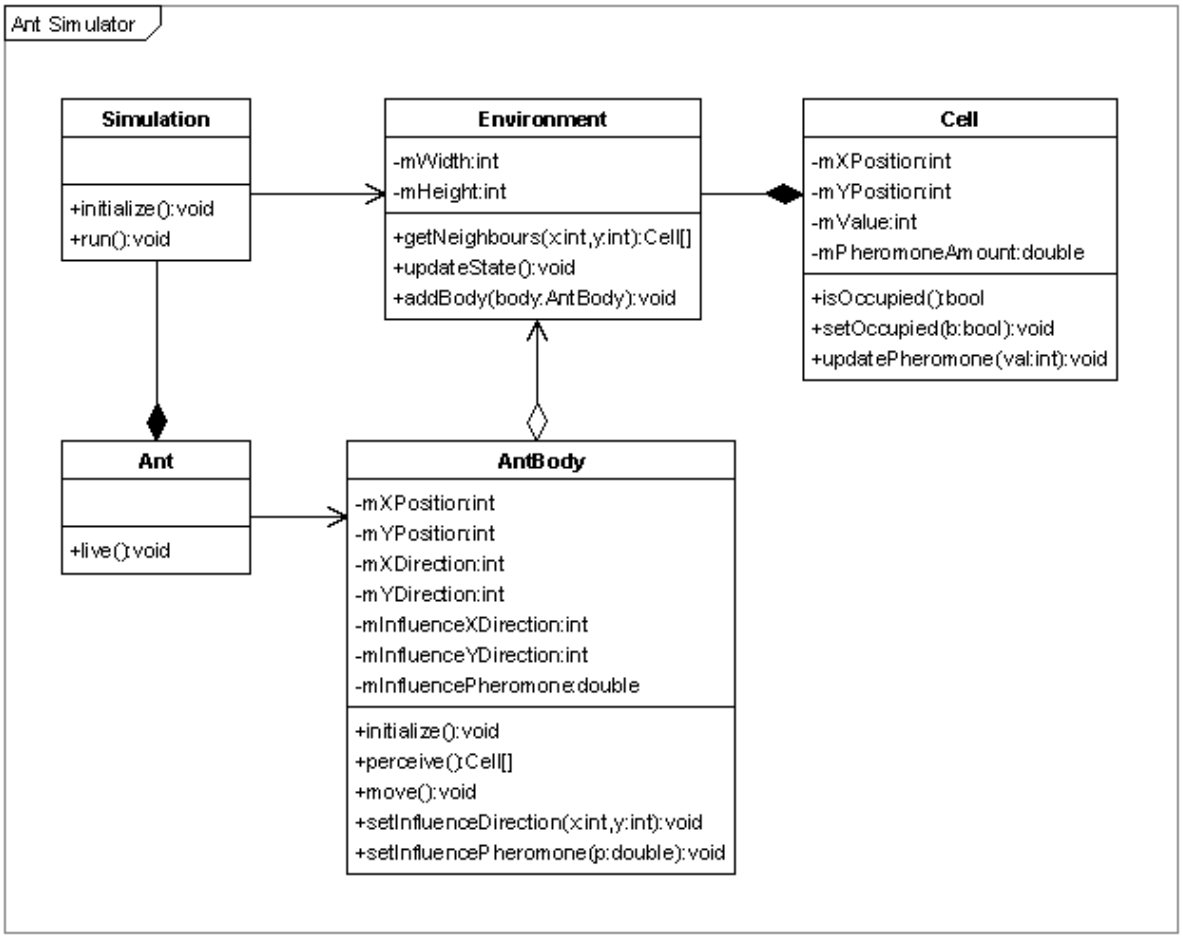
```
Pendant t itérations
        Pour chaque fourmi
                Déterminer sa prochaine position en fonction des phéromones
                Déterminer le nombre de phéromones à déposer en fonction des niveaux de gris
                Déplacer la fourmi si la case n'est pas occupée
                Déposer les phéromones
        Fin Pour
        Evaporer k phéromones sur chaque cellule de l'environnement
Fin
```

II. Architecture du simulateur

Pour créer notre population de fourmis et notre environnement, il est nécessaire de créer un simulateur multi-agents.

