



3D object processing and Image processing by numerical methods

Abdul Rahman El Sayed

► To cite this version:

Abdul Rahman El Sayed. 3D object processing and Image processing by numerical methods. General Mathematics [math.GM]. Normandie Université, 2018. English. NNT : 2018NORMLH19 . tel-01951684

HAL Id: tel-01951684

<https://theses.hal.science/tel-01951684>

Submitted on 11 Dec 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THESE

Pour obtenir le diplôme de doctorat

Spécialité : Mathématiques Appliquées et Informatique

Préparée au sein de « Université Le Havre Normandie - LMAH »

Traitement des objets 3D et images par les méthodes numériques sur graphes

Présentée et soutenue par
Abdul Rahman EL SAYED

Thèse soutenue publiquement le 24 octobre 2018
devant le jury composé de

M. Cyril FONLUPT	Professeur, Université du Littoral - Côte d'Opale (ULCO)	Rapporteur
M. Abderrafiaa KOUKAM	Professeur, Université de Technologie de Belfort-Montbéliard	Rapporteur
Mme Marianne DE BOYSSON	Maître de Conférences, Université Le Havre Normandie	Examinatrice
M. Henri BASSON	Professeur, Université du Littoral - Côte d'Opale (ULCO)	Examineur
M. Abdallah EL CHAKIK	Maître de Conférences, Beirut Arab University, Lebanon	Examineur
M. Hassan ALABBOUD	Maître de Conférences, Université Libanaise, Tripoli – Liban	Co-encadreur de thèse
M. Adnan YASSINE	Professeur – Université Le Havre Normandie	Directeur de thèse

Thèse dirigée par Adnan YASSINE, Laboratoire de Mathématiques Appliquées du Havre (LMAH), Université Le Havre Normandie – France.

Etablissement



Ecole Doctorale



Laboratoire



Je dédie ce modeste travail :

À mes chers parents, pour leur soutien, leur patience et leur amour

À ma femme Nivine, pour sa patience

Remerciements

Cette thèse a été préparée au sein du Laboratoire de Mathématiques Appliquées du Havre, sous la direction du Professeur Adnan YASSINE, à l'Université Le Havre Normandie.

Je souhaite remercier en premier lieu mon directeur de thèse, Monsieur le Professeur Adnan YASSINE, pour m'avoir accueilli au sein de son équipe, pour ses encouragements, ses aides multiples et ses conseils pertinents. Je lui suis également reconnaissant pour le temps conséquent qu'il m'a accordé. J'ai beaucoup appris à ses côtés et je lui adresse ma gratitude pour tout cela.

J'adresse aussi mes remerciements à toutes les personnes qui m'ont soutenu lors de la réalisation de ce travail de thèse :

À Dr. Hassan AL ABOUD, Co-encadrant de cette thèse, pour l'attention qu'il m'a apportée durant ma thèse. Je lui suis très reconnaissant pour son aide, ses conseils, ses qualités humaines et ses encouragements. J'ai pris un grand plaisir à travailler avec lui.

À Dr. Abdallah EL CHAKIK, pour m'avoir guidé en me ramenant sur le juste chemin au cours de mes travaux. Ses commentaires et ses suggestions sont et seront toujours appréciés. Toute ma reconnaissance lui est adressée ici non seulement pour sa disponibilité, mais également pour la confiance qu'il m'a témoignée, pour son soutien et sa compréhension. Qu'il soit persuadé de mon profond respect.

Je tiens aussi à remercier le Directeur et tous les membres du Laboratoire de Mathématiques Appliquées du Havre.

Je présente mes remerciements les plus sincères à monsieur Cyril FONLUPT, Professeur à l'Université du Littoral - Côte d'Opale (ULCO), ainsi qu'à monsieur Abderrafiaa KOUKAM, Professeur à l'Université de Technologie de Belfort-Montbéliard (UTBM), pour avoir accepté de juger ce travail et d'en être rapporteurs.

J'adresse également mes sincères remerciements à monsieur Henri BASSON, Professeur à l'Université du Littoral - Côte d'Opale (ULCO), pour avoir accepté d'examiner mon travail.

Je ne remercierais jamais assez certaines personnes, mais je vais essayer : je remercie infiniment ma mère, mon père, ma femme Nivine, pour leur soutien constant, leurs encouragements, leurs sentiments et leurs prières.

Je remercie particulièrement mes sœurs, mes frères et mes chers amis...

Enfin un grand merci chaleureux à vous tous ...

Résumé

La détection de peau consiste à détecter les pixels correspondant à une peau humaine dans une image couleur. Les visages constituent une catégorie de stimulus importante par la richesse des informations qu'ils véhiculent car avant de reconnaître n'importe quelle personne il est indispensable de localiser et reconnaître son visage. La plupart des applications liées à la sécurité et à la biométrie reposent sur la détection de régions de peau telles que la détection de visages, le filtrage d'objets 3D pour adultes et la reconnaissance de gestes. En outre, la détection de la saillance des mailles 3D est une phase de prétraitement importante pour de nombreuses applications de vision par ordinateur. La segmentation d'objets 3D basée sur des régions saillantes a été largement utilisée dans de nombreuses applications de vision par ordinateur telles que la correspondance de formes 3D, les alignements d'objets, le lissage de nuages de points 3D, la recherche des images sur le web, l'indexation des images par le contenu, la segmentation de la vidéo et la détection et la reconnaissance de visages. La détection de peau est une tâche très difficile pour différentes raisons liées en général à la variabilité de la forme et la couleur à détecter (teintes différentes d'une personne à une autre, orientation et tailles quelconques, conditions d'éclairage) et surtout pour les images issues du web capturées sous différentes conditions de lumière. Il existe plusieurs approches connues pour la détection de peau : les approches basées sur la géométrie et l'extraction de traits caractéristiques, les approches basées sur le mouvement (la soustraction de l'arrière-plan (SAP), différence entre deux images consécutives, calcul du flot optique) et les approches basées sur la couleur. Dans cette thèse, nous proposons des méthodes d'optimisation numérique pour la détection de régions de couleurs de peaux et de régions saillantes sur des maillages 3D et des nuages de points 3D en utilisant un graphe pondéré. En se basant sur ces méthodes, nous proposons des approches de détection de visage 3D à l'aide de la programmation linéaire et de fouille de données (Data Mining). En outre, nous avons adapté nos méthodes proposées pour résoudre le problème de la simplification des nuages de points 3D et de la correspondance des objets 3D. En plus, nous montrons la robustesse et l'efficacité de nos méthodes proposées à travers de différents résultats expérimentaux réalisés. Enfin, nous montrons la stabilité et la robustesse de nos méthodes par rapport au bruit.

Abstract

Skin detection involves detecting pixels corresponding to human skin in a color image. The faces constitute a category of stimulus important by the wealth of information that they convey because before recognizing any person it is essential to locate and recognize his face. Most security and biometrics applications rely on the detection of skin regions such as face detection, 3D object filtering, and gesture recognition. In addition, saliency detection of 3D mesh is an important pretreatment phase for many computer vision applications. 3D segmentation based on salient regions has been widely used in many computer vision applications such as 3D shape matching, object alignments, 3D point-point smoothing, searching images on the web, image indexing by content, video segmentation and face detection and recognition. The detection of skin is a very difficult task for various reasons generally related to the variability of the shape and the color to be detected (different hues from one person to another, orientation and different sizes, lighting conditions) and especially for images from the web captured under different light conditions. There are several known approaches to skin detection: approaches based on geometry and feature extraction, motion-based approaches (background subtraction (SAP), difference between two consecutive images, optical flow calculation) and color-based approaches. In this thesis, we propose numerical optimization methods for the detection of skins color and salient regions on 3D meshes and 3D point clouds using a weighted graph. Based on these methods, we provide 3D face detection approaches using Linear Programming and Data Mining. In addition, we adapted our proposed methods to solve the problem of simplifying 3D point clouds and matching 3D objects. In addition, we show the robustness and efficiency of our proposed methods through different experimental results. Finally, we show the stability and robustness of our methods with respect to noise.

Table of Contents

1. State of the arts	5
1.1. Literature review	6
1.1.1. Skin region segmentation and face detection	6
1.1.2. Salient regions detection	8
1.1.3. 3D point cloud and mesh simplification	11
1.1.4. Matching and object recognition	12
1.2. Context and motivations	13
1.3. Conclusion	14
2. INTRODUCTION TO DATA MINING AND GRAPHS	15
2.1. Data mining and Knowledge Discovery	16
2.1.1. Introduction	16
2.1.2. Data mining process	16
2.1.3. Classification	17
2.1.3.1. Supervised classification	17
2.1.3.2. Unsupervised classification	18
2.1.3.3. Aggregation criterion	18
2.1.3.4. Evaluation of a classification system	18
2.1.3.5. Classification techniques	19
2.1.3.6. Classification by decision tree	19
2.1.4. Confusion matrix	21
2.1.5. Cross validation	22
2.1.6. Bootstrap	23
2.1.7. Data mining software: THE WEKA WORKBENCH	23
2.2. Graphs	23
2.2.1. Definitions and Notations	23
2.2.2. Graphs types	25
2.2.3. Sub graph	26
2.2.4. Graphs construction	26
2.2.4.1. Metric and Similarity	26
2.2.4.2. Similarity measurement	27
2.2.4.3. Unorganized domains	27
2.2.4.4. Organized domains	29
2.3. Conclusion	31
3. EFFICIENT 3D POINT CLOUDS CLASSIFICATION USING LINEAR PROGRAMMING AND DATA MINING	33
3.1. Introduction	34
3.2. Skin detection method	34
3.2.1. Learning dataset construction	35
3.2.2. Prediction rules generation	37
3.3. Skin Region Segmentation	39
3.3.1. Notations and definitions	39
3.4. Face detection	44
3.4.1. Refinement based on number of non-skin regions	44
3.4.2. Refinement based on size	45

3.4.3. Refinement based on the distance between face parts	46
3.5. Experimental results.....	47
3.5.1. 3D point clouds and 3D meshes	47
3.5.2. 2D images	52
3.6. Comparison with the state-of-the-art	54
3.7. Conclusion	55
4. EFFICIENT 3D MESH SALIENT REGIONS DETECTION	57
4.1. Introduction	58
4.2. Saliency method	60
4.2.1. Surface modeling and normal estimation.....	60
4.2.2. Saliency first approach.....	61
4.2.2.1. Deviation factor.....	62
4.2.2.2. Local patch descriptor.....	62
4.2.2.3. Saliency computation	63
4.2.2.4. Non-local Saliency computation	64
4.2.3. Saliency second approach	65
4.2.3.1. Invariant patch descriptor.....	65
4.2.3.2. Single-scale saliency	67
4.2.3.3. Multi scale saliency.....	67
4.3. Experimental results.....	68
4.3.1. Comparison with the ground truth	68
4.3.2. Influence of different parameters	70
4.4. Comparison with the state-of-the-art methods.....	73
4.5. Comparison between the proposed approaches	76
4.6. Conclusion	76
5. APPLICATIONS.....	79
5.1. 3D objects segmentation	80
5.1.1. Multi Search Hill Climbing – Algorithm.....	80
5.1.2. Heuristic algorithm	81
5.1.3. Experimental results.....	82
5.2. Feature points.....	83
5.2.1. Feature regions detection	83
5.2.2. Feature points detection	84
5.3. 3D point cloud simplification	85
5.3.1. Introduction	85
5.3.2. Contributions	87
5.3.3. Simplification process.....	87
5.3.4. Experimental results.....	88
5.3.5. Evaluation	91
5.4. 3D models matching	92
5.4.1. Template matching approach	92
5.4.2. Linear approach.....	93
5.4.2.1. 3D Object matching	94
5.4.2.2. Experimental results.....	95
5.4.2.3. Search experiments	98
5.4.2.4. Quantitative evaluation	99
5.4.2.5. Others applications.....	99

5.4.2.6. Comparison between the proposed approaches	100
5.5. Conclusion	100
6. 3D FACE DETECTION BASED ON GEOMETRICAL AND SKIN COLOR FEATURES ..	101
6.1. Introduction	102
6.2. Surface modelling and normal estimation	104
6.3. Proposed method	104
6.3.1. Notations and definitions	104
6.3.2. Candidate face region.....	104
6.3.3. Skin and saliency regions segmentation	104
6.3.4. Skin and saliency regions segmentation	105
6.3.5. Face detection method	106
6.4. Experimental results.....	107
6.4.1. 3D point clouds and 3D meshes	107
6.4.2. 2D images	113
6.5. Comparison with the state-of-the-art	115
6.6. Conclusion	115
General conclusion	119
Conclusion général	123
Bibliography	127

