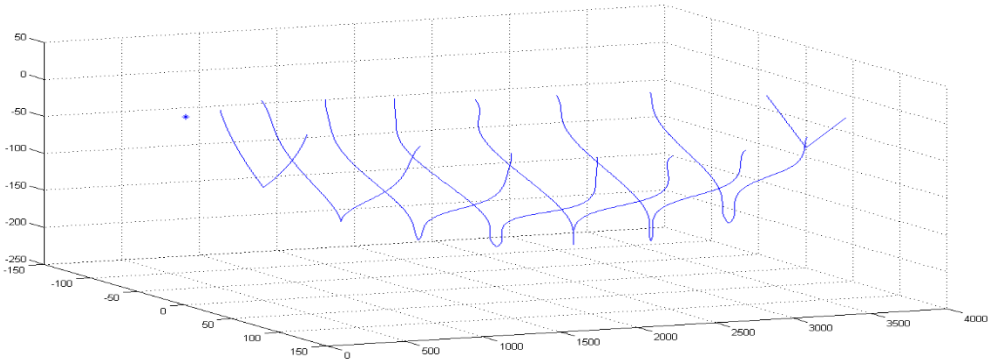
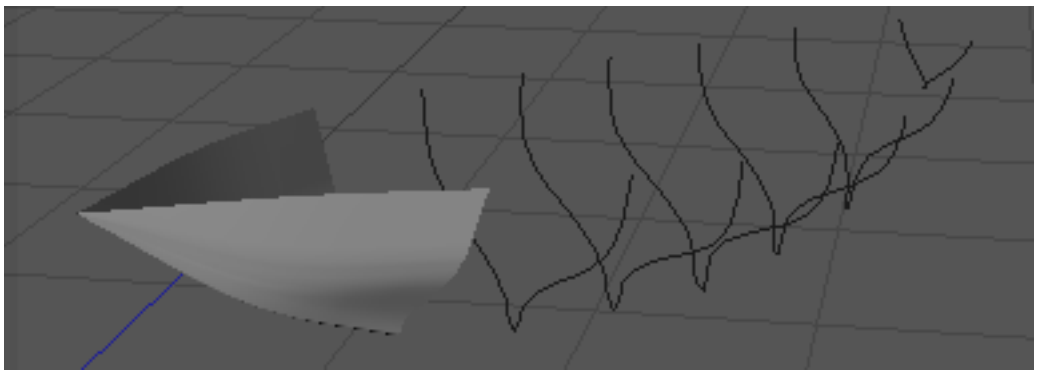
Rapport de Stage

Analyse de plan 2D de bateaux pour la génération d'une vue 3D





Réalisé par BIABIANY Emmanuel

L3 Maths – Informatique

Février 2014

Sommaire

Introduction

Ce rapport vient illustrer les différentes parties de mon stage qui s'est déroulé au LAMIA *(Laboratoire de Mathématique Informatique et Application)*, l'un des laboratoires présent sur le campus universitaire de Fouillole, en Guadeloupe. Il se trouve dans UFR Sciences Exactes et Naturelles.

Sous la tutelle de Mr Lionel Prévost et avec le soutien de Mr Vincent Page, tous deux enseignants chercheurs au sein du laboratoire, et spécialiste en traitement d'image et du signal. J'ai entrepris de répondre au mieux à la requête de Mr Jacques Laminie enseignants chercheur dans le domaine des mathématiques, et passionné par les voiliers. Etant un adepte de la construction navale, il a put m'apporter les informations essentielles à la réalisation de ce projet.

J'ai choisi ce sujet parce qu'à première vue, il demande un certain nombre de connaissance en graphisme et en modélisation 3D, et surtout parce que je pratique le graphisme 3D depuis plusieurs années.

L'objectif du projet est la génération relativement automatique, d'un modèle 3D représentant le plus fidèlement possible le plan papier d'origine.

Pour commencer, une analyse conceptuelle est de rigueur *(en gros trouver le plan d'attaque)*, ensuite une phase de traitement d'image *(par le biais d'algorithme et de l'utilisation des principes de morpho-maths)*, puis l'extraction des éléments importants ainsi que leur interprétation, et enfin pour terminer la génération du modèle en trois dimensions.