Diagramme des Classes

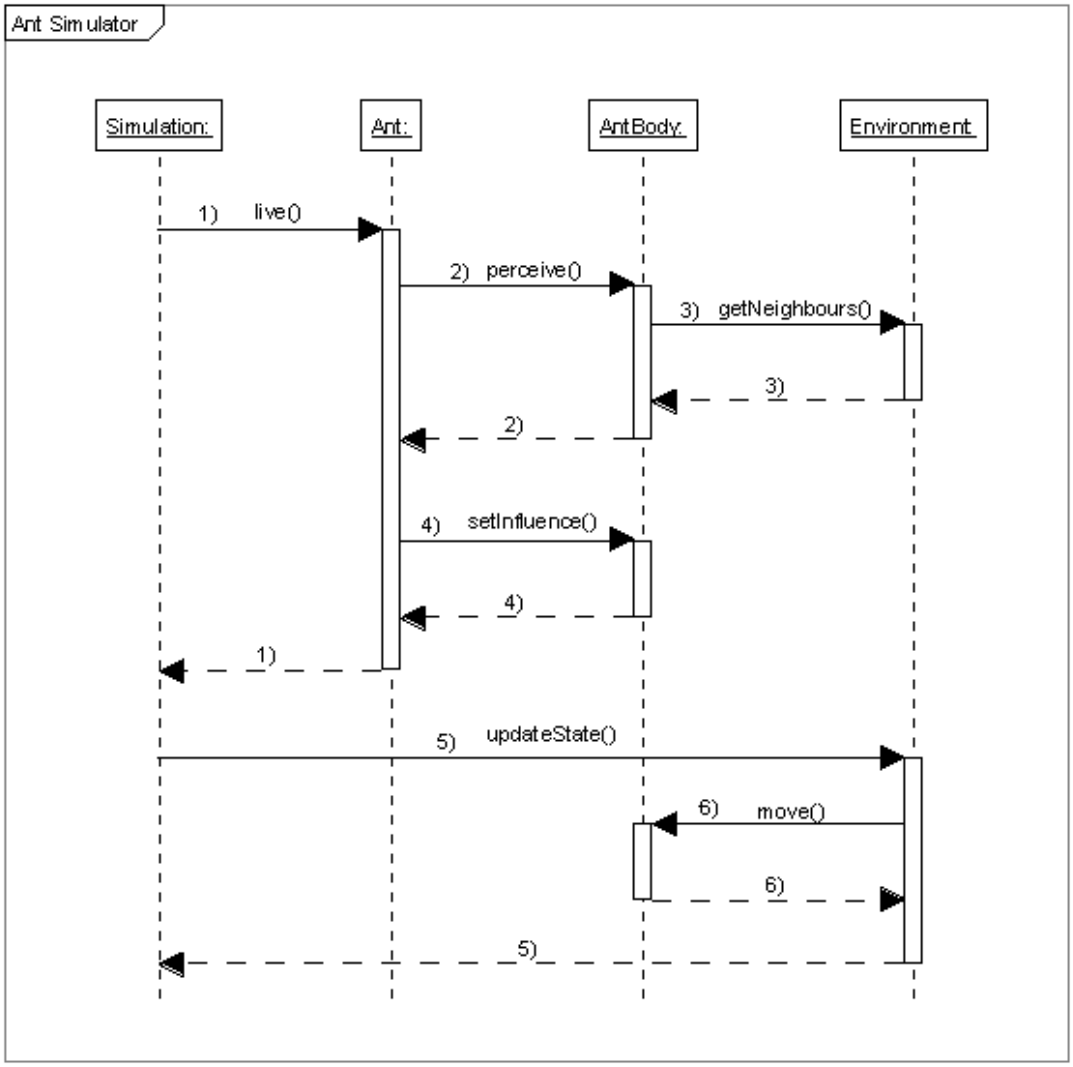
Le simulateur est composé d'un environnement et d'une liste de fourmis. Chaque fourmi dispose d'un corps qui va interagir avec l'environnement. On sépare ainsi les tâches de traitement (la décision) et les phases de déplacement et d'interaction avec l'environnement (l'action).
L'environnement, lui, est composé de cellules qui représentent les pixels de l'image est qui contiennent l'information sur les niveaux de gris et les phéromones déposées.