Figures

FIGURE 1.1 - Framework of skin-color based face detection proposed by [7]	6
FIGURE 1.2 - Detailed diagram of the face detection method proposed by [8]	7
FIGURE 1.3 - The framework for Hybrid Face Detection System proposed by [13]	8
FIGURE 1.4 - Synopsis of the mono-scale saliency proposed by [20].....	9
FIGURE 1.5 - Some results of related work compared with Ground truth.....	10
FIGURE 1.6 - 3D point cloud simplification	11
FIGURE 1.7 – General procedure of reverse engineering and inspection [36].....	12
FIGURE 1.8 - Matching planar shapes (a) and 3D shapes (b)	12
FIGURE 2.1 - Knowledge Discovery process :	16
FIGURE 2.2 - Example of a decision tree :	20
FIGURE 2.3 - Complete graph :	25
FIGURE 2.4 - Cycle graph :	25
FIGURE 2.5 - Sub graph :	26
FIGURE 2.6 - Graph construction on an unorganized domain :	28
FIGURE 2.7 – Delaunay graph construction :	29
FIGURE 2.8 - Construction of graphs on an image :	30
FIGURE 2.9 - Construction of a graph from a 3D mesh :	30
FIGURE 3.1 - Flowchart of our method :	34
FIGURE 3.2 - Flowchart of prediction rules generation :	35
FIGURE 3.3 - Extract from “Helen dataset” :	36
FIGURE 3.4 - Learning dataset creator :	36
FIGURE 3.5 - The Weka software :	37
FIGURE 3.6 - Vertices neighbors’ construction :	39
FIGURE 3.7 - Vertices classification :	40
FIGURE 3.8 - Influence of color information and similarity measure on vertices labelling process : ..	41
FIGURE 3.9 - Illustration of labelling process :	43
FIGURE 3.10 - Segmentation process :	44
FIGURE 3.11 - Skin region-containing gaps :	44
FIGURE 3.12 - Minimum bounding box of candidate region :	46
FIGURE 3.13 - Human face golden ratio :	46
FIGURE 3.14 - Distance between face parts :	46
FIGURE 3.15 - Skin detection on 3D colored point clouds :	47
FIGURE 3.16 - Influence of vertices neighbors on labelling process :	48
FIGURE 3.17 - Detected face on 3D colored point clouds with noise :	49
FIGURE 3.18 - Detected Faces with different expressions :	50
FIGURE 3.19 - Detected face on a noisy 3D colored point cloud :	50
FIGURE 3.20 - Influence of parameters NBFormsMax and <i>th</i> on accuracy :	51
FIGURE 3.21 - Detected faces on colored 2D images :	53
FIGURE 3.22 - Detected Faces from an image with different expressions :	53
FIGURE 3.23 - Accuracy comparison with other methods :	54
FIGURE 3.24 - Result obtained by [54] (a) and our method (b) :	55
FIGURE 4.1 - Saliency computation steps – first approach :	59
FIGURE 4.2 - Saliency computation steps – second approach :	60
FIGURE 4.3 - Illustration of the vertex neighboring construction :	61
FIGURE 4.4 - Illustration of the patch construction and vertices projection :	62
FIGURE 4.5 - Patch vertices :	63

FIGURE 4.6 - Local and Non-local saliency :	65
FIGURE 4.7 - Saliency first approach maps with ground truth maps :	69
FIGURE 4.8 - Saliency second approach map :	69
FIGURE 4.9 - Deviation factor threshold Influence :	70
FIGURE 4.10 - Saliency map with a Local and non-local saliency computation :	71
FIGURE 4.11 - Results of our method on 3D point clouds :	72
FIGURE 4.12 - Shows a comparison :	73
FIGURE 4.13 - Shows a comparison :	74
FIGURE 4.14 - Comparisons of our saliency map :	75
FIGURE 4.15 - Comparisons of our results (second row) with spectral mesh saliency :	76
FIGURE 5.1 - Multi Search Hill Climbing – Algorithm :	81
FIGURE 5.2 - 3D mesh segmentation based on saliency degree :	83
FIGURE 5.3 - Feature regions :	84
FIGURE 5.4 - Illustration of distribution of interest points :	84
FIGURE 5.5 - Illustration of feature region boundary :	85
FIGURE 5.6 - Boundary feature points :	85
FIGURE 5.7 - Flowchart of our method :	87
FIGURE 5.8 - Region gaps treatment :	88
FIGURE 5.9 - Contribution of the factor α :	89
FIGURE 5.10 - Contribution of deviation factor threshold :	90
FIGURE 5.11 - Simplification results :	91
FIGURE 5.12 - Dragon model: (a) our method, (b) other method :	92
FIGURE 5.13 - An isomorphism and a slate transition to a new isomorphism :	95
FIGURE 5.14 - Matching articulated shapes :	96
FIGURE 5.15 - Matching between articulated shapes :	97
FIGURE 5.16 - Matching between Gorilla mesh with many transformation :	97
FIGURE 5.17 - Matching detecting parts of 3D shapes :	97
FIGURE 6.1 - Flowchart of our method :	103
FIGURE 6.2 - Illustration of Multi Search Hill Climbing algorithm step :	106
FIGURE 6.3 - Examples of polytopes :	106
FIGURE 6.4 - Examples of Minkowski sum of two sets in two-dimensional real space :	107
FIGURE 6.5 - Results of applying our proposed model for face detection on some 3D colored point clouds :	110
FIGURE 6.6 - Candidates regions are identified by applying first phase of our proposed method :	110
FIGURE 6.7 - Two candidates regions detected by first phase and a single face recognized after applying second phase :	110
FIGURE 6.8 - Influence of parameter deviation factor threshold x : image a shows the salient regions detected with $x=12$ and image b shows the salient regions detected with $x=9$:	111
FIGURE 6.9 - Using our method with 3D point clouds and meshes containing multi faces :	112
FIGURE 6.10 - Using our method with 3D mesh not colored :	113
FIGURE 6.11 - Influence of x and k on face detection process :	113
FIGURE 6.12 - Using our method with 2D image :	114
FIGURE 6.13 - Using our method with 2D colored image containing multi faces :	114
FIGURE 6.14 - Using our method with 2D colored image containing face and has a complex background :	115
FIGURE 6.15 - Using our method with 2D colored image containing face of black skin :	115
FIGURE 6.16 - Result using our method applied on a colored image containing horizontal face :	115
FIGURE 6.17 - Image (a) shows a screen shot from experimental result :	117
FIGURE 6.18 - Face detection from a noisy 3D colored point cloud :	117

Tables

TABLE 2.1 - Confusion matrix for 2 classes A and B :	21
TABLE 3.1 - Information extraction from pixels database :	37
TABLE 3.2 - Testing results of J48 and JRip :	38
TABLE 3.3 - Results obtained from 3D point clouds and 3D meshes :	52
TABLE 3.4 - Results obtained from 2D image :	53
TABLE 5.1 - Geometric error measurement:	90
TABLE 5.2 - Contribution of deviation factor threshold :	90
TABLE 5.3 - Comparison with some related works :	91
TABLE 5.4 - Detecting parts of the object :	92
TABLE 5.5 - Results of different 3D models matching :	93
TABLE 5.6 - Retrieval results :	98
TABLE 5.7 - Quantitative evaluation of our proposed method in term of accuracy :	99
TABLE 6.1 - Results obtained from 3D point cloud :	113
TABLE 6.2 - Results obtained from 2D images :	115
TABLE 6.3 - Comparison of our method and others in terms of accuracy, FPR and FNR :	115

Equations

Equation 2.1	17
Equation 2.2	18
Equation 2.3	19
Equation 2.4	19
Equation 2.5	22
Equation 2.6	22
Equation 2.7	22
Equation 2.8	22
Equation 2.9	24
Equation 2.10	24
Equation 2.11	24
Equation 2.12	24
Equation 2.13	24
Equation 2.14	24
Equation 2.15	24
Equation 2.16	27
Equation 2.17	27
Equation 2.18	27
Equation 2.19	28
Equation 2.20	28
Equation 3.1	38
Equation 3.2	39
Equation 3.3	39
Equation 3.4	39
Equation 3.5	40
Equation 3.6	40
Equation 3.7	41
Equation 3.8	41
Equation 3.9	41
Equation 3.10	51
Equation 3.11	51
Equation 3.12	51
Equation 4.1	61
Equation 4.2	61
Equation 4.3	61
Equation 4.4	62
Equation 4.5	62
Equation 4.6	62
Equation 4.7	63
Equation 4.8	63
Equation 4.9	63
Equation 4.10	63
Equation 4.11	64
Equation 4.12	64
Equation 4.13	64
Equation 4.14	65

Equation 4.15	66
Equation 4.16	66
Equation 4.17	66
Equation 4.18	66
Equation 4.19	66
Equation 4.20	67
Equation 4.21	67
Equation 4.22	67
Equation 4.23	67
Equation 4.24	67
Equation 4.25	67
Equation 5.1	80
Equation 5.2	80
Equation 5.3	80
Equation 5.4	80
Equation 5.5	81
Equation 5.6	82
Equation 5.7	86
Equation 5.8	86
Equation 5.9	86
Equation 5.10	86
Equation 5.11	86
Equation 5.12	90
Equation 5.13	90
Equation 5.14	94
Equation 5.15	94
Equation 5.16	94
Equation 5.17	94
Equation 5.18	94
Equation 5.19	95
Equation 5.20	95
Equation 5.21	95
Equation 5.22	96
Equation 5.23	99
Equation 5.24	99
Equation 5.25	99
Equation 6.1	104
Equation 6.2	104
Equation 6.3	105
Equation 6.4	105
Equation 6.5	107
Equation 6.6	107
Equation 6.7	107
Equation 6.8	107
Equation 6.9	114

Introduction générale

La convergence entre l'informatique, Internet et l'audiovisuel conduit de plus en plus à des informations visuelles. Progressivement, beaucoup d'applications produisent, utilisent et partagent des données visuelles, incluant des images, vidéos et objets 3D. L'augmentation significative des informations visuelles sur Internet et dans les organisations s'est accompagnée d'une prise de connaissance de l'importance de développer des ressources informatiques pour traiter ces informations. Ce traitement signifie de filtrer, de modéliser, de classer, de rechercher et d'indexer cette quantité massive de données visuelles. Il est donc impératif de pouvoir classer ces données en fonction de leurs thèmes ou de leurs contenus. Cette classification permettra de faire une sélection ou un contrôle d'accès selon la sémantique et selon le type de ces données.

Dans ce contexte, la détection de régions de peau dans les données visuelles tels que les images, vidéo et objets 3D est importante dans la mesure où il est indispensable avant d'envisager des analyses et des traitements de niveau supérieur. Autrement dit, la plupart des applications liées à la sécurité et à la biométrie reposent sur la détection de régions de peau telles que détection de visage, filtrage d'objets 3D pour adultes et reconnaissance de gestes.

D'autre part, la détection des visages devient un domaine de recherche important pour les applications de vision par ordinateur telles que la reconnaissance et la vérification humaines, l'analyse des émotions pour les tâches multimédias et le suivi visuel. De nos jours, le développement de systèmes biométriques est devenu un défi important pour les chercheurs. Les empreintes digitales et les techniques d'iris sont les plus utilisées, mais la reconnaissance faciale semble être la meilleure approche surtout dans les aéroports et les zones critiques.

La plupart des approches de détection de visage pour les images 2D présentent de nombreux inconvénients tels que les variations d'éclairage, la pose et les expressions faciales. En outre, les données visuelles ne sont pas homogènes. Et comme la plupart de ces données peuvent être convertie en graphes pondérés. Par conséquent, pour rendre le processus de détection de visages applicable sur la plupart de données visuelles et moins sensible aux conditions d'éclairage et au point de vue, il est indispensable d'utiliser des méthodes qui fonctionnent sur les nuages de points colorés 3D représentés par de graphe pondéré.