I. Analyse conceptuelle

Avant toute chose une petite mise au point sur l'ensemble des difficultés de ce projet, ainsi que sur le pseudo cahier des charges.
Pour commencer, la qualité des plans papier. Ils doivent être en bonne état et pour leur numérisation une résolution supérieure à 360p[1] garantira un rendu correct, et privilégie des formats d'images non compressés *(tel que le .tif)*.

L'utilisateur devra fournir à l'application les vues qui composent le plan séparément. On décide également de se concentrer sur la coque du bateau, car celle-ci est vrai semblablement ce qui caractérise le plus un navire. Après la récupération des vues, celle-ci suivent une série de traitement, et ensuite deux possibilités s'offrent à nous.

A partir de toutes ces vues, on utilise des logiciels de graphisme *(tel que CINEMA 4D, Illustrator, ou encore 3dsMax)* pour extraire le tracé des couples[2] et faire une génération 3D manuellement, nous garantissant une qualité esthétique optimale.

Ou alors, au moyen de l'outil Matlab on extrait les couples et utilise des fonctions mathématiques pour simuler leurs courbures et enfin grâce au maillage on génère un squelette du navire en 3D. L'intervention des mathématiques d'interpolations rend la tâche plus complexe.

J'ai commencé par faire la première option, les résultats étaient excellents et cela grâce à mes connaissances en modélisation 3D. J'ai donc écrit un rapport pour expliquer la marche à suivre *(Rapport – Synthèse 3D en Annexe).* Et cela à la fin de la première semaine de stage.

J'ai donc décidé de poursuivre en effectuant la deuxième option, c'est ce dont la suite du rapport fait l'objet.

[1] 360 pixels par pouce
[2] Eléments qui forment le squelette de la coque d'un navire.

II. Programme de traitement d'image

Toute la partie traitement d'image se fait sur Matlab. Car ce logiciel développe un langage interprété qui permet d'écrire des algorithmes, mais ce qui le rend intéressant, c'est sa capacité à manipuler des matrices de façon efficace. Et il semble évident que la matrice est l'une des structures les plus judicieuses pour représenter une image.
J'ai donc commencé par écrire des algorithmes pour nettoyer l'image, ensuite des algorithmes pour récupérer la valeur du maillage et le supprimer par la suite, et pour finir des algorithmes d'extraction et de séparation des couples.

A. Algorithme de nettoyage

Pour traiter les images plus rapidement j'en ai fait des images binaires[3], c'est-à-dire que celles-ci sont en noir et blanc *(non pas en niveau de gris)*, cela rend les opérations logiques possibles sur les images. Grâce ma fonction Binarise *(voir en Annexe).*

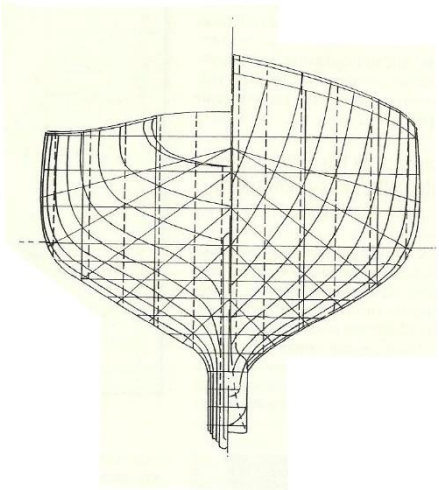


Figure 1 : Anahita, vue frontale, plan original

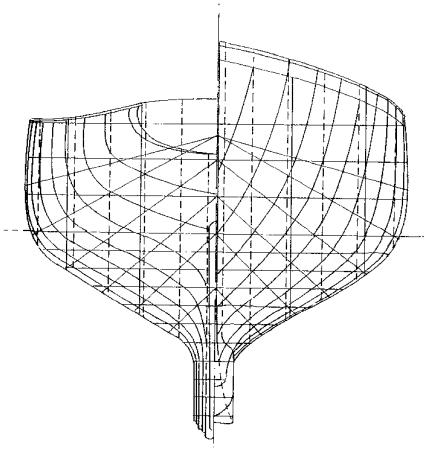


Figure 2 : Anahita, vue frontale, résultat binaire

[3] Les pixels prennent la valeur de 0 ou de 1.

Ensuite, interviennent les algorithmes de nettoyage, certaine image peuvent être en de mauvaise qualité, ou alors peuvent contenir du bruit. Et grâce à mon algorithme Filtrise *(voir en Annexe)*, qui selon les arguments passés reprend le principe du filtre médian[4], ou alors utilise la morpho-maths[5] et son principe d'ouverture et de fermeture ; et fait disparaître le bruit.

