12/04/2022

Compte Rendu MAP401

Vectorisation et simplification d'image bitmap

Tobias SAVARY – Albin HORLAVILLE
MIN - 1

Table des matières

Tâche 1 : Module Image

Cette tâche nous a permis de nous familiariser avec les différents modules mis à notre disposition tel que le module image. Dans cette tâche, nous avons créé un module image dans lequel on trouve des fonctions permettant de calculer le négatif d'une image ainsi que d'écrire une image dans le terminal. Pour écrire une image, on parcourt chaque pixel. Si le pixel est noir, on affiche un « X » et sinon, le pixel est alors blanc. Dans ce cas, on affiche un espace. Pour la négation de l'image, on parcourt chaque pixel. Si le pixel est noir, on le remplace par un pixel blanc. Sinon, on fait l'inverse.
Nous avons aussi créé un programme de test pour ce module image nommé test_image. Il se lance grâce à la commande « ./test_image image.pbm » (où l'image.pbm représente l'image que l'on veut tester). Ce programme prend une image en paramètre et permet de vérifier si les fonctions du module image fonctionnent correctement.

Tâche 2 : Module Geom2d

Pour réaliser cette tâche, nous avons créé un module geom2d qui permet la gestion de la géométrie. Ce module contient la définition des types Points et Vecteur ainsi que plusieurs fonctions permettant la gestion de ces éléments. Ces fonctions permettent d'effectuer des opérations avec ces deux types : Somme des vecteurs et des points, produit entre vecteurs et points, produit scalaire, la norme, la distance…
Nous avons aussi créé un programme de test pour ce module geom2d nommé test_geom. Il est exécuté avec la commande « ./test_geom ». Il permet de tester les fonctions set_point, add_point, produit_entre_réel_et_points ou entre_réels_et_vecteurs, vect_bipoint, norme_vecteur, distance_entre_points, produit_scalaire…

Tâche 3 : Extraction de Contour

Pour cette tâche, nous avons créé un module calcul_contour qui permet la gestion des déplacements du robot sur l'image, la détection des pixels candidats pour le calcul du contour, la création et la gestion de l'Image de Masque ainsi que le stockage de ce contour dans une liste. A noter également la création du module ecriture_contours pour gérer l'écriture d'une liste de contours passée en argument dans un fichier ou dans le terminal. Nous avons mis au point un module pour gérer les listes chainées de points qui s'appelle listes_chainees_points.

Pour tester cette tâche, nous avons implémenté un programme test_contour qui prend en argument une image et un fichier de sortie. Il est exécuté avec la commande «./test_contour image.pbm image.txt ». Ce programme permet de calculer les contours de l'image entrée en argument et d'écrire la liste des points qui composent le contour dans un fichier donné en second argument.

Tâche 4 : Sortie au format PostScript encapsulé

Dans cette tâche, il est question de créer un programme qui va mettre l'image sous forme EPS (Encapsulated PostScript), pour pouvoir l'afficher avec la commande gv. Pour cela, nous avons créé un module sortie_EPS qui nous permet de gérer toutes les fonctions relatives au format EPS.

Pour tester notre programme, nous avons créé test_EPS qui prend en argument une image en « .pbm » et un fichier de sortie et écrit dans celui-ci l'image au format EPS. L'exécutable affiche aussi le nombre de segment de l'image dans le terminal. Cette tâche peut être exécuté à l'aide de la commande « ./test_EPS image.pbm image.EPS »

Tâche 5 : Extraction des contours d'une image

Pour répondre à la question de la tâche 5, nous avons mis à jour les modules calcul_contour et sortie_EPS pour permettre de mettre dans une liste de contours tous les contours d'une image. Ce module permet de traiter des images qui contiennent plusieurs contours.