Dans un contexte différent, la détection de la saillance sur les images ou bien maillages 3D a connu un progrès significatif. Les régions saillantes sont des zones particulières, qui sont distinguées des zones adjacentes sur une surface. Nombreuses des méthodes de détection de saillance ont été proposées pour 2D images. Cependant, les technologies d'acquisition de données 3D ont connu des développements significatifs, qui ont conduit à l'acquisition de grandes quantités de données sous la forme de maillages 3D et nuage des points 3D.

La détection de la saillance des maillages 3D est une phase de prétraitement importante pour de nombreuses applications de vision par ordinateur. La segmentation d'objets 3D basés sur des régions saillantes a été largement utilisée dans de nombreuses applications de vision par ordinateur telles que la correspondance de formes, les alignements d'objets, le lissage de nuages de points 3D et la segmentation et reconnaissance du visage. Parmi ces applications, on peut citer également la simplification du maillage 3D qui vise à maintenir une qualité mieux perçue en simplifiant les régions à faibles degrés de saillance.

Dans cette thèse, nous introduisons des méthodes numériques pour la détection de régions de peau et la détection de régions saillantes sur des images, maillages 3D et nuage de points 3D avec quelques applications directes.

Outre l'introduction et la conclusion, ce travail est organisé en 6 chapitres comme suit :

Le contexte et la motivation de notre travail, ainsi que la description de quelques travaux de recherche déjà réalisés et en relation à notre travail, sont présentés dans le chapitre 1.

Le deuxième chapitre est composé de deux parties : la première présente les différents concepts du Fouille de données, où on a décrit les différentes étapes d'un processus d'extraction de connaissances à partir des données. Tandis que la deuxième est consacrée, quant à elle, à la description de différents types de graphes avec leur construction et de différentes mesures de similarité.

Dans le troisième chapitre, nous proposons un modèle de détection de couleurs de peau dans de nuages de points colorés et nous décrivons la méthode de détection développée. Ensuite, nous étendons cette méthode pour résoudre le problème de la détection de visage 3D. Pour ce faire, nous construisons un graphe pondéré à partir des nuages de points 3D colorés initiaux. Puis, nous présentons un algorithme de programmation linéaire (LP) utilisant un modèle prédictif basé sur une approche d'exploration de données afin de classer et d'étiqueter les sommets de graphes en tant que régions de peau et de non-peau. De plus, nous appliquons des règles de raffinement sur les régions de peau pour confirmer la présence d'un visage.

Le quatrième chapitre, est consacré à la présentation d'une méthode de détections des régions de saillance basée sur les notions de graphes pondérés comportant deux approches différentes. En outre, nous montrons l'efficacité de nos approches à partir d'une étude comparative avec "Ground truth maps" et avec d'autres travaux déjà réalisés et en relation avec notre problème étudié. A la fin de ce chapitre, nous présentons différents résultats numériques expérimentaux pour confirmer la robustesse et l'efficacité de notre méthode proposée.

Le cinquième chapitre est réservé à la présentation de plusieurs applications basées sur les approches détaillées dans le quatrième chapitre. Parmi ces applications, on peut citer la segmentation d'objets 3D, la simplification du nuage de points 3D et la correspondance entre objets 3D.

Dans le sixième chapitre, nous présentons en détail une méthode hybride de détection de visages dans de nuages de points 3D en utilisant les approches présentées dans les troisième et quatrième chapitres.

Enfin, la dernière partie comporte une conclusion générale sur notre travail réalisé dans la thèse, quelques réflexions et des perspectives de nos travaux futurs.

General Introduction

The convergence between informatics, Internet and audiovisual sectors leads more and more to visual information. Gradually, many applications produce, use and share visual data, including images, videos and 3D objects. The significant increase in visual information on the Internet and in organizations has been accompanied by an awareness of the importance of developing computing resources to process this information. This treatment means to filter, model, classify, search and index this massive amount of visual data. It is therefore imperative to be able to classify this data according to their themes or their contents. This classification will make it possible to do a selection or an access control according to the semantics and according to the type of these data.

In this context, the detection of skin regions in visual data such as images, video and 3D objects is important as it is essential before considering higher level analyzes and treatments. In other words, most security and biometrics applications rely on the detection of skin regions such as face detection, 3D object filtering for adults and gesture recognition.

On the other hand, face detection is becoming an important area of research for computer vision applications such as human recognition and verification, emotion analysis for multimedia tasks and visual tracking. Nowadays, the development of biometric systems has become a major challenge for researchers. Fingerprints and iris techniques are the most used, but facial recognition seems to be the best approach especially in airports and critical areas.

Most face detection approaches for 2D images have many disadvantages such as lighting variations, exposure, and facial expressions. In addition, the visual data is not homogeneous. Moreover, since most of this data can be converted into weighted graphs. Therefore, to make the face detection process applicable to most visual data and less sensitive to lighting conditions and point of view, it is essential to use methods that work on the 3D colored point clouds represented by weighted graph.

In another context, the detection of saliency on images or 3D meshes has made significant progress. The salient regions are particular areas, which are distinguished from adjacent areas on a surface. Numerous methods of saliency detection have been proposed for 2D images. However, 3D data acquisition technologies have experienced significant developments, which have led to the acquisition of large amounts of data in the form of 3D meshes and 3D point cloud.

Saliency detection on 3D mesh is an important pretreatment phase for many computer vision applications. The segmentation of 3D objects based on salient regions has been widely used in many computer vision applications such as shape matching, object alignments, 3D point cloud smoothing, and face segmentation and recognition. Among these applications, one can also mention the simplification of the 3D mesh, which aims to maintain a better-perceived quality by simplifying the regions with low degrees of saliency.

In this thesis, we introduce numerical methods for skin regions detection and salient regions detection on images, 3D meshes and 3D point clouds with some applications.

In addition to the introduction and conclusion, this work is organized into six chapters as follows:

The context and motivation of our work, as well as the description of some related works, are presented in chapter 1.

The second chapter is composed of two parts: the first presents the different concepts of Data mining, where we described the different stages of a knowledge extraction process from the data. While the second one describes different types of graphs with the construction and different similarity measures.

In the third chapter, we propose a skin color detection model in colored point clouds and we describe the developed detection method. Then we extend this method to solve the problem of 3D face detection. To do this, we construct a weighted graph from the initial colored 3D point clouds. Then, we present a linear programming algorithm (LP) using a predictive model based on a data mining approach to classify and label the vertices of graph as skin and non-skin regions. In addition, we apply refinement rules on skin regions to confirm the presence of a face.

The fourth chapter presents a salient region detection method based on the concepts of weighted graphs with two different approaches. In addition, we show the effectiveness of our approaches from a comparative study with "Ground truth maps" and with other works already done and related to our studied problem. At the end of this chapter, we present various experimental results to show the robustness and efficiency of our proposed method.

The fifth chapter is dedicated to the presentation of several applications based on the approaches detailed in the fourth chapter. These applications include the segmentation of 3D objects, the simplification of the 3D point cloud and the matching between 3D objects.

In the sixth chapter, we present in detail a hybrid method of face detection in 3D point clouds using the approaches presented in the third and fourth chapters.

Finally, the last one presents a general conclusion of this thesis work and perspectives of our future work.

Chapter 1

SATE OF THE ARTS

Summary

1. State of the arts	5
1.1. Literature review	6
1.1.1. Skin region segmentation and face detection.....	6
1.1.2. Salient regions detection	8
1.1.3. 3D point cloud and mesh simplification	11
1.1.4. Matching and object recognition.....	12
1.2. Context and motivations	13
1.3. Conclusion	14

In this thesis, we are interested in image and 3D objects processing using numerical optimization methods. More specifically, we are working on two themes, which are the detection of skin regions and salient regions on 3D meshes and 3D point clouds. Moreover, we present many applications related to these two topics. In this chapter, we discuss some approaches related to our work by focusing on their disadvantages. These similar works include skin region segmentation, face detection, salient regions detections, 3D point clouds simplifications, 3D objects matching, and recognition. Finally, we finish this chapter by specifying the context and motivations, which pushed us to carry out this work.

1.1. Literature review

1.1.1. Skin region segmentation and face detection

In this subsection, we review some existing approaches for face detection and recognition based on skin region detection or geometry features.

Moudani et al. [5] presented a method based on data mining techniques for 2D face detection. They have demonstrated the robustness of the method by showing some experimental results and comparing them to other related works. In this work, we propose a generalized approach to deal with 3D colored point clouds.

Regarding skin detection, in [6], Joenes and Rehg used a method for skin detection based on Statistical Color Models in 2D images. Their method achieved a correct detection rate of 80%.

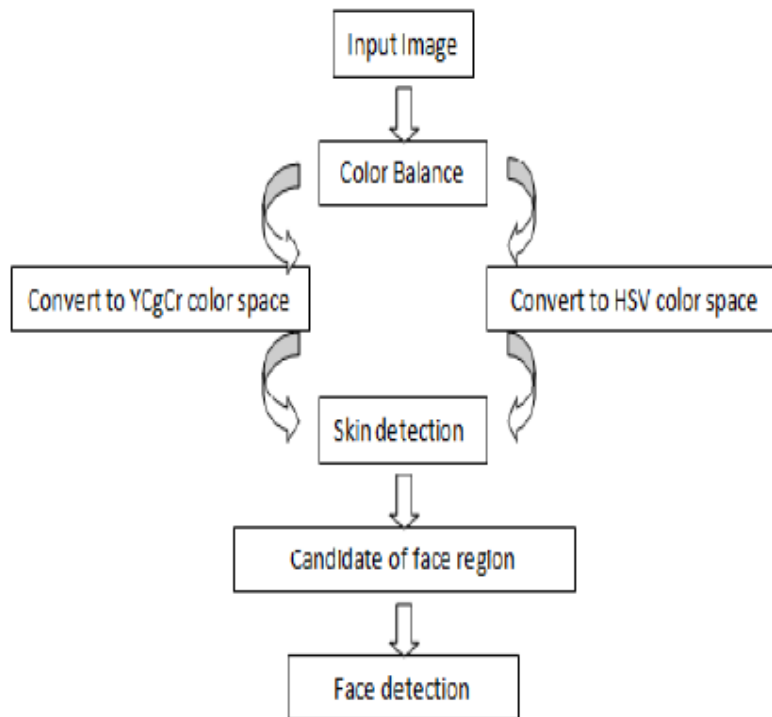


FIGURE 1.1 - Framework of skin-color based face detection proposed by [7]

Bin Ghazali et al. [7] proposed a method for 2D images face detection under varying illumination conditions. This method considers the color image and the color balance model to convert the RGB color space into the YCgCr color space. Then, they proposed to detect faces by merging the Gaussian skin-color model, template matching, and face verification technique.

Colombo et al. [8] proposed a method for 3D face detection that identifies the subject's eyes and the nose, and then classifies the regions of the eyes and noses. Nonetheless, the authors mentioned that this technique is highly sensitive to noises and to the presence of holes surrounding the regions of the nose and eyes.