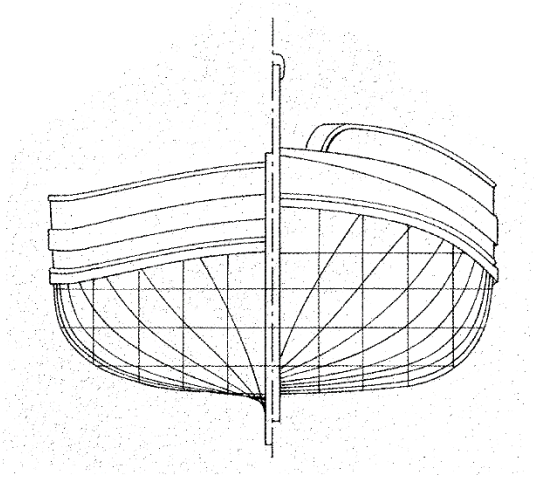


Figure 3 : Saint-Pierre, vue frontal, binaire

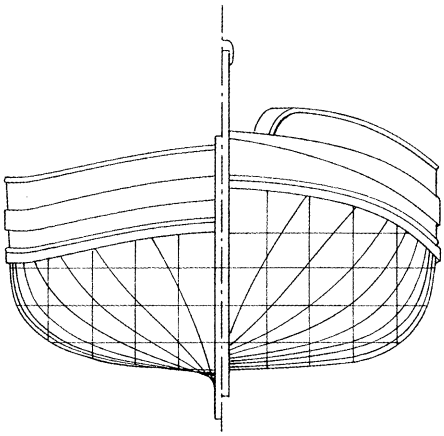


Figure 4 : Saint-Pierre, vue frontale, résultat filtré

B. Algorithme de maillage

Suite au nettoyage, on en vient au calcul de l'unité de maillage grâce à un autre algorithme qui parcoure l'image, repère le maillage et le schématise par un graphe, il suffit ensuite de déterminer la distance entre chaque pique du graphe, puis calculer la moyenne de ces écarts. Cette fonction s'appelle Segment *(voir en Annexe)*, c'est une fonction fourni par mon tuteur Mr Lionel Prévost.

Une fois l'unité relevée, le maillage devient dérangeant pour la suite des opérations, il faut donc l'éliminer. J'ai donc écris Demaillage *(voir Annexe)*, cette fonction élimine l'intégralité du maillage. En gros, celle-ci recherche les lignes et les colonnes de pixels les plus charger en pixels noir et les transforment en pixels blancs.

[4] Filtre qui récupère le voisinage du pixel courant et en déterminer le pixel médian.
[5] Algorithme qui utilise la convolution, et d'autre principe mathématiques.

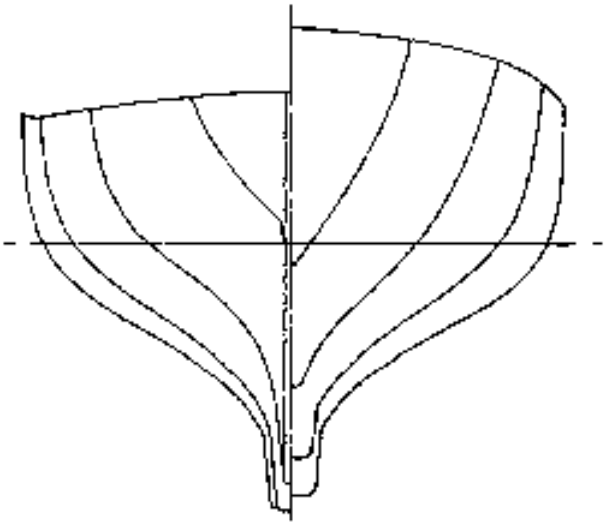


Figure 5 : Legh, vue frontale, résultat démaillé

Après le démaillage seul les lignes de flottaison et de séparation des vues restent sur l'image. On passe à l'étape suivante la séparation de vues.

C. Algorithme de séparation et d'extraction

Suite au démaillage une séparation est nécessaire sur la vue frontale. Car elle est composée d'une vue avant et d'une vue arrière. J'ai donc écrit un algorithme qui repère la jonction entre ces deux vues, et ensuite les séparent. Je l'ai appelé VfSeparator *(voir en Annexe)*, puis j'utilise CbExtarctor *(voir en Annexe)*, qui identifie les couples et élimine tous les traits qui ne sont pas des couples.