Fonctionnement

A chaque tour, on fait vivre les fourmis. En fonction des informations sur le voisinage rapportées par leur corps (perception), elles choisissent une direction et un taux de phéromones à déposer sur l'environnement (influence).

Une fois que toutes les fourmis ont effectuées leurs calculs, on met à jour l'environnement avec les influences. C'est l'environnement qui décide si une fourmi peut se déplacer et déposer des phéromones. Si une fourmi veut se déplacer sur une case déjà occupée, l'environnement ne va pas appliquer l'influence.



III. Ant.live()

Le calcul du déplacement et du nombre de phéromones à déposer est effectué dans la méthode live() de la classe Ant.

Déplacement

Le mouvement des entités est déterminé selon deux critères :

- le précédent déplacement (sa direction)
- le taux de phéromone dans les huit cases environnantes.

On inclut donc un biais directionnel dans le déplacement. Une fourmi aura une probabilité beaucoup plus importante de continuer dans sa direction que de se retourner.
La probabilité est définie comme ceci suivant l'angle de rotation :

Soit Δ_i la rotation à effectuer pour aller dans la cellule i et $w(\Delta_i)$ la fonction de calcul du biais.

Δ_i	0°	45°	90°	135°	180°
$w(\Delta_i)$	1	1/2	1/4	1/12	1/20

Si une entité vient du sud, on aura donc :

½	1	1/2
1/4	Ant	1/4
1/12	1/20	1/12

La fonction qui fait intervenir le niveau de phéromones dans le déplacement est celle-ci :

$$W(\sigma_i) = (1 + \frac{\sigma_i}{1 + \delta\sigma_i})^{\beta}$$

σ_i est le nombre de phéromones présentes dans la cellule i.
β représente la sensibilité osmotropotaxique, c'est-à-dire la tendance, pour une fourmi, à suivre la phéromone ou non. Plus la valeur est élevée, plus les fourmis vont être attirées par la phéromone présente dans la cellule.
$1/\delta$ est la capacité sensorielle. Lorsque le taux de phéromones est élevé, ce facteur permet de faire diminuer l'habilité des fourmis à sentir cette phéromone.

La formule globale de déplacement des fourmis est donc la suivante :
Soit P_{ik} la probabilité pour une entité k de se déplacer dans la cellule i.

$$P_{ik} = \frac{W(\sigma_i)w(\Delta_i)}{\sum_{j/k} W(\sigma_j)\, w(\Delta_j)}$$

Après avoir calculé cette probabilité pour chaque cellule du voisinage, nous choisissons la cellule vers laquelle l'entité va se déplacer par une méthode aléatoire basée sur les probabilités.

Phéromones

Le calcul du taux de phéromone à déposer est fonction de la différence des niveaux de gris entre la cellule avant le déplacement et la cellule après le déplacement.
A chaque itération, la fourmi dépose une quantité constante η de phéromone et une quantité variable $p\Delta_{\mathrm{gl}}$.

Δ_{gl} est la différence entre la valeur médiane des niveaux de gris de la précédente cellule et de ses voisines, et la valeur médiane des niveau de gris de la cellule courante et de ses voisines.

Nous avons donc un nombre de phéromones déposées comme ceci :

$$T = \eta + p\frac{\Delta_{gl}}{255}$$

Δ_{gl} est compris entre 0 et 255 (différences des niveaux de gris), donc le taux variable de phéromones déposées est compris entre 0 et la constante p.
T est donc compris entre η et $\eta + p$. Dans la cas de grandes surfaces uniformes en niveaux de gris, T sera égal à η. Dans le cas de contours maximaux, T sera égal à $\eta + p$.

Grâce à ces comportements de décision du déplacement et du nombre de phéromones déposées, ajoutés à l'évaporation des phéromones à chaque itération, nous avons un mécanisme qui va faire apparaître un taux de phéromone élevé sur les contours de l'image (gradient fort).