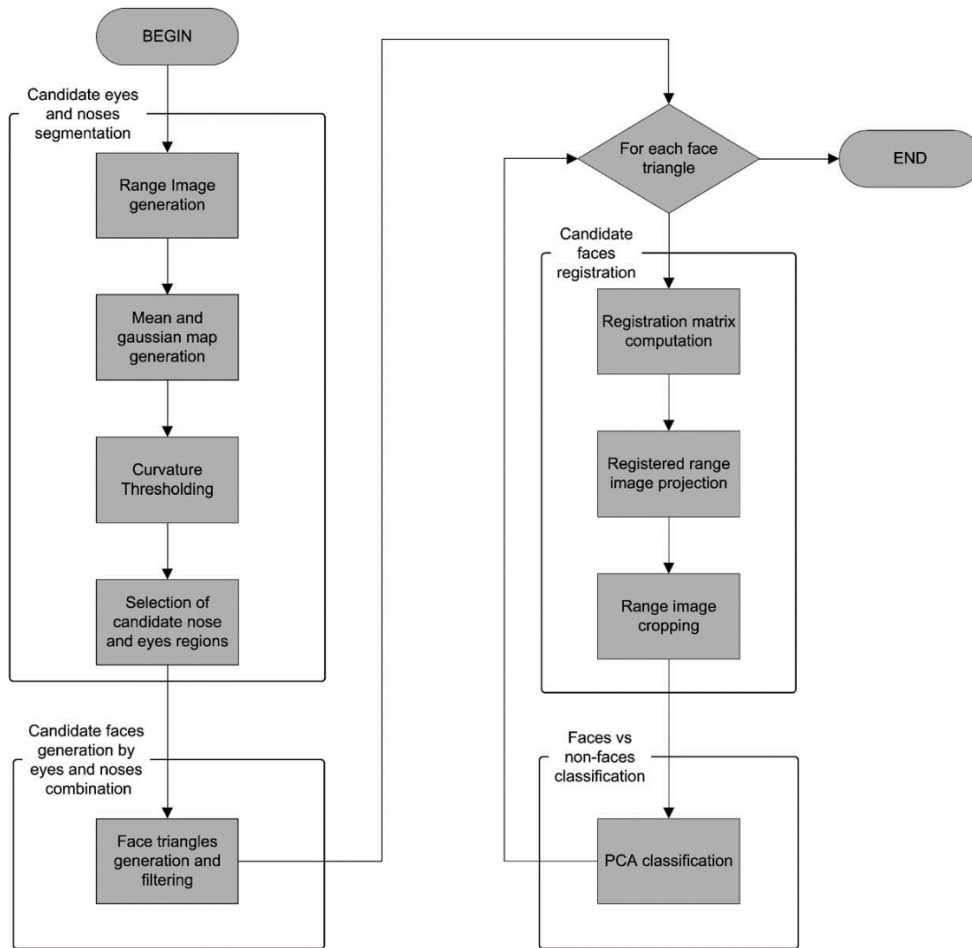


FIGURE 1.2 - Detailed diagram of the face detection method proposed by [8]

Mian et al. [9] proposed to localize the 3D face by detecting the nose tip, and then the segmentation can be made by cropping the sphere centered by the detected nose tip. They mentioned that their approach is strongly limited to the dataset used, as every input 3D object is inferred to contain only one face. Furthermore, the radius of the cropping sphere has a fixed value over the entire dataset and therefore the segmentation cannot handle scale changes. Niese et al. [10] proposed a method for 3D face detection based on color-driven clustering of the 3D points derived from stereo. They used a variant of the Iterative Closest Point algorithm to match the 3D mesh with the post-processed face cluster. Then, they used the correspondence to derive the face pose. Finally, pose and model information are used for the synthesis of the face normalization. Some methods [11, 12] divided the view sphere into a number of sectors and collected templates for each view. Given a new candidate object, the face detection is obtained using the template matching approach.

Kheirkhah et al. [13] presented an approach using Adaboost-based face detection based on skin-color information. *Adaboost is a known approach in machine learning to create a highly accurate prediction rules by combining many relatively weak and inaccurate rules.* They proposed to extract the regions first, which have more potential to be faces in the image using image illumination correction, skin tone extraction, skin region segmentation, and connected component separation. Then they apply some rules to eliminate non-face regions. Finally, they used the Adaboost-based face detection system as an improved appearance- based approach.

Lin et al. [14] proposed a human face detection method based on skin color and neural networks technique. First, they proposed to construct a so-called skin map by searching for the candidates' regions. Then, they applied some morphological operations to detect the locations of pair eyes in the candidates' regions. Finally, they affirm a presence of a face using a 3-layer back-propagation neural network to verify the face candidates' regions. Wang et al. [15] proposed a hierarchical face detection method using the template-matching and two dimensional PCA algorithms [79]. Their method consists in using two different classifiers. The first classifier filtrates the non-face regions using the template- matching algorithm, and then the second one uses the 2DPCA algorithm to detect faces.

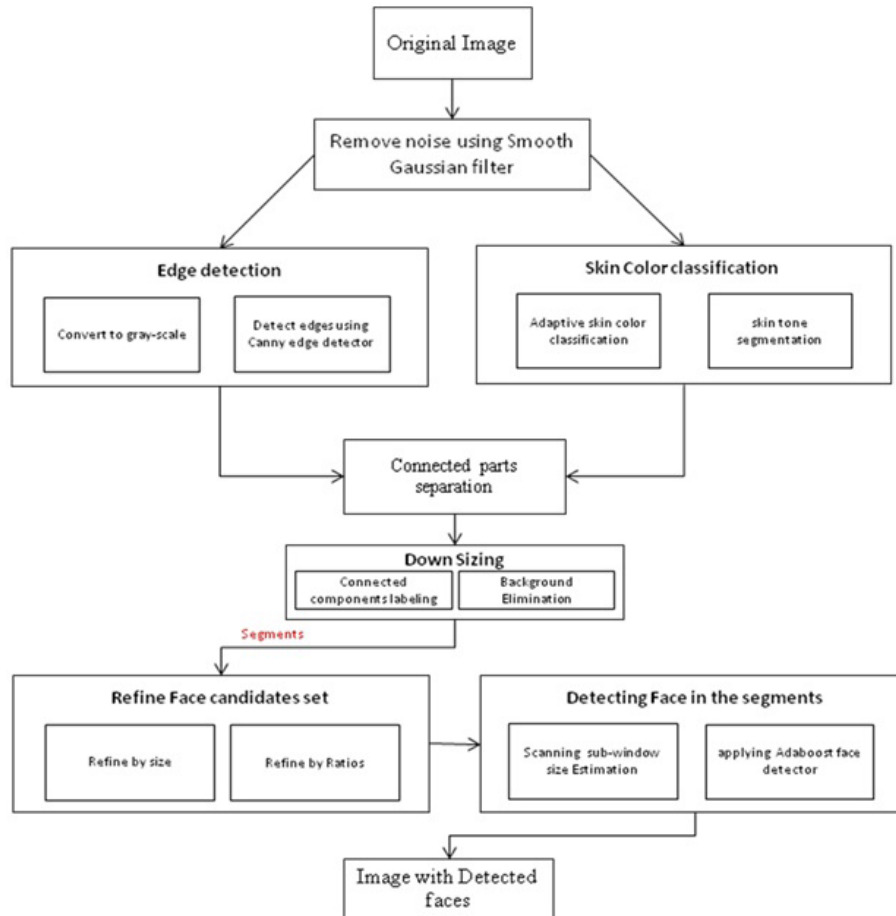


FIGURE 1.3- The framework for Hybrid Face Detection System proposed by [13]

Recently, Huang et al. [16] proposed to use a support vector machine model with skin and non-skin colors to detect skin points in 3D facial point clouds. Then, they have used an approach based on the k-nearest neighbor (k-NN) method to fill holes on the obtained skin point clouds. Finally, they proposed to use a threshold to determine whether a point is on the face surface.

1.1.2. Salient regions detection

In this subsection, we discuss some existing approaches for salient regions detection. Visual saliency is known as an important research field in computer vision history. It is used to locate the position of regions, which attract our attention in a 3D surface or a 2D image. Many saliency detection models have been proposed. Most of mesh saliency detection methods extend 2D saliency methods by computing the 3D surface saliency in its 2D projection or treat the 3D surface directly. These methods can be classified into two categories: point saliency detection and regions saliency detection.

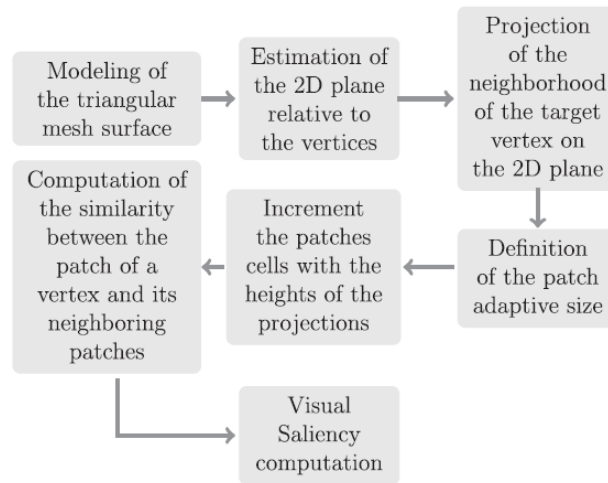


FIGURE 1.4- Synopsis of the mono-scale saliency proposed by [20]

Inspired from 2D mesh saliency methods, Lee et al. [17] introduced the idea of mesh saliency as a measure of regional importance for graphics meshes. They proposed to define the saliency in a scale-dependent manner using a center-surround operator on Gaussian-weighted mean curvatures. They used a nonlinear weighting scheme to promote few distinctive maps with high peaks. The main disadvantage of this method is that the Gaussian-weighted difference of different scales can produce the same values of saliency for two opposite and symmetric vertices due to the absolute difference. Song et al. [18] estimate the salience in the spectral domain by analyzing deviations and characteristics of the log-Laplacian spectrum. To compute the saliency map, they transformed back the spectral residual in spectral domain to the special domain. Spectral analysis is known as an efficient approach for image saliency detection.

Some saliency detection algorithms compute the 3D mesh saliency by projecting the set of 3D mesh node in a 2D plan. Wu et al. [19] proposed a method to detect saliency on 3D meshes using the local contrast and global rarity. They calculated square map of the height projections to describe the local shape surrounding a vertex. Then, local and global saliencies are combined together for each vertex to define the final visual saliency at different scales. A. Nouri et al. in [20] have proposed a method to detect saliency on 3D meshes using a multi-scale approach. They defined a local surface descriptor using a patch with an adaptive size. Subsequently, the adaptive patch is divided into cells where each cell contains the absolute value of the sum of projections heights. Then, they defined a single-scale saliency for each vertex using the similarity between adaptive patches and the local curvature. Finally, they proposed to compute the multi-scale saliency as the average of single-scale saliencies weighted by their entropies. Leifman et al. proposed in [21] to detect saliency using a diffusion process. They built for each vertex a 2D histogram of spin images, and then they measured the dissimilarity between these histograms by discretizing the diffusion using a Gaussian pyramid. Then they used a single scale computation to evaluate the distinctness of vertices and a multiple scale computation to decrease the importance of 3D textures. Finally, they selected 20% of vertices with highest distinctness value as focus points. It is true that close regions to focus points have more attention than other regions, but other salient regions where the distinctness of all vertices is less than 20% is missed. The multi-scale operation used in this method may not reduce all the noise effects.

Some earlier methods compute the 3D mesh saliency by treating the surface directly. In [22], Tao et al. proposed a method to detect 3D salient regions based on manifold ranking in a descriptor space. Firstly, they segmented the mesh into patches and computed a descriptor for each patch based on Zernike coefficients. Then, they estimated the local distinctness of each patch by a center-surround operator.

Patches with small named background and those with high local distinctness are named foreground patches. Furthermore, the saliency of all the patches is computed based on their relevance to the given queries via manifold ranking using a self-adaption graph in the descriptor space of patches. Finally, they achieve a smooth vertex saliency map by propagating the patch saliency. This method is high robust against noises, but the authors mentioned that it does not incorporate any high-level priors. Zhao. et al [23, 24] proposed to performed a filtering phase to remove high frequencies from the mesh and compute similarities between vertices. Then, they transformed the mesh into a multi-scale volumetric data. The scale of volumetric data is set by the user, thus the patch is located within the voxel. The dissimilarity map is then computed using the dissimilarity measure between two patches associated to two sub-voxels. Finally, the saliency of a single patch is calculated from the average of saliency maps with different scales. The mesh enhancement phase leads to locate meaningful region completely, while the disadvantage of this method is the computational time complexity due to the volumetric data construction and the intensive computation of all patches comparison. Recently, Zhao et al. [25] suggested to detect interest points by measuring the saliency degree of vertices. They implemented the Retinex theory [26] to improve local details of the surface and estimate its invariant properties. Finally, they segmented the 3D surface to regions, and then they proposed to estimate the saliency degree of vertices using the spatial distance between the obtained regions. Liu et al [27] proposed to detect the saliency on a 3D mesh using absorbing Markov chain. They have segmented the mesh into patches in order to select some background patches by computing feature variance within the segments. Furthermore, they computed the absorbed time of each node using absorbing Markov chain with the background patches as absorbing nodes, which give a saliency measure. Then, they generated a refined saliency measure using the similar approach but with foreground nodes extracted from the saliency map. Finally, they applied a Laplacian-based smoothing procedure to propagate the patch saliency to vertices.