Figure 6 : Legh, vue dorsal, couples extraits

Ensuite il faut isoler les couples un par un, pour pouvoir faire une vectorisation de chacun d'entre eux. Pour les séparer, j'ai écrit une fonction c'est CbSeparator *(voir en Annexe)*, j'en ai fait une autre qui fait le même traitement mais sur les vues zénithale[6] et latérale, c'est Epluchise *(voir en Annexe)*, qui elle retire le contour de la vue au fur et à mesure, pour avoir toutes les couches qui la composent.

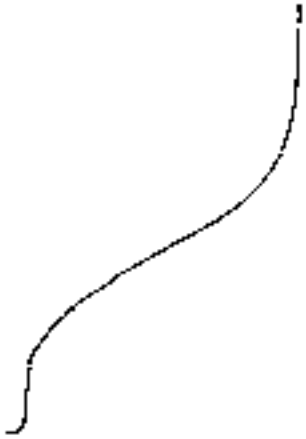


Figure 7 : Legh, vue frontal, couple 4

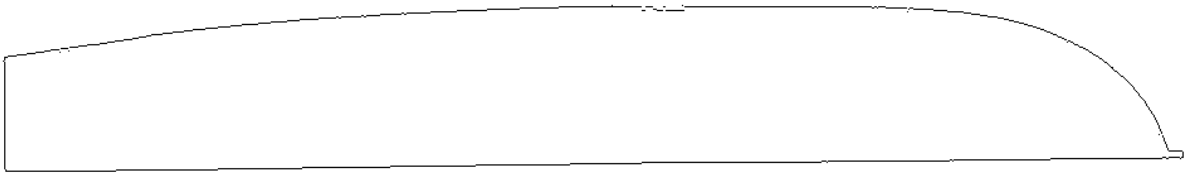


Figure 8 : Hermione, vue zénithale, contour 1

III. Modélisation graphique

La modélisation en trois dimensions, n'est possible que par la reproduction des couples à partir d'un échantillon *(plus précisément 1/10)*, de point de l'image que l'on envoie à une fonction polynomiale de degré aléatoire *(il est aussi possible d'utiliser une fonction de type spline)*. J'ai donc crée une fonction qui déterminer exactement

[6] Vue d'en dessous

le degré correspondant à la courbure et qui renvoie une interpolation du couple donné. Cette fonction s'appelle Vectorise *(voir en Annexe).* On obtient donc une courbe entièrement paramétrable.

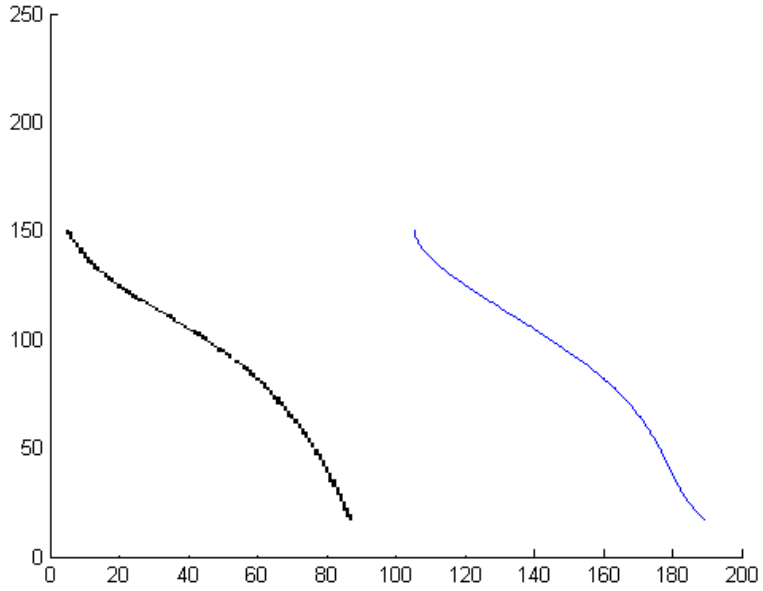


Figure 8 : Graphique de vectorisation, couple 2, interpoaltion de degré 3

La courbe en noir sur la figure 8, est l'ensemble des points pixels qui constitue le couple sur l'image et la courbe noir correspond à une fonction polynomiale de degré 3, formée par interpolation grâce à 1/10 des points la courbe noir. Le degré de la fonction est déterminé automatiquement grâce une boucle qui évalue l'erreur minimale.