L'exécutable de la tâche 5 est ecrire_liste_contours. Cet exécutable peut être utilisé de 2 façons : Pour la Tâche 5.1, l'exécutable prend deux arguments : une image au format pbm et un fichier txt. L'objectif de ce programme est de renvoyer dans le fichier texte le nombre total de contours sur la première ligne. Et, pour chaque contour, le nombre de points présent dans ce contour ainsi que les coordonnées de tous les points qui composent ce contour. Il peut être lancé à l'aide de la commande « ./ ecrire_liste_contours image.pbm image.txt ». Il est aussi possible d'utiliser ce programme avec trois arguments. Dans ce cas, le programme prend un argument une image.pbm, un fichier de sortie.EPS et un mode (1 pour remplissage et 0 pour contours). Ce programme permet alors d'écrire dans le fichier de sortie l'image au format EPS selon le format spécifier afin que celle-ci soit affichable par la commande gv. Il peut être lancé à l'aide de la commande « ./ ecrire_liste_contours image.pbm image.EPS Mode ».

Tâche 6 : Simplification de contour par segment

Tâche 6.1:

Pour cette tâche, nous avons mis à jour le module geom2d pour y intégrer une fonction distance_point_segment qui calcule la distance entre un point (donné en argument) et un segment dont les deux extrémités sont données en paramètre.

L'exécutable est distance_pts_seg.c et permet à l'utilisateur d'entrer les coordonnées de 3 points pour calculer la distance entre le premier et le segment formé par les deux autres. Il peut être lancé à l'aide de la commande « ./ distance_pts_seg».

Tâche 6.2:

Pour cette tâche nous avons codé le module Douglas_Peucker qui comporte l'algorithme du même nom. Nous avons créé le module liste_chainees_contours, qui nous permet d'utiliser des listes chainées de contours (liste de liste de points). Chaque liste de points correspond à un contour. Une liste de contours est donc composée de plusieurs listes de point. La fonction Douglas Peucker reçoit un tableau de point, deux entiers j1 et j2 ainsi que la distance maximale entre un point et un segment. Elle renvoie une liste de points, correspondant à l'image simplifiée.

Dans cette tâche l'exécutable est nommé simplification_par_segment. Ce programme prend en argument l'image.pbm que l'on cherche à simplifier, une distance pour effectuer la simplification, un mode (1 pour remplissage et 0 pour contours) et un fichier de sortie.EPS dans lequel sera inscrite l'image simplifiée grâce à la méthode des segments. Il est possible d'éxecuter ce programme à l'aide de la commande
« ./ simplification_par_segment image.pbm mode distance image.EPS » .Ensuite avec la commande gv on peut visualiser l'image créée par le programme.

Tâche 7 : Simplification de contour par courbe de Bézier

Tâche 7.1 : Simplification par courbe de degré 2

Dans cette tâche nous avons mis au point les modules liste_contour_bezier et listes_chainees_bezier. Le premier permettant de gérer des listes de listes de courbe de degré 3, et le deuxième permettant de gérer des listes de courbes de Bézier de degré 3. On a aussi créé le module courbe_bezier qui définit les types Bézier 2, Bézier 3 et comporte une fonction pour calculer un point d'une courbe de degré 2, une pour convertir une courbe de degré 2 en courbe de degré 3, une fonction d'approximation d'une courbe de Bézier 2 à partir de 2 points ou plus et une fonction renvoyant la distance entre un point et une courbe de Bézier de degré 2. Le module Douglas_Peucker a été amélioré pour pouvoir accueillir une nouvelle version de l'algorithme alias simplification_douglas_peucker_bezier2 qui prends un tableau de point, deux entiers j1 et j2, et la distance seuil d et renvoie une liste de Bézier 3 c'est à dire le contour simplifié par courbe de Bézier de degré 2. A noter qu'avant d'être ajouté à la liste chainée de Bézier 3, la courbe est convertie en courbe de degré 3 car le format EPS ne prend en charge que des courbes de Bézier de degré 3. Le module sortie_EPS se trouve élargie et voit arriver la fonction ecriture_fichier_EPS_bezier3. Elle reçoit une liste de contours de courbes de Bézier de degré 3 et écrit dans un fichier de sortie en ".EPS" les courbes de Bézier à l'aide de l'instruction "curveto" au lieu de "lineto".