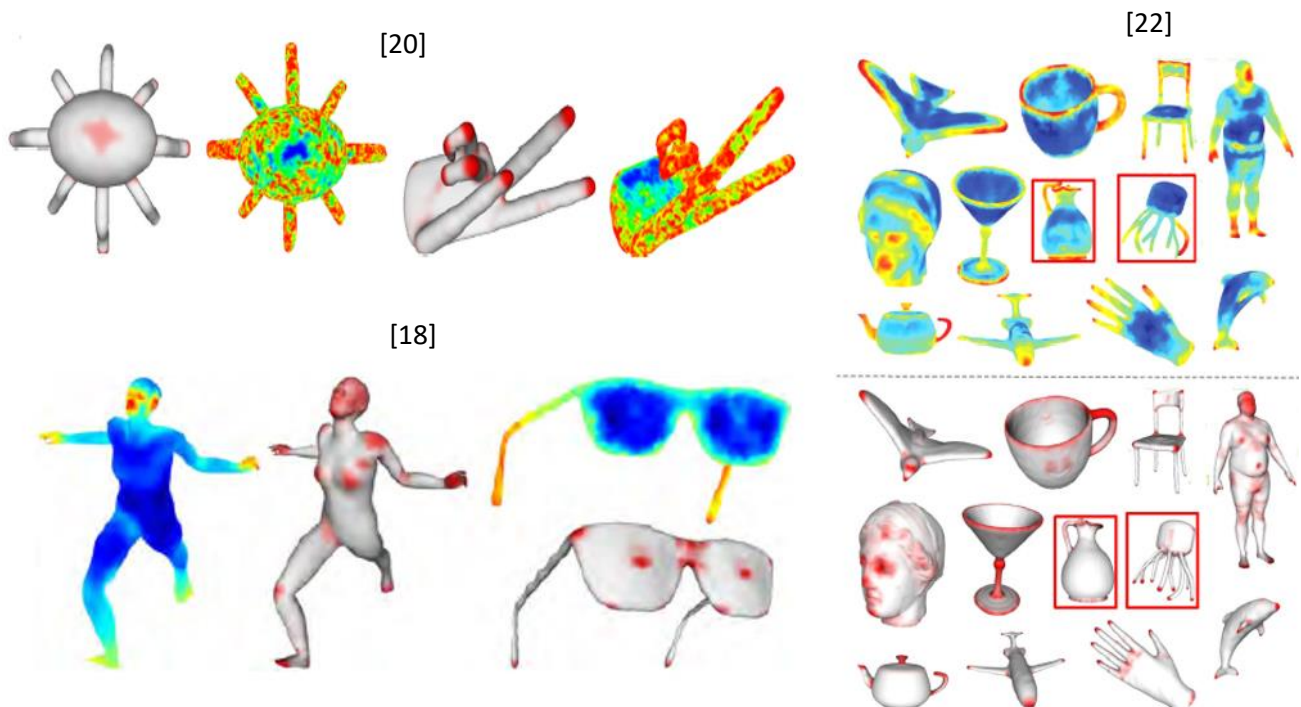


FIGURE 1.5- Some results of related work compared with Ground truth

1.1.3. 3D point cloud and mesh simplification

In this subsection, we review available approaches for 3D point cloud and mesh simplification by mention their defects.

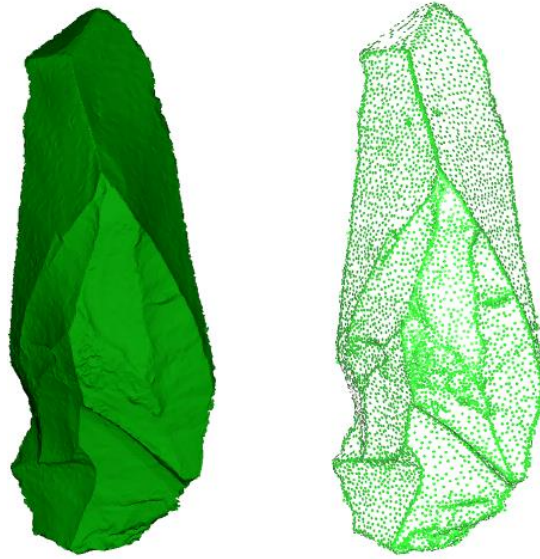


FIGURE 1.6- 3D point cloud simplification

Lee and al. [28] introduced a simplification method based on the part geometry information and normal values of points using 3D grids. Pauly et al. [29] presented and studied different approaches for surface simplification of 3D objects from unstructured point clouds. In [30, 31], Moaning and Dodgson used the idea of progressive intrinsic farthest point sampling of a surface in point clouds. They presented a uniform simplification algorithm coarse-to-fine with user-controlled density guarantee.

In order to find the weight of 3D models features, lee et al. [32] suggested a novel simplification method by adopting the Discrete Shape Operator.

Based on the feature extraction, Peng et al. [33] proposed a new simplification algorithm for unstructured point clouds described by its unit normal vectors.

In [34], Song et al. proposed a global clustering simplification method for point cloud. It consists of finding a subset of the original input data set according to a specified data reduction ratio. Then, they obtained a global optimal result by minimizing the geometric deviation between the input point sets and the simplified ones. The drawback of this method is that the approximated point-to-surface distances may no longer get accurate values, and it is hugely time-consuming when the number of points is much reduced.

In [35], Miao et al. proposed a curvature-aware adaptive re-sampling method to simplify point clouds based on an adaptive mean-shift clustering pattern. The usage of adaptive mean-shift clustering was to generate a simplification result non-uniformly distributed. However, it is difficult to incorporate the simplified geometric error in their approach and the simplification rate is low.

Lastly, in [36], Shi et al. presented a new method for simplification “cluster subdivision” based on k-means clustering approach according to two factors: user-defined space interval and normal vector tolerance. Thus, this method is very sensitive to the user-defined parameters.

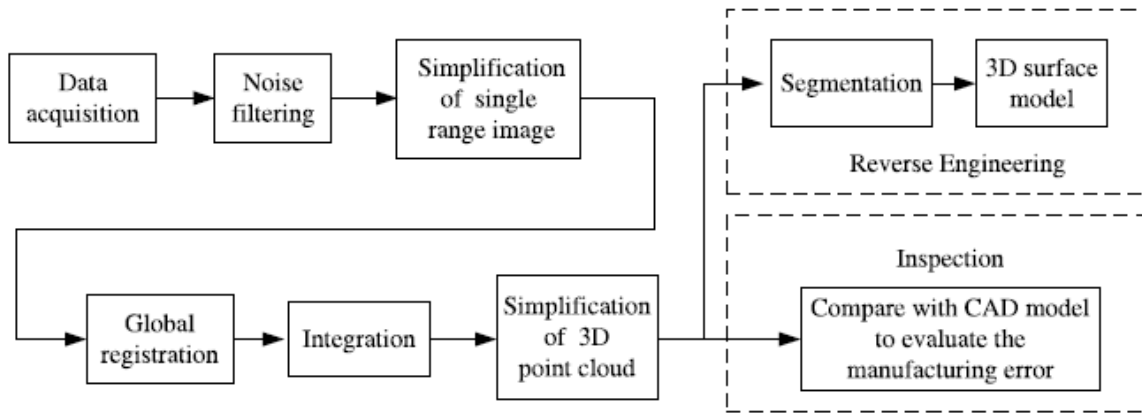


FIGURE 1.7- General procedure of reverse engineering and inspection [36]

1.1.4. Matching and object recognition

In this subsection, we present existing approaches for 3D objects matching and recognition.

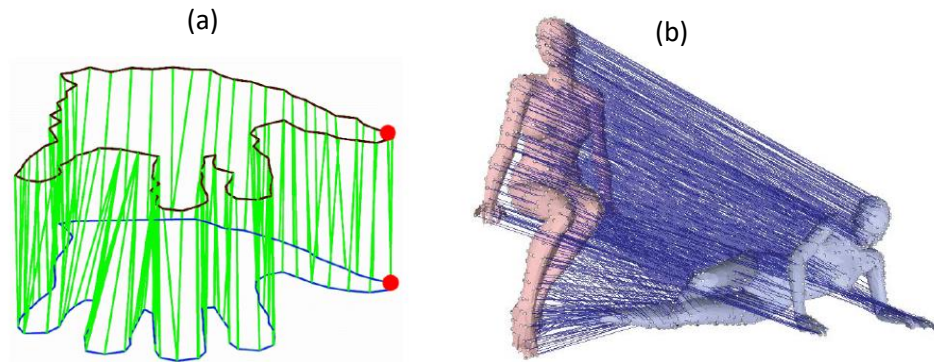


FIGURE 1.8- Matching planar shapes (a) and 3D shapes (b)

The matching between planar shapes can be solved using dynamic programming where the complete search over the initial correspondence leads to cubic runtime in the number of sample points [37]. In contrast, the concepts of Dijkstra's algorithm and dynamic programs were not extended to 3D dimension where the solution is a minimal closed surface in a space of higher dimension. Thus, the most of existing approaches for 3D object matching usually rely on local optimization techniques. In this scope, our approach tries to find a relational maximum sub-graph matching between featured regions representing the 3D objects. Authors in [38] proposed a method for computing a geometrical matching between two 3D shapes by mapping tiny surface patches while preserving the geometric structures instead of matching points to points. They considered matching as diffeomorphisms between the objects' surfaces, which can be represented as closed and continuous surfaces in the product space of the two shapes. This leads to a minimal problem with discrete formulation that describes the search space with linear constraints. Authors in [39] proposed the paradigm of the Gromov–Hausdorff framework to compute the correspondence that minimizes the geodesic distortion. In [40], authors proposed an approach for calculating correspondences in a coarse-to-fine strategy similarly as of optical flow approaches. Other approaches of shape matching rely on techniques such as conformal [41] or Riemannian [42] geometries. All these methods minimize a nonconvex energy using a local optimization technique.

Thus, the solution quality depends deeply on a suitable initialization and a coarse-to-fine strategy well designed. Moreover, the solution do not come with any guarantee of optimality.

Recently, Zeng and coworkers [43] proposed a method for computing the shape matching by formulating it as a graph-matching problem then applying the QPBO algorithm [44]. The major disadvantages of this approach are the very high computational complexity and lacks of a continuous counterpart, as it just matches discrete points rather than surface patches.

1.2. Context and motivations

In this manuscript, we decide to work on two themes **Skin color** and **Saliency regions detection** due to their important applications in computer vision.

Skin and face detection is a trivial task for a human brain but a very complex one for computers. There are many difficulties encountered to implement such approaches as:

- **Skin color:** Human beings have different skin colors, hence the difference in the pixel value of each person's skin.
- **Illumination conditions:** In any detection action, light is an important factor and it is the most difficult problem to solve. Hence the need for image preprocessing such as normalization and histogram equalization to minimize illumination and illumination effects.
- **Face position:** A face can be located in different positions in an image, 3D objects or 3D point clouds; in this case, the computer must be able to detect the face whatever its position.
- **Face size:** The size of faces is different from one image to another or from one person to another hence the difficulty of implementing an algorithm that detects faces without having consequences of this factor. There is also the size of the components of the face such as the nose, the eyes, the mouth or something else varying from one person to another which implies a larger number of parameters when performing the detection.
- **Absence of some facial components:** These are whiskers, beards, glasses that must be taken into consideration when implementing the algorithm.
- **Occultation:** A face that can appear half-in-picture or sometimes partially obscured by an object forces us to define the conditions of acceptance of the face by the system. For example, it can be assumed that the face must appear entirely for it to be admitted.
- **Rotation:** Faces are not always facing the camera. Some faces have an inclination with a certain degree, which affects the detection system hence the need to establish conditions to define a candidate face for acceptance.
- **Image or 3D objects complexity:** The detection can be on very complex images with several people in the same image, faces hidden or half hidden by objects with possibly complex backgrounds, which increases the difficulty of detection.

Concerning salient regions detection and its applications, the difficulties faced are:

- **Noise problem**
- **The computational time complexity**
- **Invariant surface patch descriptor**

- **Scale varying**

In addition, due to disadvantages of related works presented in the previous section, our objective in this thesis is to propose accurate and robust methods that deal with any data represented by weighted graph such as 3D colored point clouds, 3D meshes and 2D images.

1.3. Conclusion

In this chapter, we have discussed several related works by mention their defects. Successively, we showed the motivations and context of our work. In the next chapter, we will briefly introduce Data mining and Graph as we used these approaches in this thesis.