Après avoir vectorisé tous les couples on utilise la fonction Dispose *(voir en Annexe)*, qui à partir de l'unité du maillage les replacent correctement dans un repère en trois dimension, que propose Matlab avec son plot3 *(voir en Annexe).* Cette fonction en appelle une autre, Symetrise *(voir en Annexe)*, qui comme l'indique recrée la partie complémentaire du couple. Ce sont toute les deux des fonctions que j'ai faites.

Pour finir, j'ai écrit un programme qui permet d'exécuter une génération 3D, à partir d'une vue frontale. Elle reprend l'intégralité des opérations faites jusque-là. Ce programme est entièrement automatique et ne nécessite que la sélection de l'image à traiter.

Cet algorithme se nomme ImVf *(voir en Annexe)*, et a un temps d'exécution inférieur à 0,9 secondes.

Il génère un modèle 3D, que l'on peut sauvegarder en .fig ou en .mat, à partir du menu de la fenêtre.

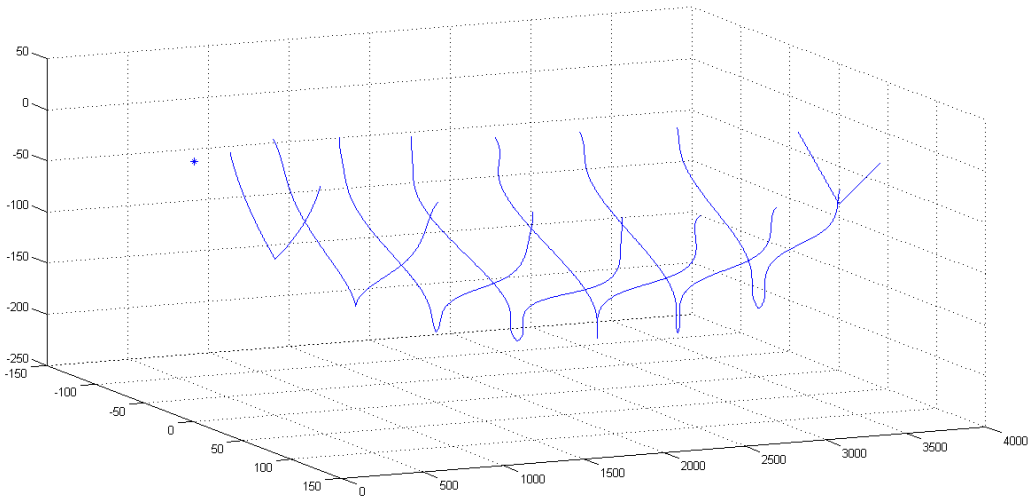


Figure 9 : Legh, Squelette 3D

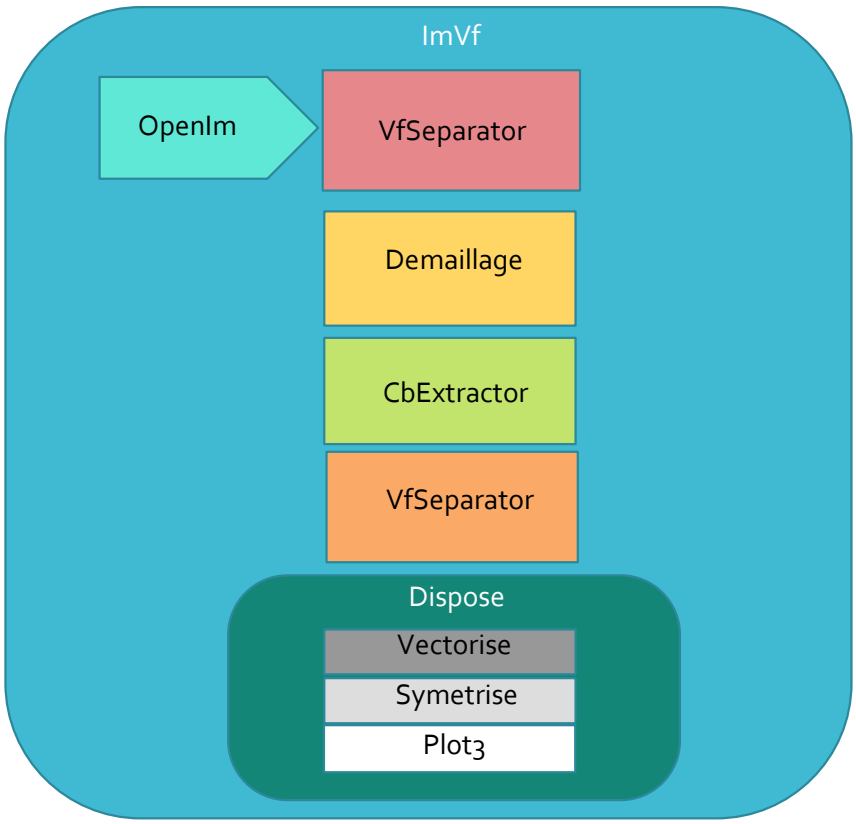


Figure 10 : Schéma fonctionnel

Conclusion

Ce stage a abouti sur des résultats plutôt concluant, car l'on a dépassé largement l'objectif fixé au début du stage.

De plus j'ai découvert le métier de chercheur en laboratoire, j'ai su faire preuve de beaucoup d'autonomie, et mon tuteur a toujours su me donner de bonnes astuces pour mon organisation ainsi que pour l'orientation de mes recherches.

Je compte continuer à travailler sur ce projet, en collaboration avec le Mr Lionel Prévost et Mr Jacques Laminie, afin d'améliorer ce programme.

A terme il faudra crée un plan normalisé afin de pouvoir extraire automatique les vues à partir du plan complet. Ensuite recoder ce programme en un autre langage, crée du rendu avec texture grâce au des bibliothèques graphiques tel que OpenGL, et pour finir développer ce programme en application web.

Je tiens à remercier Mr Lionel Prévost, Mr Vincent Page, Mr Jacques Laminie ainsi que mon collègue Mr Benjamin Zamour pour ce mois de stage qui s'est déroulé dans une atmosphère assez sympathique.