L'exécutable de cette tâche se nomme simplification_par_bezier2. Il a besoin en argument d'une image « .pbm », d'une distance seuil d, du mode d'affichage (0=strocke ou 1=fill) et d'un fichier de sortie en ".EPS". On applique la fonction simplification_douglas_peucker_bezier2 à chaque contour de l'image et le programme écrit l'image simplifiée dans le fichier de sortie. On peut l'exécuter avec la commande
« ./ simplification_par_bezier2 image.pbm mode distance image.EPS ».
En bref, cet exécutable permet d'effectuer une simplification de l'image.pbm assez similaire à celui de la Tâche 6.2 mais en utilisant la méthode des courbes de Bézier de

degrés 2 au lieu de celle des segments. L'utilisateur peut alors visualiser le résultat grâce à la commande gv.

Tâche 7.2 : Simplification par courbe de degré 3

Pour cette tâche nous reprenons la tâche 7.1 mais en adaptant les types et fonctions pour que le programme simplifie non plus par courbe de Bézier de degré 2, mais par courbe de Bézier de degré 3. Douglas_Peucker a été remis au goût du jour grâce à l'implémentation de la fonction simplification_douglas_peucker_bezier3 qui construit directement des courbes de degré 3 avec approx_bezier3. Naturellement il devient inutile de convertir la courbe, cette ligne n'est donc plus présente dans la fonction. Nous avons mis à jour le module courbe_bezier pour y ajouter une fonction de calcul d'un point d'une courbe de Bézier de degré 3, une fonction d'approximation de Bézier 3 citée plus haut approx_bezier3 qui reçoit un tableau de point et deux entiers j1 et j2 et renvoie une Bézier de degré 3. La fonction distance_point_bezier2 est reprise et adapté pour devenir distance_point_bezier3.

Son exécutable est simplification_par_bezier3. Il a besoin en argument d'une image « .pbm », de la distance seuil d, du mode et du fichier de sortie ".EPS". Le programme écrit ensuite l'image simplifiée en courbe de Bézier de degré 3 en fonction de la distance seuil dans le fichier de sortie. On peut l'exécuter avec la commande
« ./ simplification_par_bezier3 image.pbm mode distance image.EPS ».
En bref, cet exécutable permet d'effectuer une simplification de l'image.pbm assez similaire à celui de la Tâche 7.1 mais en utilisant la méthode des courbes de Bézier de degrés 3 au lieu de celle des courbes de Bézier de degrés 2. L'utilisateur peut alors visualiser le résultat grâce à la commande gv.