Taille de la population variable

Il existe deux types d'algorithmes pour les populations de fourmis :

- On peut choisir d'appliquer l'algorithme des fourmis avec une population à taille fixe (*Swarms with Fixed Population Size - SFPS*).
- Dans notre cas, nous allons aussi mettre en place une population dont la taille est variable, ce qui permettra de réguler le nombre de fourmis en fonction des contours de l'image (*Swarms with Variable Population Size - SVPS*).

La vie est la mort des entités sont définies dans la méthode live() de la classe Ant.

Les fourmis stockent une quantité d'énergie qui va croître ou décroître en fonction :

- de son âge
- des niveaux de gris de l'image

Plus cette énergie est faible, plus l'agent à une probabilité forte de mourir.
Soit e(a) cette énergie.
a est l'âge de la fourmi en nombre d'itérations.

A la naissance, cette quantité d'énergie est définie comme étant :

$$e(0) = 1 + \alpha$$

Le calcul de l'énergie se fait à chaque itération comme ceci :

$$e(t) = e(t-1) - \alpha + \alpha(\frac{\Delta_{gl}}{max\Delta_{gl}})$$

Δ_{gl} est la différence de niveaux de gris que l'on a calculé précédemment.
$max\Delta_{gl}$ est le maximum des Δ_{gl} que l'on a obtenu sur toutes les précédentes itérations.

Au final, une entité aura une probabilité de mourir qui sera définie comme ceci :

$$P = 1 - e(a)$$

La naissance d'une fourmi est conditionnée par

- le nombre de cellules occupées par d'autres agents autour de la mère
- la différence des niveaux de gris entre deux cellules

Comme pour les déplacements, on donne un poids suivant le nombre de fourmis voisines. Celui-ci est défini comme ceci :
Si n est le nombre de cellules voisines occupées, le poids W(n) est défini comme ceci :

n	0	1	2	3	4	5	6	7	8
W(n)	0	0,25	0,5	0,75	1	0,75	0,5	0,25	0

A cette probabilité, il faut ajouter la prise en compte de la variation du niveau de gris de l'environnement.

La formule finale qui nous donne une probabilité de mise au monde d'un enfant est donc celle-ci :

$$P = W(n) * \left(\mu + (1-\mu)\left(\frac{\Delta_{gl}}{max\Delta_{gl}}\right)\right)$$

μ est une constante qui permet de laisser une chance minimale de reproduction aux fourmis, même si la différence de niveaux de gris entre deux cellules est nulle.

Avec cette méthode, nous régulons la population en éliminant les fourmis se situant sur des espaces sans variation de niveaux de gris et en ajoutant des fourmis là où le gradient est le plus fort, c'est-à-dire sur les contours. Ceci permet de faire converger les fourmis sur les contours en adaptant le pourcentage de fourmis dans l'environnement au pourcentage de contours de l'image.

IV. Développement

Afin de mettre en œuvre cet algorithme sur une série d'image et de vidéo, et comparer chacun des méthodes que nous avons vue, à savoir le travail avec une population fixe ou variable, nous avons utilisé la bibliothèque de fonction OpenCV qui permet de manière extrêmement simple et rapide de rendre tout ceci visuellement.

Nous avons utilisé la bibliothèque, dans sa version 1.1pre1, pour réaliser quelques tâches importantes, à savoir :

- Charger des images et vidéos comme base de travail (environnement).
- Créer et modifier des images pour le rendu (carte des phéromones et fourmis).

La classe EdgeDetector se charge de travailler à partir de la plateforme de simulation multi agents pour en extraire les informations que l'on souhaite afficher. Nous avons pris soin de la dédier uniquement à l'affichage, et donc la détacher complètement du système multi agent.

Cette classe se charger de créer et actualiser trois fenêtres :

- La fenêtre source contenant l'image ou la vidéo d'origine.
- La fenêtre phéromones contenant la carte des phéromones de la simulation en cours.
- La fenêtre fourmis contenant la carte des fourmis (leur position est signalée par un point blanc).