Annexe

Glossaire de fonction Matlab :

Binarise permet de binariser une image (N/B)

im = Bianarise(image,seuil,id)

Binarise l'image avec un seuil compris entre 0 et 1

id : 1 pour afficher le resultat

2 pour enregister en -bin

im = Binarise(image,id)

Binarise l'image avec un seuil de 0.5

id : 1 pour afficher le resultat

2 pour enregister en -bin

im = Binarise(image)

Binarise l'image avec un seuil de 0.5

See also VfSeparator, Symetrise, CbExtractor, CbSeparator etc...

By Emmanuel B.

CbExtractor Elimine tout le bord supérieur du plan afin de ne garder que

les courbes qui nous intéressent

im = CbExtractor(image,id)

Renvoie une image composé exclusivement de courbe c'est-à-dire

uniquement les demi-couples

id : 1 pour afficher le resultat

2 pour enregister en -bin

im = CbExtractor(image)

Renvoie une image composé exclusivement de courbe c'est-à-dire uniquement les demi-couples

See also CbSeparator, Contourise, Vectorise, etc..

By Emmanuel B.

CbSeparator

Contourise

Demaillage Permet d'éliminer le maillage d'une image binaire en déterminant la moyenne de chaque ligne et de chaque colonne, puis en supprimant celles ayant une moyenne superieur à au seuil

im = Demaillage(image,ratio,seuil,id)

Renvoie une image binaire sans maillage, a partir d'un ratio et d'un seuil donné

ratio : valeur comprise entre 0 et 1

Permet d'accentuer le démaillage sur partie inférieur de l'image, pratique avec une vue frontal

seuil : valeur comprise entre 0 et 1

Joue sur la tolérance du traitement, plus il est élevé plus le demaillage sera efficace

id : 1 pour afficher le resultat

2 pour enregister en -d

im = Demaillage(image,ratio,seuil)

Renvoie une image binaire sans maillage, a partir d'un ratio et

d'un seuil donné

im = Demaillage(image,ratio)

Renvoie une image binaire sans maillage, a partir d'un seuil donné

im = demaillage(image)

Renvoie une image binaire sans maillage

See also Squelettise, Binarise, VfSeparator, etc..

By Emmanuel B.

No help found for Dispose.m.