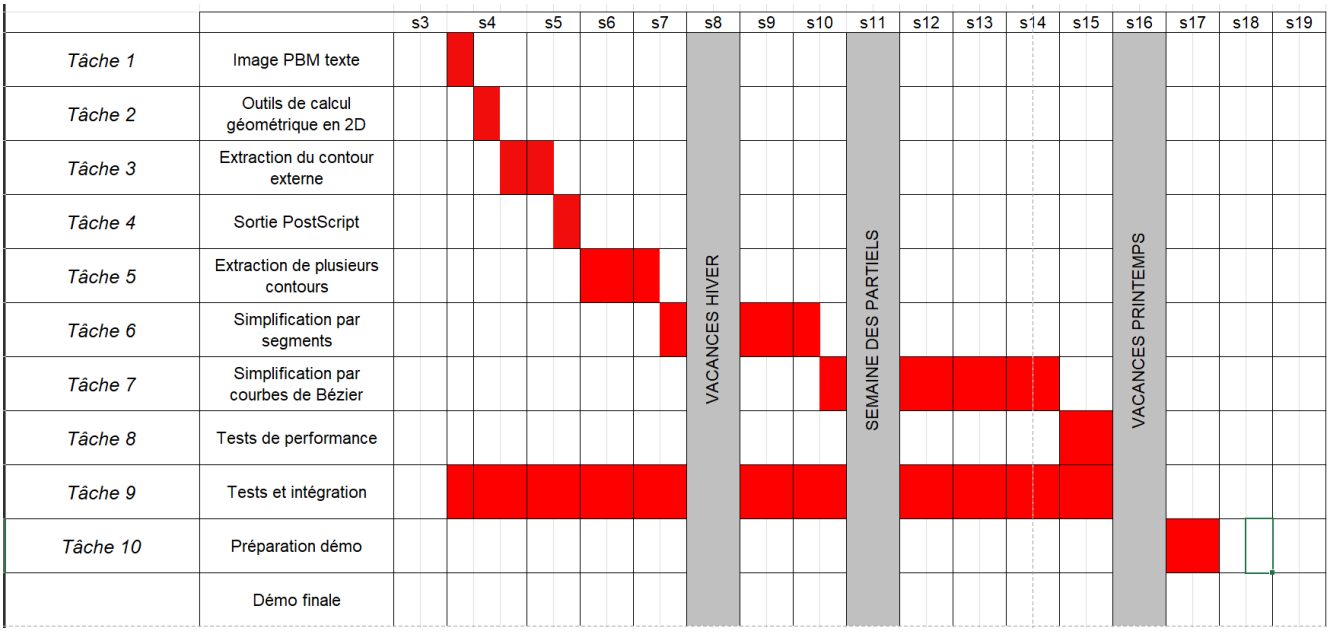
Tâche 8 : Optimisations

On a modifié l'algorithme de la première fonction du module Douglas Peucker afin de transformer la liste de contours en tableau avant l'appel récursif de cet algorithme. Cette modification nous a permis d'éviter la conversion de la liste chainée en tableau à chaque appel récursif.

Nous remarquons que plus la distance seuil augmente, plus le nombre de segments et de courbes de Bézier diminue. Cette observation est cohérente car une image simplifier doit contenir moins de détails et par conséquent, n'as pas besoin d'autant de segments et de courbes qu'une image plus détaillée. De plus, on remarque que les simplifications par segment entre les distances 2 et les distances 4 représente le point de rupture, c'est-à-dire que la qualité visuelle de l'image se dégrade fortement entre ces deux distances. Cependant, pour la simplification par courbe de Bézier 2 et 3, ce virage s'opère entre les distances 4 et 8. Il est donc cohérent d'utiliser une distance de simplification de 2 par segment et 4 par Bézier afin de préserver l'aspect visuel de l'image tout en diminuant l'espace prise par cette image pour une taille d'affichage similaire à celle-ci-dessus.

Enfin, nous remarquons que la simplification par segments est assez grossière et dégrade la qualité de l'image très rapidement. Lorsqu'on utilise des courbes de Bézier, cette dégradation de la qualité d'image est moins grande ce qui nous permet de simplifier plus les images. On remarque aussi que les courbes de Bézier de degrés 3 permettent d'obtenir une image simplifier de meilleure qualité, qui ressemble plus à celle initial, par rapport à une courbe de Bézier de degrés 2.