Application à une image

Lors de l'application de l'algorithme à une image unique, la procédure suivante est suivie. On charge dans un premier temps l'image initiale (source) via la fonction cvLoadImage, ainsi que sa version niveau de gris. On initialise ensuite une simulation de taille NxM avec un nombre initial de fourmis. Ce nombre restera constant dans le cas d'une SFPS (population fixe), mais pourra varier lors de l'utilisation en mode SVPS (population variable).

Après avoir modifié l'environnement de la simulation pour qu'il corresponde à l'image initiale en niveau de gris, on lance la simulation sur un nombre d'itérations donné. La carte des phéromones est alors extraite. On recherche dans un premier temps la phéromone maximum afin d'étaler les valeurs sur l'intervalle [0 ; 255] et l'afficher. Elle correspond aux contours de l'image. Afin de laisser à l'utilisateur juger du déplacement des fourmis, on extrait de la même manière la carte des fourmis en assignant un niveau de gris égale à 255 au pixel (x,y) si et seulement si une fourmis se trouve à cette position.

L'utilisateur à la possibilité de boucler sur l'ensemble afin de voir évoluer la création des contours et le déplacement des populations de fourmis, et ceci jusqu'à de qu'une touche soit pressée (cvWaitKey).

Application à une vidéo

Lors de l'utilisation de l'algorithme sur une vidéo (provenant d'un fichier ou d'une caméra), on travail frame par frame. Le chargement de la vidéo se faire via la fonction cvCaptureFromAvi. Il est alors possible d'extraire les frames de la vidéo de manière séquentielle via les fonctions cvGrabFrame et cvRetrieveFrame.

Dans ce mode, on prépare la simulation avec la première image de la vidéo. On réalise ensuite une détection de contours pendant un nombre d'itérations X. Une fois la détection réalisée, on modifie l'environnement de la simulation en y introduisant les données de l'image suivante, sans modifier ni la carte des phéromones, ni la disposition des agents. On relance alors la simulation (depuis son état précédent, mais avec un environnement modifié donc) pour un nombre Y d'itérations. Les fourmis se déplacent donc depuis leur position sur l'image précédente et tentent de s'adapter à la nouvelle image, à leur nouvel environnement.

Paramètres de la simulation

On constatera plus loin que la qualité de l'extraction dépend en partie du choix des paramètres en relation avec le type d'image. Afin de permettre à l'utilisateur de stocker un ensemble de configurations différentes pour sa simulation, nous avons mis en place un système de script de paramétrage décrivant tout ou partie des données nécessaires au bon fonctionnement de la simulation. Celui-ci se décompose en trois grandes parties.

- Propriétés générales de la simulation

Propriété	Type	Description
MEDIA_TYPE	(image|movie|camera)	Le type de media à savoir image, vidéo ou caméra.
INPUT_PATH	string	L'image ou la vidéo en entrée.
SIMULATION_TYPE	(sfps|svps)	Le type de simulation, à population fixe ou variable.
NB_ANTS	int	Le nombre de fourmis initial.
NB_ITERATIONS	int	Le nombre d'itérations initial, pour la première extraction.
NB_ITERATIONS_FOLLOW	int	Le nombre d'itérations entre deux étapes consécutives.
LOOP	(true|false)	Souhaite-t-on boucler l'analyse d'une image (pas applicable aux vidéos) ?

- Propriétés des fourmis

CONSTANT_PHEROMONE_DEPOSIT	double	La quantité de phéromone déposée quoi qu'il arrive.
VARIABLE_PHEROMONE_DEPOSIT	double	La quantité de phéromone déposée fonction du gradient.
OSMOTROPOTAXIC_SENSITIVITY	double	La tendance à suivre les phéromones.
SENSORY_CAPACITY	double	La capacité à détecter de forte quantité de phéromones.
ENERGY_FACTOR	double	L'énergie ajouté ou retiré à la fourmi à chaque étape de sa vie.
REPRODUCTION_CHANCE	double	La chance de se reproduire dans un milieu homogène.

- Propriétés de l'environnement

PHEROMONE_EVAPORATION	double	La quantité de phéromone évaporée dans chaque cellule par itération.
PHEROMONE_MAXIMUM	double	La quantité maximale de phéromone qu'une cellule peut accueillir.