Chapter 2

INTRODUCTION TO DATA MINING AND GRAPHS

Summary

2. INTRODUCTION TO DATA MINING AND GRAPHS	15
2.1. Data mining and Knowledge Discovery	16
2.1.1. Introduction	16
2.1.2. Data mining process	16
2.1.3. Classification	17
2.1.3.1. Supervised classification	17
2.1.3.2. Unsupervised classification	18
2.1.3.3. Aggregation criterion	18
2.1.3.4. Evaluation of a classification system	18
2.1.3.5. Classification techniques	19
2.1.3.6. Classification by decision tree	19
2.1.4. Confusion matrix	21
2.1.5. Cross validation	22
2.1.6. Bootstrap	23
2.1.7. Data mining software: THE WEKA WORKBENCH	23
2.2. Graphs	23
2.2.1. Definitions and Notations	23
2.2.2. Graphs types	24
2.2.3. Sub graph	25
2.2.4. Graphs construction	26
2.2.4.1. Metric and Similarity	26
2.2.4.2. Similarity measurement	27
2.2.4.3. Unorganized domains	27
2.2.4.4. Organized domains	28
2.3. Conclusion	31

In this chapter, we present the process of knowledge discovery (KDD) by describing Data mining and its associated steps. The different types of classification are presented while being limited to describe the technique of decision tree with its different variants. Furthermore, all notations and works presented in this thesis are based on the concept of weighted graph. Therefore, we introduce in this chapter the different definitions, notations and concepts inherent to these graphs.

2.1. Data mining and Knowledge Discovery

2.1.1. Introduction

The progress of Information Technology has generated massive data in various areas. These data can be stored in data warehouses, distributed databases or on the Internet. The research in databases and information technology fields has given rise to an approach to store and process this precious data for further decision making.

Data mining is the core concept of the knowledge extraction process (FIGURE 2.1.) that allows accessing and analyzing data. In other words, Data mining is the art of extracting knowledge from data. It is not limited to processing structured digitalized; it offers the tools to approach the corpus in natural language (text mining), images (image mining), sound (sound mining) or videos and more generally the multimedia mining.

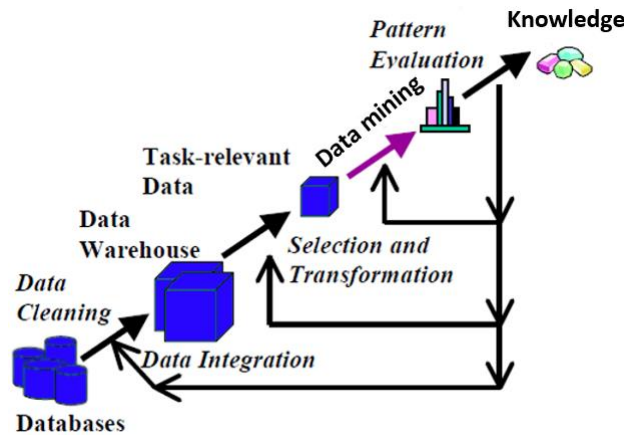


FIGURE 2.1 - Knowledge Discovery process [1,2]

2.1.2. Data mining process

In general, a knowledge discovery process consists of an iterative sequence of several steps. In order to transform the data to knowledge, we have to go back and forth between the steps to improve and enrich the knowledge produced.

- **Data cleaning:** it is a phase in which noisy, missing, or irrelevant data are removed from the collection.
- **Data integration:** at this step, multiple heterogeneous data sources may be combined in a common source.
- **Data selection:** where data relevant to the analysis task are retrieved from the database, where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations, mining, which is an essential process where intelligent methods are applied in order to extract data patterns.
- **Data transformation:** or data consolidation, it is a stage in which the selected data is transformed into forms suitable for the mining process.

- **Data mining:** it is the essential step in which clever methods are applied to extract patterns potentially useful.
- **Pattern evaluation:** which is to identify the truly interesting patterns representing knowledge based on some measures.
- **Knowledge presentation:** where visualization and knowledge representation techniques are used to present mined knowledge to the user. This essential step uses visualization techniques to help users understand and interpret the data mining results.

Some of these steps can be combined together, such as, data cleaning and data integration can be executed together as a pre-processing phase to generate a data warehouse. In addition, data selection and data transformation can be combined where the consolidation of the data is the result of the selection.

2.1.3. Classification

Several techniques of data mining have been developed and used for knowledge discovery from databases. Among these techniques, we can cite:

- Association,
- Classification,
- Clustering,
- Prediction,
- ...

In this manuscript, we are interesting to classification and some related techniques.

Automatic classification consists of grouping various objects (individuals) into subsets of objects (classes). There are two types of classifications:

- **Supervised:** classes are known a-priori; they usually have an associated semantics.
- **Unsupervised clustering:** classes are based on the structure of objects, the semantics associated with classes is more difficult to determine.

In both cases, we need to define the notion of distance between two classes: the aggregation criterion.

2.1.3.1. Supervised classification

Let $D = \{d_1, d_2, \dots, d_i, \dots, d_m\}$ a set of documents each of them is represented respectively by a description $\vec{d_1}, \vec{d_2}, \dots, \vec{d_i}, \dots, \vec{d_m}$ and $C = \{C_1, C_2, \dots, C_k, \dots, C_c\}$ a set of classes, the supervised classification assumes two known functions. The first matches any individual d_i with a class C_k . It is defined by means of (d_i, C_k) given as examples to the system. The second matches any individual d_i with his description $\vec{d_i}$. The supervised classification then consists in determining a classification procedure:

$$C^f: \vec{d_i} \rightarrow C_k \quad (2.1)$$

where from the description of the element determines its class with the lowest error rate. The performance of the classification depends in particular on the effectiveness of the description. Moreover, if one wants to obtain a system of learning, the procedure of classification must make it possible to classify efficiently any new example (predictive power).

2.1.3.2. Unsupervised classification

Unsupervised classification is used when there are documents that are unclassified. At the end of the unsupervised classification process, the documents must belong to one of the classes generated by the classification. There are two categories of unsupervised classifications: **hierarchical** and **non-hierarchical**.

In the **hierarchical classification (CH)**, the created subsets are nested hierarchically in each other. We distinguish the descending (or divisive) CH which starts from the set of all the individuals and splits them into a certain number of subsets, each subset then being broken up into a certain number of subsets, and so on. Moreover, the ascending (or *agglomerative*) CH that starts with single individuals that are grouped into subsets, which are in turn grouped, and so on. To determine which classes we are going to merge, we use the aggregation criterion.

In the **non-hierarchical classification**, individuals are not hierarchically structured. If each individual is only part of a subset, we speak of partition. If each individual can belong to several groups, with the probability P_i to belong to group i , then we speak of recovery.

2.1.3.3. Aggregation criterion

The aggregation criterion makes it possible to compare classes two by two to select the most similar classes according to a certain criterion. The classic criteria are the **nearest neighbor**, the **maximum diameter**, the **average distance** and the **distance between the centers of gravity**.

2.1.3.4. Evaluation of a classification system

In this subsection, we present a method for evaluating supervised classification, and standard techniques for measuring and comparing unsupervised classification systems.

- **Test corpus (supervised case)**

To test the quality of a supervised classification procedure, the classified elements are randomly separated between a reference bases (R) and a test base (T). Then, the classification procedure C^f is determined from the examples of the reference base. Then, we use C^f to find the class of elements of the test database. Finally, the error of the classification procedure is considered.

To estimate the error rate TE of a classification procedure C^f , a simple method is to calculate the number of badly classified elements on the number of elements to classify:

$$TE(C^f) = \frac{1}{card(T)} \sum_{t=1}^{card(T)} (C^f(\overrightarrow{dt}) \neq C_{dt}) \quad (2.2)$$

where C_{dt} is the original class of d_t .

In the case of simple classifications, we may have to calculate the error resulting from a purely random classification C^a to compare it with the error made by our procedure C^f in order to check the performance of our system.

Let P_k the frequency (or a priori probability) of the k class in the test database, we call TE_a error of the random system:

$$TE_a = 1 - \sum_{k=1}^c (P_k)^2 = 1 - \sum_{k=1}^c \left(\frac{\text{card}(C_k|T)}{\text{card}(T)} \right)^2 \quad (2.3)$$

where c is the number of classes and $\text{card}(C_k|T)$ is the number of elements of T that are in the C_k class.

The apparent error $TE(C^f)$ is dependent on the sample under consideration. However, the larger the number of elements in the sample, the more the measured error tends to the real error of C^f .

- **Unsupervised case**

In the case of unsupervised, classification can be assessed against some of these characteristics. One distinguishes on the one hand, numerical characteristics: the number of classes obtained, the number of elements per class, the average number of elements per class, the standard deviation of the classes obtained, and on the other hand, the semantic features. For example, if a document is associated with a set of keywords, the semantics associated with a class may consist of the most common words in the class.

To evaluate the homogeneity of the number of images per class, we can use the variance:

$$V = \sigma^2 = \frac{1}{c} \sum_{k=1}^c (\text{card}(C_k) - \text{moy})^2 \quad (2.4)$$

where $\text{moy} = \frac{1}{c} \sum_{k=1}^c \text{card}(C_k)$ is the average number of elements per class and c is the number of classes obtained. The standard deviation $\sigma = \sqrt{V}$ expresses the dispersion in the same unit as the average.

2.1.3.5. Classification techniques

Among the classification techniques, in this manuscript, we will be required to use classification by decision tree.

2.1.3.6. Classification by decision tree

A decision tree is a graphical representation of a classification procedure. The internal nodes of the tree are tests on the fields or attributes, the leaves are the classes. Each leaf represents a prediction of a solution to the related problem. The buildings of decision tree classifier do not need any domain knowledge or parameter setting and therefore is suitable for exploratory knowledge discovery. High dimensional data can be handled by decision tree. When the tests are binary, the left son corresponds to a positive test response and the right son to a negative answer.

An example of a decision tree is shown in (FIGURE 2.2):

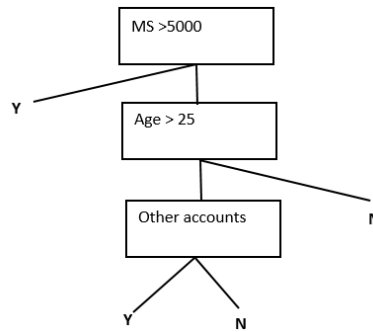


FIGURE 2.2 - Example of a decision tree; *MS* is the average balances of the current account, *other accounts* is a binary field that is yes if the customer has other accounts, class *Y* indicates a prior favorable for the allocation of a loan

Decision trees offer several benefits in data mining:

- Easy to track when compacted and understandable
- Able to hold a variety of input data: numeric, nominal, and textual
- Work with datasets that may have missing values or errors
- High predictive performance for a relatively small computational effort
- Offered in many data mining packages over a variety of platforms
- Useful for various tasks, such as regression, classification, feature selection and clustering.

- **Learning decision tree**

With an input of a sample of m classified records $(\vec{x}, c(\vec{x}))$, a learning algorithm must output a decision tree. Most algorithms proceed in a descending way that is, they choose the root of the tree (usually a test) and then, recursively, choose the label of the leaves. To simplify the presentation, we limit ourselves, in this section, to problems where the attributes are discrete and the number of classes equal to 2. The generic algorithm can be written:

Decision Tree Learning Algorithm

Input : a sample S of m classified records $(\vec{x}, c(\vec{x}))$

initialization : empty tree; current node: root; current sample: S

repeat

decide if the current node is terminal

if the current node is terminal **then**

label the current node with a leaf

else

select a test and create the subtree

End if

current node: a node not yet studied

current sample: sample reaching the current node

until production of a decision tree

prune the decision tree obtained

output : pruned decision tree

- **CART**

CART was defined in the 1980s [3]. Since then, it has been integrated with many data mining environments in many variations. We give here its main peculiarities. Originally, the algorithm only considered binary tests. The function that measures the degree of mixing (and therefore the gain) is the Gini function (the broadcast versions offer other choices). For pruning, an upward progression of the constructed tree is carried out. To decide if a sub-tree can be pruned, the actual estimated error of the current tree is compared with the pruned tree. The estimate of the actual error is measured on a test set or by cross validation.

- **C5**

C5 is the most recent version of an ID3 algorithm developed by R. Quinlan [4]. The algorithm can take into account attributes of any arity. The function that measures the degree of mixing (and thus the gain) is the entropy function. This function tends to favor attributes with a large number of values. To avoid this bias, an information gain function is also available. Pruning is done with the learning set by a pessimistic assessment of the error. Although this technique may seem inappropriate, it gives good results in practice.