Diagramme de GANTT



Journal de bord

Réf.	Date	Problème/Information	Action/Décision	Date de réalisation prévue	Date de réalisation réelle	Etat
		TACHE 1				
1	25/01/22	Paquetage Image : écriture des fonctions écrire_image et negatif_image	Ecriture du fichier de test	25/01/22	25/01/22	terminé
2	25/01/22	Programme de test : OK		25/01/22	25/01/22	terminé
3	25/01/22	Test du paquetage Image	Passage à la Tache 2	25/01/22	25/01/22	terminé
		TACHE 2				
4	25/01/22	Ecriture du paquetage Geometrie2D : types Point et vecteur, fonctions addition, soustraction, multiplication par un scalaire, norme, distance	Réaliser le programme de test	25/01/22	25/01/22	terminé
5	25/01/22	Réalisation d'un programme de tests	Modifier le Makefile	25/01/22	25/01/22	terminé
6	25/01/22	Ecriture du paquetage Geometrie2D : norme, distance		25/01/22	25/01/22	terminé
7	25/01/22	Modification du Makefile	Passage à la Tâche 3	25/01/22	25/01/22	terminé
		TACHE 3.1				
8	25/01/22	Etape 1 complétée: création du paquetage contour : tourner a gauche/droite, changement d'orientation, trouver le pixel de départ, calcul du contour, avancer	Ecriture d'une fonction Points_egaux pour gagner du tem	25/01/22	25/01/22	terminé
9	25/01/22	Ajout d'un fonction Points_egaux.	création d'une image	25/01/22	25/01/22	terminé
10	25/1/22	Modification du Makefile, création images test pour tester à la volée		25/01/22	25/01/22	terminé
		TACHE 3.2				
12	01/02/22	Ajout des fonction permettant la gestion des listes chainés	transfert des coordonnées dans un fichier	01/02/22	01/02/22	terminé
13	01/02/22	Ecriture des fonctions permettant l'écriture dans coordonnées dans un fichier	Ecriture du programme de test	01/02/22	01/02/22	terminé
14	01/02/22	Ecriture du programme test_contour.c	Debogage du programme de test	01/02/22	01/02/22	terminé
15	01/02/22	Debogage du programme de test => segmentation Fault	Effectuer les tests	01/02/22	01/02/22	terminé
16	01/02/22	Test avec les images de la tâche 3 pour réalisation du fichier resultat-tache3-2.txt	Passage à la Tâche 4	01/02/22	01/02/22	terminé
		TACHE 4				
17	01/02/22	Création de la fonction d'écriture des fichiers image au format EPS	Réctifier les ordonnées (inverser du à l'écriture au format	01/02/22	01/02/22	terminé
18	01/02/22	Réctification des ordonnées		01/02/22	01/02/22	terminé
19	01/02/22	Récriture des fonctions dans le module image afin d'utiliser les accédeurs		01/02/22	01/02/22	terminé
20	01/02/22	Test sur les images indiquées dans le compte rendu	passage à la tâche 5	01/02/22	01/02/22	terminé

		TACHE 5				
21	01/02/22	création d'une fonction pour initialiser l'image de masque	A faire : création d'une séquence se contours	01/02/22	01/02/22	terminé
22	01/02/22	création d'une séquence se contours		01/02/22	08/02/22	terminé
23	08/02/22	Création de différents modules pour organiser le code		08/02/22	08/02/22	terminé
24	08/02/22	Création d'un module de liste chainée permettant de stocker une liste de contours		08/02/22	08/02/22	terminé
25	08/02/22	Modification du Makefile pour s'adapter aux différents modules créés		08/02/22	08/02/22	terminé
26	08/02/22	préparation de la fonction main pour la tâche 5.1	Finir main pour l'écriture du fichier EPS	08/02/22	08/02/22	terminé
27	15/2/22	Débogage fonction qui met à jour l'image de masque, création de la sortie EPS et débogage.	Réaliser comptes rendu tâches 5.1 et 5.2	15/2/22	15/2/22	terminé
28	15/2/22	Réalisation comptes rendu tâches 5.1 et 5.2	Commencer Tâche 6	15/2/22	15/2/22	terminé