Le script comporte sur chaque ligne une propriété suivie de sa valeur conformément au type de la propriété, séparés par un espace. Une ligne commençant par le caractère # sera ignorée. On pourra donc s'en servir pour commenter le script.

V. Résultats obtenus

Nous allons maintenant pouvoir, à travers un ensemble de jeux d'essais, constater de l'efficacité de chacune des méthodes, à savoir la SFPS (population fixe) et la SVPS (population variable), à la fois sur des images fixes mais aussi sur des séquences d'images pour lesquelles nous tenterons d'adapter le modèle.

Dans tous les cas, il s'avère que le choix des paramètres est complexe et fortement dépendant de ce que l'on cherche à réaliser. L'introduction d'une population variable nivelle de façon relative les effets de certains paramètres en introduisant une dynamique différente aux agents, mais introduit une nouvelle complexité concernant la régulation de la taille de la population.

Sur l'ensemble des tests effectués, on constate que l'utilisation d'une population fixe ne permet pas une flexibilité et une adaptation rapide à l'environnement comparée à la population variable. Une des particularités de système à population variable est de s'auto réguler. Il est ainsi capable de s'adapter à l'environnement.

Lors de l'utilisation d'une population de taille variable, on constate deux comportements différents, en fonction du nombre d'entités initialement placées dans l'image :

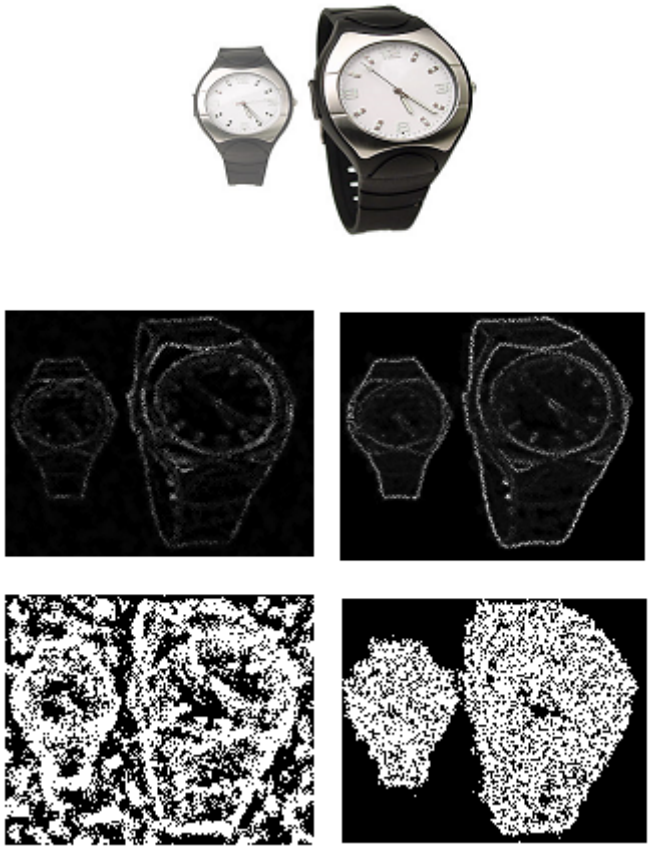
- S'il y a trop peu d'entités placées initialement dans une image en grande partie homogène, la population chute jusqu'à la disparition de toutes les fourmis. Elles n'ont en fait pas réussi à atteindre des zones intéressantes pour se reproduire.
- S'il y a un nombre raisonnable d'entités pour couvrir l'ensemble des zones de l'image, elles régulent, en fonction des paramètres, leur population pour atteindre un nombre indépendant du nombre d'entités initialement placées.

On constate qu'un mauvais réglage des paramètres concernant la reproduction, et en particulier « REPRODUCTION_CHANCE », peut mener à une extinction plus rapide en présence de grandes zones homogènes ou à contrario une prolifération trop importante (pour atteindre généralement les limites de l'image) en raison de conditions trop favorables. Dans les deux cas, le résultat est critique, d'un coté par la non détection des contours, et de l'autre par l'apparition d'artéfacts et la perte importante de performances.

Voyons maintenant à travers plusieurs exemples comment se traduises ces différents phénomènes.