2.1.4. Confusion matrix

The confusion matrix presents in the form of a contingency table comparing the assignment class (in column) with the class of origin (in line) of the individuals composing the sample. We have two types of information:

- The number of times the model was wrong
- The type of error when ranking

Figure 2 presents a confusion matrix for a model of 2 classes A and B.

In general, the performance of a model is assessed through a confusion matrix, which compares the actual situation and the situation predicted by the model in order to estimate the error rate.

TABLE 2.1 - Confusion matrix for 2 classes A and B

ACTUAL CLASS	PREDICTED CLASS	
	A	B
A	$n_{A.A}$	$n_{A.B}$
B	$n_{B.A}$	$n_{B.B}$

In this matrix, $n_{A.B}$ represents the number of cases of class A assigned to class B and $n_{B.A}$ represents the number of cases of class B assigned to class A, while $n_{A.A}$ and $n_{B.B}$ represent the correct number of classification.

From this confusion matrix, we can identify three types of indicators:

The global error rate in resubstituting: This error rate is calculated on the learning sample Ω_a , it is generally optimistic, i.e. lower than the theoretical error rate, which represents the probability that we are wrong if we apply the classifier over the entire population, which is impossible. In order to overcome this bias of optimism, which is very difficult to estimate, it is generally proposed to subdivide the two-

part samples, a learning database Ω_a and a test database Ω_t . These two bases include different records. From experience, the learning base will resume from 70% to 80% of the recordings, the test base consisting of the remaining 20 to 30. The learning base is then used to build the model and the test base is used to check the stability of the model. In this case, we calculate the error rate called global error rate in validation. For the matrix above the overall error rate is calculated as follows:

$$\varepsilon_{global} = 1 - success \quad (2.5)$$

$$\varepsilon_{global} = 1 - \frac{n_{A.A} + n_{B.B}}{card(M)} \quad (2.6)$$

where card (M) is the total number of individuals.

The error rate a priori: it is the probability that an individual belonging to the class k is not assigned to the class k. For our example and for class A, the error rate a priori is given by the following equation:

$$\varepsilon_{a \text{ priori}} = \frac{\sum_{k \neq A} n_{A.k}}{\sum_k n_{A.k}} \quad (2.7)$$

with k represents the different classes, in our case A or B.

The error rate a posteriori: it is the probability that an individual assigned to the class k actually belongs to the class k. for our example and for class A, the error rate a posteriori is:

$$\varepsilon_{a \text{ posteriori}} = \frac{\sum_{k \neq A} n_{k.A}}{\sum_k n_{k,A}} \quad (2.8)$$

The overall error rate allows us to know how a classifier will act on all the data; however, it does not allow us to distinguish what is the level of success for each class. This is why the error rate a priori has been calculated. It is in fact to calculate the success rate relative to each class. This is the complement of the classic recall rate criterion used most often in information retrieval systems (IRS). The error rate a posteriori allows us to focus on the credibility of a classification, such as, for example, know the certainty that an individual rated A is of this class A. It is therefore the complement of the classic rate precision as used in SRIs.

The evaluation of partial performances makes it possible to determine on which errors the classifier is less efficient, to compare several classifiers, to make them cooperate by using on the categories that they predict better.

2.1.5. Cross validation

Cross-validation proposes to divide the sample base into "s" equal parts, with learning on the (s-1) parts of the database, and test on the remaining part. Then, there is a permutation of the tested databases, and thus a confusion table is created by averaging the "s" tests performed. It is therefore a repetition of the "learning-validation" pair, but ensuring that there is no overlap between the validation samples.

2.1.6. Bootstrap

The idea is to use the sample of the observations to allow a finer statistical inference. A number of samples, referred to as the bootstrap sample, are obtained by random sampling of observations from the initial sample. On each of the bootstrap samples, the different parameters of the model are estimated. A sequence of parameters is therefore obtained. Under certain conditions of regularity, the theory shows that the distribution of the sequence of parameters obtained converges towards the real distribution of the parameter.

The bootstrap has now become established in the statistical field as a very practical technique of statistical inference. It requires, indeed, "few" assumptions and is relatively easy to program, they are, in fact, only random draws. However, the big disadvantage lies in the large computational capabilities that the application of these techniques requires; in addition, it is often recommended to conduct at least a hundred repetitions to hope to have good reliability.

2.1.7. Data mining software: THE WEKA WORKBENCH

The objective of WEKA software is to offer a comprehensive collection of learning machine algorithms and data preprocessing tools to researchers and practitioners alike. It can be used to try out and compare different methods on new data sets. Its modular, extensible architecture allows sophisticated data mining processes to be built up from the wide collection of base learning algorithms and tools provided. Extending the toolkit is easy thanks to a simple API, plugin mechanisms and facilities that automate the integration of new learning algorithms with WEKA's graphical user interfaces. The workbench includes algorithms for classification, regression, clustering, association rule mining and attribute selection [48].

2.2. Graphs

2.2.1. Definitions and Notations

A graph $G = (V, E, w)$ consists of a finite set V of *vertices* (or *nodes*) and a set E of weighted edges and a weight function, denoted w .

- **Definition of vertex**

In general, a vertex $u \in V$ is an abstraction of an element of the data structure represented by the graph. It may be a pixel, a network or mesh point or even an entry into a database. The relationship between the data structure and the set of vertices is given by:

$$V \subseteq \Omega \subset \mathbb{R}^m,$$

where Ω denotes the discrete domain of the data structure.

- **Definition of edge**

An edge connects two vertices u and v (u, v) of a graph, describes an interaction between the elements represented by these vertices. Two connected vertices are called adjacent and denoted as $u \sim v$. The set E of the edges of G is a subset of $V \times V$ and is defined by:

$$E = \{(u, v) \in V \times V \mid u \sim v \text{ and } u \neq v\} \quad (2.9)$$

The property of symmetry implies that if $(u, v) \in E$, then $(v, u) \in E$. By definition, a vertex cannot be connected to itself.

- **Vertex neighbors**

The neighborhood set of a vertex u , denoted $\mathcal{N}(u)$ is the set of vertices adjacent to u and defined by:

$$\mathcal{N}(u) = \{v \in V \mid (u, v) \in E\} \quad (2.10)$$

In the case of a complete graph, we have $\mathcal{N}(u) = V - \{u\}$, $\forall u \in V$.

- **Weight function**

The weight function $w: V \times V \rightarrow [0, 1]$, associates with each pair of vertices $(u, v) \in V \times V$ a similarity value representing the amount of interaction between the vertices u and v . By definition, the function is symmetric and $\forall (u, v) \in V \times V$, we have the relation $w(u, v) = w(v, u)$. In the following of this manuscript, we will use the condensed notation w_{uv} to denote the weight $w(u, v)$. By convention, the weight function checks for the following properties:

$$w_{uv} = \begin{cases} \in [0,1] & \forall (u, v) \in E \\ = 0 & \forall (u, v) \notin E \\ w_{vu} & (Symmetric). \end{cases} \quad (2.11)$$

- **Vertex degree**

The degree of a vertex, denoted $\delta(u)$, denotes the number of adjacent vertices with u . The degree $\delta: V \rightarrow \mathbb{N}$ defined as:

$$\delta(u) = \text{Card}(\mathcal{N}(u)). \quad (2.12)$$

The weighted degree of a vertex, denoted $\delta_w(u)$, represents the total quantity interaction of a vertex with its adjacent vertices. The weighted degree $\delta_w: V \rightarrow \mathbb{R}^+$ is defined by:

$$\delta_w(u) = \sum_{v \in \mathcal{N}(u)} w_{uv}. \quad (2.13)$$

In the case where the graph is unweighted (i.e. the weight function is always 1 on the edges), we have

$$\delta(u) = \delta_w(u), \forall u. \quad (2.14)$$

The normalized degree of a vertex, denoted $\delta^N(u)$, represents the interaction overall of a peak of the graph, or average interaction. The normalized degree $\delta^N: V \rightarrow \mathbb{R}^+$ is defined by:

$$\delta^N(u) = \frac{\delta_w(u)}{\delta(u)}. \quad (2.15)$$

• Terminology

- In a simple graph, two different vertices are connected with single edge only.
- Multigraphs may have multiple edges connecting the same two vertices. When m different edges connect the vertices u and v , we say that (u, v) is an edge of multiplicity m .
- An edge that connects a vertex to itself is called a loop.
- A pseudo graph may include loops, as well as multiple edges connecting the same pair of vertices.

2.2.2. Graphs types

There are many types of graphs:

- Undirected graphs
- Directed graphs
- Complete graphs
- A cycle
- ...

Two vertices u, v in an undirected graph G are called adjacent (or neighbors) in G if there is an edge e between u and v . For all $v, w \in V$: $(v, w) \in E \Leftrightarrow (w, v) \in E$. In contrast, each edge in direct graph is an ordered pair of vertices. The directed edge (u, v) is said to start at u and end at v . Then u is the initial vertex of this edge and is adjacent to v and v is the terminal (or end) vertex of this edge and is adjacent from u . The initial and terminal vertices of a loop are the same.

A complete graph on n vertices, denoted by K_n , is the simple graph that contains exactly one edge between each pair of distinct vertices.

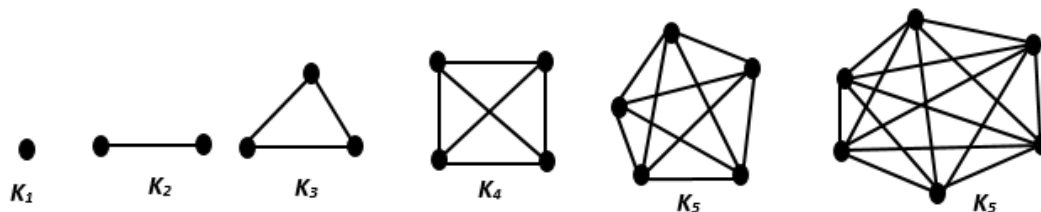


FIGURE 2.3 - Complete graph

A cycle C_n for $n \geq 3$ consists of n vertices v_1, v_2, \dots, v_n and edges $(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n), (v_n, v_1)$.

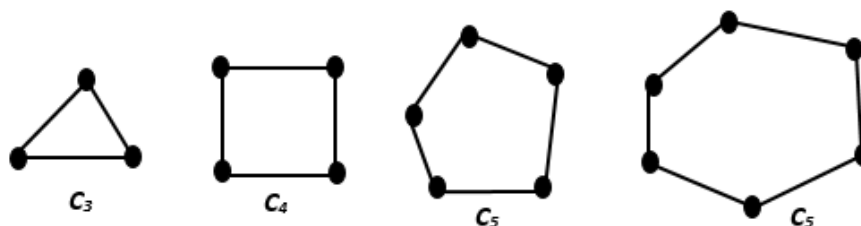


FIGURE 2.4 – Cycle graph

Note. In this manuscript, we consider simple undirected graphs only (without multiple edges and without loops), weighted [45] and whose edges are all symmetrical.

2.2.3. Sub graph

A subgraph of a graph $G = (V, E, w)$ is a graph (K, F, w) , where $K \subset V$ and $F \subset E$. A subgraph H of G is a proper subgraph of G if $H \neq G$. The subgraph induced by a subset K of the vertex set V is the graph (K, F, w) ; where the edge set F contains an edge in E if and only if both endpoints are in K .

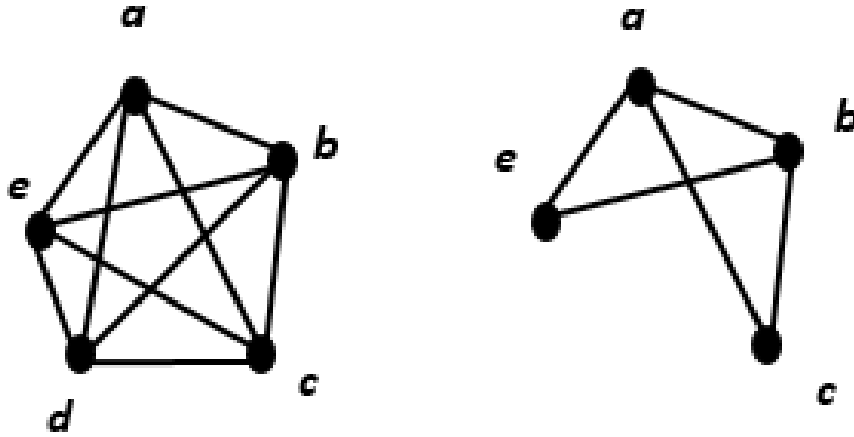


FIGURE 2.5 – Sub graph

2.2.4. Graphs construction

In this subsection, we present the different types of graphs constructions used in this manuscript. The construction of these graphs is crucial to obtain a good treatment data to be represented and strongly depends on the nature of these data, whether or not organized and whether or not they have a representation natural in the form of a graph.