Epluchise

Filtrise Enleve le bruit sur une image

im = Filtrise(image,n)

Renvoie une image sans bruit

n : un entier naturel, correspond à la taille du voisinage traiter

See also flirtemed, etc..

By Emmanuel B.

No help found for ImVf.m.

No help found for Maillage.m.

mouse1.m (VERSION 3)

function [Xred, Yred,tabflagred] = mouseBase(action, titre);

Les coordonnees sont stockes automatiquement

dans des variables Xred et Yred. Donc attention aux conflits

de noms de variables.

X, Y - Cordonnees des points successifs

tabflagred - Vecteur de detection de levees

Exemple de lancement :

[X, Y, tab] = mouseBase('depart','Acquistion');

OpenIm Ouvre une fenêtre de dialogue pour recupérer une image

image = OpenIm;

Renvoie la matrise correspondant à l'image sélectionner

image = OpenIm('*.tif;*.jpg;*.png... ');

Renvoie la matrise correspondant à l'image sélectonner avec les extentions demander

See also SaveIm, uigetfile, uiputfile, etc...

By Emmanuel B.

SaveIm Ouvre une fenêtre de dialogue pour enregistrer un image

SaveIm(image);

Permet de sauvegarder une image dans le répertoire sélectionner, au format sélectionner

See also OpenIm, uigetfile, uiputfile, etc...

By Emmanuel B.

Squelettise Permet d'affiner une image binaire

im = Squelettise(image,id)

Renvoie une image affiner, grâce à la fonction bwmorph

otption 'thin' existant dans MATLAB

id : 1 pour afficher le resultat

2 pour enregister en -sqe

im = Squelettise(image)

Est possible également

See also bwmorph, Demaillage, Contourise, etc..

By Emmanuel B.

Symetrise Permet de reconstituer une vue

im = Symetrise(image,vue,id)

Renvoie une image constitué de l'original et de la symétrie axial

selon la vue souhaité

vue : vf-top parti avant de la vue frontal

vf-bottom parti arrière de la vue frontal

vz vue zenital

id : 1 pour afficher le resultat

2 pour enregister en -sym

im = Symetrise(image,vue)

Renvoie une image constitué de l'original et de la symétrie axial

selon la vue souhaité

vue : vf-top parti avant de la vue frontal

vf-bottom parti arrière de la vue frontal

vz vue zenital

See also fliplr, fliprl, etc..

By Emmanuel B.

Vectorise Permet de vectoriser l'image pixeliser d'une courbe

Vectorise(image,n)

Affiche la courbe polynomiale correspond à l'image entrer en

paramètre, de degré n

Vectorise(image)

Affiche la courbe polynomiale correspond à l'image entrer en paramètre

See also CbSeparator, etc..

By Emmanuel B.

VfSeparator Permet de séparer les deux parties compasant la vue frontale

[Bottom Top] = VfSeparator(im,id)

Renvoie vecteur composé de la vue arrière et de celle de l'avant

id : 1 pour afficher le resultat

2 pour enregister en -bin

[Bottom Top] = VfSeparator(im,id)

Renvoie vecteur composé de la vue arrière et de celle de l'avant

See also CbExtractor, CbSeparator, Epluchise, etc..

By Emmanuel B.

No help found for filtremed.m.

peakdet Detect peaks in a vector

[MAXTAB, MINTAB] = peakdet(V, DELTA) finds the local maxima and minima ("peaks") in the vector V.

MAXTAB and MINTAB consists of two columns. Column 1 contains indices in V, and column 2 the found values.

With [MAXTAB, MINTAB] = peakdet(V, DELTA, X) the indices in MAXTAB and MINTAB are replaced with the corresponding

X-values.

A point is considered a maximum peak if it has the maximal value, and was preceded (to the left) by a value lower by DELTA.

--- help for piecewisedistribution/segment ---

segment Segment of piecewise distribution containing input values.

S=segment(OBJ,X,P) returns an array S of integers indicating which segment of the piecewise distribution object OBJ contains each value in the array X or P. X and P cannot both be specified as non-empty. If X is not empty, the result S is determined by comparing X with the quantile boundary values defined for OBJ. If P is not empty, the result S is determined by comparing P with the probability boundary values.

See also piecewisedistribution, piecewisedistribution/boundary, piecewisedistribution/nsegments.

Reference page in Help browser

doc piecewisedistribution/segment