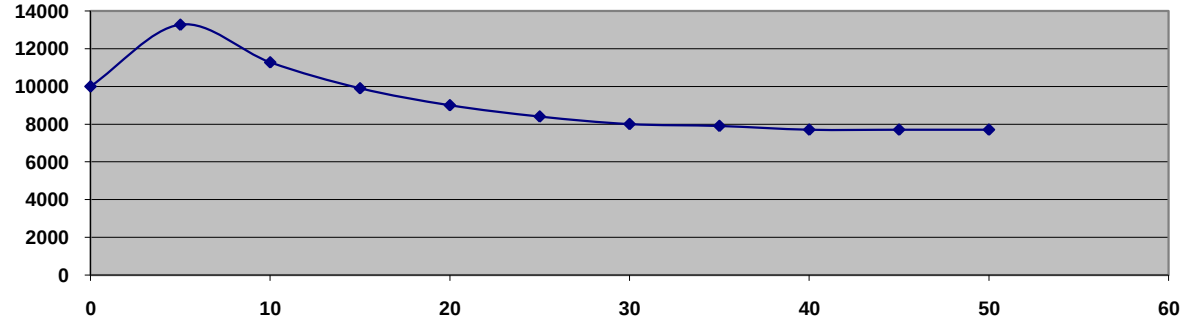
Images fixes

Dans le cadre de l'analyse d'images fixes, il n'y a aucune contrainte de temps et aucun phénomène de modification de l'environnement dans lequel les agents se déplacent. Il est généralement possible d'obtenir par les deux méthodes des résultats très correctes. Toute la recherche qu'il est intéressant de mener porte alors sur la rapidité de convergence du modèle.



Résultat après 150 itérations sur « montre ». A gauche la SFPS, à droite la SVPS.
De haut en bas : image originale, carte de phéromones, carte des fourmis.

On constate un contraste accru par la SVPS pour cette image. Il est intéressant d'analyser la carte des fourmis. On voit clairement qu'elles ont complètement disparues des zones homogènes blanches de l'image pour la SVPS, alors qu'elles subsistent en petits amas un peu partout avec la SFPS. Le nombre de fourmis placées initialement pour ce test est de 10 000 pour chacune des images. La SVPS varie comme suit :



La population augmente substantiellement dans un premier temps afin de s'étendre dans l'environnement puis se stabilise autour de 7700 individus après seulement 35 itérations. Les fourmis sont alors déjà bien en place sur les contours.

La montre est une image relativement simple à étudier, avec un fort contraste et des contours marqués. Certaines images sont beaucoup plus problématiques, surtout avec une population variable, laquelle tend parfois à extraire trop d'information.

Résultat après 100 itérations sur « baboon » en SVPS.
De gauche à droite : image originale, carte des phéromones, carte des fourmis.

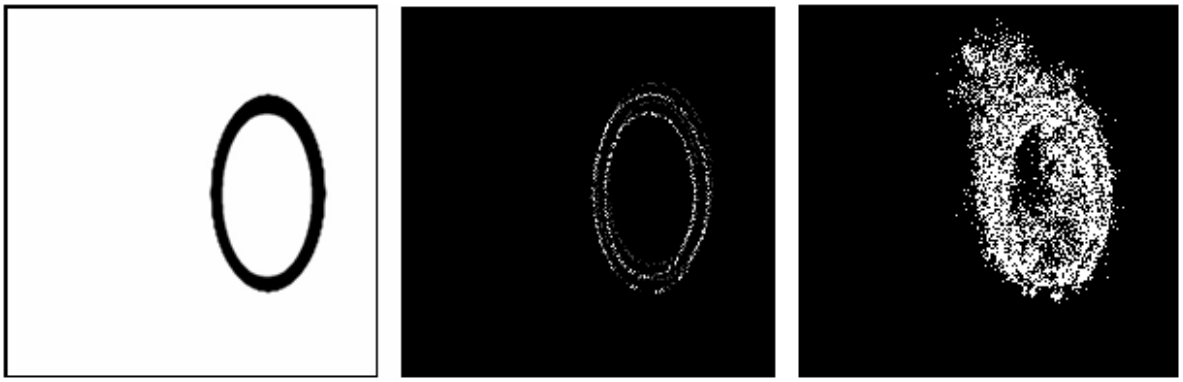
On constate ici que les contours sont visibles mais la carte des fourmis est complètement saturée (à 75% pour ce jeu d'essai). Le nombre d'individus présents permet l'extraction des contours de manière correcte mais il en résulte néanmoins une perte importante de performances.