		TACHE 6				
29	15/2/22	Création fonction distance point-segment		15/2/22	15/2/22	terminé
30	15/2/22	Cration programme pour afficher à l'écran la distance entre un point et un segment (définie par deux points)		15/2/22	15/2/22	terminé
31	15/2/22	Début Tâche 6.2		15/2/22	15/2/22	terminé
32	01/03/22	Implémentation de l'algorithme Douglas Peucker dans un module	main	01/03/22	01/03/22	terminé
33	01/03/22	Création du programme simplification de contours	erreur de segmentation dans la fonction douglas peucker	01/03/22	01/03/22	terminé
34	01/03/22	Modification du Makefile	Bug dans la fonction Douglas Peucker	01/03/22	01/03/22	terminé
35	08/03/22	Résolution des problèmes dans la fonction douglas Peucker → supprimer le premier élément lors de la concaténation des deux listes.	Problème dans le comptage du nombre de contours	08/03/22	08/03/22	terminé
36	08/03/22	Résolution du problème lié au comptage du nombre de contours → modification de la taille de la liste 2 avant la concaténation	Compte rendu	08/03/22	08/03/22	terminé
37	08/03/22	Test et élaboration du compte rendu	Passage à la Tache 7	08/03/22	08/03/22	terminé

		TACHE 7				
38	08/03/22	Lecture et compréhension de la tâche 7		08/03/22	08/03/22	En cours
39	22/03/22	Debogage Tache 6		22/03/22	22/03/22	terminé
40	22/03/22	création des types Bezier2 et Bezier3	Fonction pour créer des courbes et les initialisées	22/03/22	22/03/22	terminé
41	22/03/22	création de fonctions permettant d'initialiser des courbes de bézier de degrés 2 et 3.	écriture des fonctions d'approximation et de conversion	22/03/22	22/03/22	terminé
42	22/03/22	écriture des fonctions approx_bezier2, approx_bezier3, conversion_B2_B3	écriture de l'algo Douglas Peucker	22/03/22	22/03/22	terminé
43	22/03/22	Céation de l'algorithme Douglas Peucker pour les courbes de Bézier de degrés 2.	BUG lors de la compilation	22/03/22	22/03/22	terminé
44	29/03/22	Création d'un module de liste chainées bezier et liste contour bezier		29/03/22	29/03/22	terminé
45	29/03/22	Essaie de résoudre un segmentation fault dans la fonction douglas peuker bezier sans succès	Continuer le débug.....	29/03/22	29/03/22	terminé
46	06/04/2022	Douglas Peucker résolu, Rédaction du compte rendu et débuggage	Continuer le débug.....	06/04/2022	06/04/2022	terminé
47	12/04/2022	Fin du débug + Compte Rendu tâche 7 1 et 2	Mettre en forme le code et commencer le compte rendu g	12/04/2022	12/04/2022	terminé
48	12/04/2022	Mise en forme du code, création du manuel pour les tâches 5, 6 et 7, rédaction du compte rendu général, préparation soutenance	On continue on lâche rien	12/04/2022	12/04/2022	terminé

		TACHE 8				
49	12/04/2022	Création d'un script bash pour éxécuter les commandes de cette Tâche	Erreur "procéssus arrété"	12/04/2022	12/04/2022	terminé
50	12/04/2022	Modification de l'algorithme Douglas Peucker afin de transformer la liste chainée en tableau en dehors de la fonction. Ce qui permet une meilleure efficacitée.	Plus aucune erreur, tout fonctionne correctement	12/04/2022	12/04/2022	terminé
51	12/04/2022	Rédaction du compte rendu de la partie 1		12/04/2022	12/04/2022	terminé
52	12/04/2022	Rédaction du compte rendu de la partie 2 et comparaison des images obtenues				
		Rapport final				
53	Vacances	Création d'un script Bash permettant d'avoir une interface utilisateur.	CR final	Vacances	Vacances	terminé
54	Vacances	Rédaction du compte rendu final	Manuel Utilisateur	Vacances	Vacances	terminé
55	Vacances	Rédaction du manuel utilisateur	Commentaires	Vacances	Vacances	terminé
56	Vacances	Commentaires sur les codes sources qui n'en avaient pas	Entrainement soutenance	Vacances	Vacances	terminé
57	Vacances + 26/04	Entrainement à la soutenance		Vacances + 26/04	Vacances + 26/04	terminé