Séquences d'images

Nous avons mis en place la possibilité d'étudier le comportement sur des vidéos et captures via caméra. Le principe est exactement le même, mais de nouveaux facteurs sont à prendre en compte. Le temps de calcul mis à part, c'est l'évaporation de la phéromone qui devient un paramètre fondamental, car de lui résultera ou non un effet de rémanence sur la séquence des images des contours. En effets, on réutilise la carte des phéromones lors du

passage dans un environnement nouveau (pour une nouvelle image). Si celles-ci n'ont pas le temps de s'évaporer il en résultera des traînées sur l'image suivante.

Il est donc nécessaire d'affiner les réglages et de réaliser un compromis entre d'un coté l'évaporation des phéromones, mais aussi la convergence rapide des fourmis pour obtenir une séquence fluide, si tant est que l'on décide de la lire en temps réel. On constate qu'il faudra généralement améliorer les contrastes en mettant en place un maximum de phéromones par cellule impossible à dépasser afin de permettre l'évaporation de sa phéromone plus rapidement une fois les fourmis parties.



Suivis des contours sur la séquence « circle_move ». L'ellipse de déplace en bas à droite. De gauche à droite : image originale, carte des phéromones, carte des fourmis.

On constate sur la carte des phéromones l'effet de rémanence issu des images précédentes. Il est ici minimisé au maximum par le choix de paramètres adapté à l'image. On constate que les fourmis sont capables de suivre les contours (en termes de position sur la carte) par un phénomène de reproduction. Le travail avec d'autres séquences, comportant des images plus éloignées, montre qu'il est plus difficile pour les fourmis de retrouver la forme qu'elle suive lorsque celle-ci s'éloigne trop. Ceci est du au fait qu'il leur faudra traverser la plupart du temps un environnent homogène souvent mortel pour l'essaim si les paramètres ne sont pas ajustés en adéquation avec ce grand déplacement (à savoir une augmentation de la possibilité de reproduction dans les milieux homogènes).

Le travail avec les séquences d'images complique fortement le paramétrage mais donne dans certains cas des résultats concluants. L'idée principale est d'itérer N fois sur la première image de la séquence pour laisser aux fourmis le temps de s'adapter aux contours, puis d'itérer M ($M < N$) fois pour chaque images suivantes afin de minimiser le temps de calcul, si l'on suppose bien sûr que les fourmis sont capables de s'adapter rapidement au décalage subit.

Il est aussi possible de voir l'évolution des fourmis lors de l'utilisation d'une population fixe dans une séquence, mais leur faible faculté d'adaptation à un changement d'environnement rend la chose peu exploitable.

VI. Conclusion

Nous avons mis en place un algorithme de colonie de fourmis pour la détection de contours dans une image et le suivi dans une séquence. L'algorithme peut être utilisé sur une taille de population fixe comme sur une population à taille variable (SFPS et SVPS).

Le principal problème rencontré est dû aux nombreux paramètres de l'algorithme, qui s'avèrent au final être difficiles à régler en fonction du résultat que l'on souhaite obtenir et du type d'image. Ces paramètres sont d'autant plus difficiles à mettre en place dans le cas de l'étude d'une séquence d'images.

Les algorithmes de détection de contours étant nombreux, il serait bon de comparer les performances en termes de qualité de la détection, mais aussi de temps d'exécution. L'algorithme des fourmis étant assez long à l'exécution pour détecter un simple contour sur une image. Son avantage résiderait donc dans le fait que les fourmis s'adaptent à l'environnement, ce qui serait utile lorsqu'il y'a évolution de l'image, pour une vidéo par exemple.

Une alternative pourrait éventuellement être de combiner l'algorithme des fourmis à d'autres algorithmes de détections de contours afin d'augmenter leur efficacité.