2.2.4.1. Metric and Similarity

In this paragraph, we consider a domain $\Omega \subset \mathbb{R}^m$, and a characteristic function (or function of attributes) $\mathcal{F}: \Omega \rightarrow \mathbb{R}^e$, which associates a characteristic vector to each element of Ω . There are different choices for defining the characteristic function, these choices being generally dependent on the application. Among the most common characteristic functions we can cite those associating with each point of Ω his coordinates on Ω , as well as those which depend on an initial function $\mathcal{F}^0: \Omega \rightarrow \mathbb{R}^e$ defining the data.

The metric is an essential notion in the construction of a graph, whether for the establishment of all edges (especially in the case of unorganized domains), or for the definition of the function of weight. A metric is an application $\mu: \Omega \times \Omega \rightarrow \mathbb{R}^+$, which has each pair of points u and v ($u, v \in V$) associates a distance. The most common metric is the Euclidean distance given by:

$$\mu_2(u, v) = \sqrt{\sum_{i=1}^c (\mathcal{F}_i(u) - \mathcal{F}_i(v))^2} \quad (2.16)$$

The choice of the metric used depends strongly on the type of domain and applications considered, there is no general rule [46].

2.2.4.2. Similarity measurement

Beyond the metric, it is necessary to define the notion of similarity, which is the basis of any weight function. This measure reflects affinities between the elements of the initial domain and, more precisely, a metric, allows placing all the similarities of the domain on the same scale. Just like the metric, this measure depends on the application considered and the data type. There is no general rule for determining the most sensible similarity measure for an application but we will see later in this manuscript than for particular cases certain measures.

In general, a similarity measure is an application $g: \Omega \times \Omega \rightarrow [0, 1]$ which assigns each pair of elements $u, v \in \Omega$ a similarity index. The closer this index is to 1, the more similar the elements are.

Therefore, a function weight $w: V \times V \rightarrow [0, 1]$ is redefined as following:

$$w(u, v) = \begin{cases} g(u, v) & (u, v) \in E \\ 0 & (u, v) \notin E \end{cases} \quad (2.17)$$

We will now present the different graph structures associated with different types of domains. The construction of these structures also depends on a metric and a weight function.

2.2.4.3. Unorganized domains

Let us first consider the most general case, corresponding to the domains having no organization (or at least no sufficient organization to directly induce a graph structure).

Let $\Omega \subset \mathbb{R}^m$ an unorganized discrete domain, and $G = (V, E, w)$, a graph representing the domain Ω . By definition, each vertex of the graph represents an element of Ω .

- **Graph of k nearest neighbors**

Let $k \in \mathbb{N}$ and μ be a metric for the discrete domain. The graph of k nearest neighbors (k -NN) is a weighted graph whose each vertex is connected to its k closest neighbors, according to the metric μ . The set of k plus close neighbors of a vertex u is noted $\mathcal{K}(u)$. By nature, the neighborhood relations of a k-NN graph are not symmetrical. In order to preserve the property of symmetry of the edges, we will use in this manuscript a symmetrical (or reciprocal) version of the graph k-NN. In this version, the E set of edges is defined by:

$$E = \{(u, v) | u \in \mathcal{K}(v) \text{ or } v \in \mathcal{K}(u)\} \quad (2.18)$$

This construction ensures that each vertex of the graph is connected to at least k other vertices. In return, a vertex can have to N - 1 neighbors (where N is the total number of vertices).

Note. There is no general rule for determining the number k of neighbors. This choice is strongly dependent on the application considered.

- **ε -neighborhood graph**

The " ε -neighborhood" graph is a weighted graph whose set of edges is defined by a threshold on a metric μ . A vertex v belongs to neighborhood of a vertex u if the distance between u and v is below the threshold ε .

On the other hand, if one places oneself in the space of the function characteristic \mathcal{F} , on which is defined the metric μ , the ε -neighborhood of a vertex u , denoted $\mathcal{N}_\varepsilon(u)$, consists of all the vertices whose image by \mathcal{F} belongs to the ball of radius ε and centered in $\mathcal{F}(u)$. The ε -neighborhood of a vertex u is then

$$\mathcal{N}_\varepsilon(u) = \{v | \mu(u, v) < \varepsilon\} \quad (2.19)$$

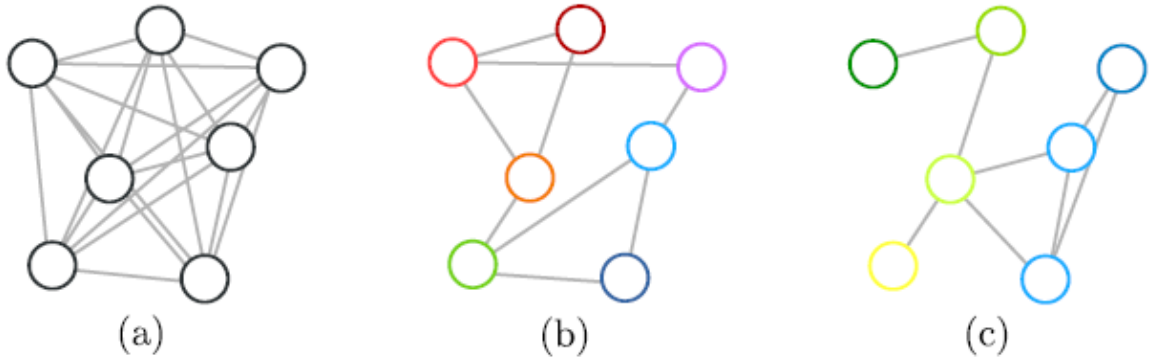


FIGURE 2.6 - Graph construction on an unorganized domain. ((a) complete graph, there is an edge connecting each pair of vertices. (b) A graph of the k nearest neighbors, where each vertex is connected to two vertices that are closest to it (here in the sense of color). (c) A ε -neighborhood graph, where each vertex is connected to the set of vertices whose distance (here in the sense of the color) does not exceed a value ε

and the set of edges is defined by:

$$E = \{(u, v) | v \in \mathcal{N}_\varepsilon(u)\} \quad (2.20)$$

For a more complete review of neighborhood graphs, the interested reader can refer to [47].

- **Delaunay graph**

The Delaunay graph is a weighted graph whose set of edges is defined by a circumscribed spheres from a given 3D surface point P_i such that no surface point is inside these spheres. Then, the set of points located on the circumscribed spheres borders P_j will be defined as the set of P_i neighbours. These border points are the set of vertices located on the corners of all triangles (faces) sharing P_i 's Neighbours as shown in (FIGURE 2.5).

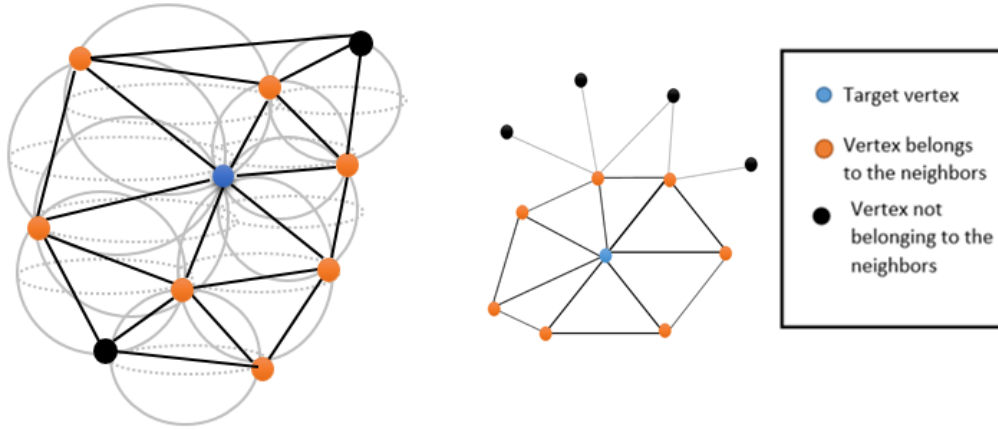


FIGURE 2.7. Delaunay graph construction

2.2.4.4. Organized domains

In this section, we consider the case of domains naturally organized, and therefore possessing at least one intrinsic graph structure.

- **Cases of images and other discrete signals**

A discrete signal can be seen as a function $f: \mathbb{Z}^q \rightarrow \mathbb{R}^c$, where $q = 1, 2$ or 3 , respectively corresponding to a 1D signal, an image (2D), or a volume image (3D).

Whatever the size of the signal, the set of vertices V is defined as a subset of the signal definition domain. Therefore, each vertex represents a sample of the signal and we have $V \subset \mathbb{Z}^q$.

The neighborhood set of a vertex, from a discrete signal, is usually defined according to a distance measurement on the spatial coordinates associated with each vertex. The most usual distance and corresponding to the intrinsic structure of the space of integers is the distance from Manhattan (norm \mathcal{L}_1), where each vertex u is connected to the set of vertices adjacent to u on the different dimensions of the signal. Other constructions also highlight the distance of Chebyshev (norm \mathcal{L}_∞), for a wider neighborhood.

Considering an image, and imposing a construction of ε -neighborhood with $\varepsilon = 1$, we find the standard tessellations to train the 2D grids representing the images.

- In the case of Euclidean distance μ_2 , only the 4 adjacent vertices are added to the neighborhood and the graph is a so-called 4-connectivity graph.
- In the case of a Chebyshev distance μ_1 , the 4 vertices on the diagonals are also added to the neighborhood and the graph is a graph said of 8-connectivity.

These two standard constructions are illustrated in (FIGURE 2.8).

***Note.** A discrete signal can also be considered as an unorganized domain. In this case, the intrinsic structure of the data is not taken into account and the graph is constructed in the same way as for an unorganized domain. The graph can also be constructed by combining both approaches and using*

the edges generated by both types of constructions (considering an organized field, then unorganized).

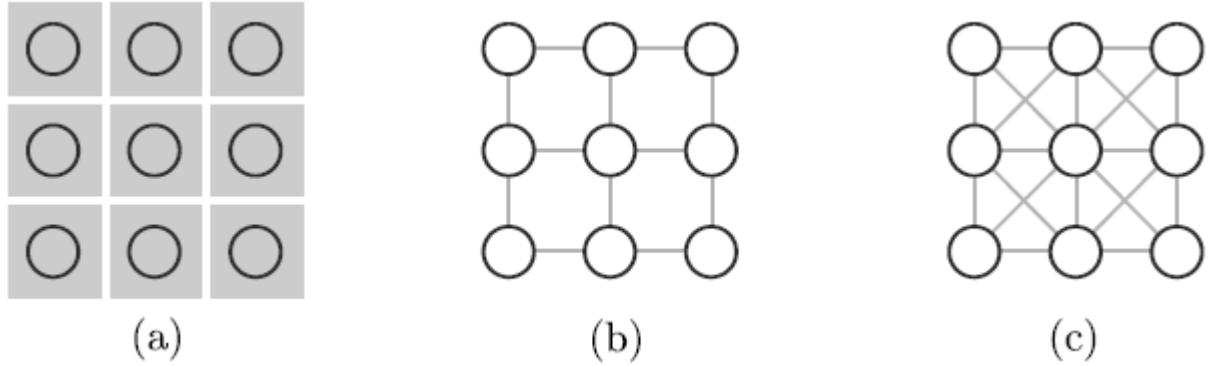


FIGURE 2.8- Construction of graphs on an image. (a) A regular grid pixels with superimposed vertices. (b) Construction of a graph of 1-neighborhood with a L_2 standard. (c) Construction of a 1-neighborhood graph with a L_1 standard

In the case of simple images, the weight function is usually based on a comparison between the intensity (or the color vector) of the pixels. Therefore, the characteristic function \mathcal{F} is simply defined such that $\mathcal{F} = f$.

- **Case of 3D meshes**

A 3D mesh (commonly called a mesh) is a discrete surface representation of a 3D object. It is composed of a set of points contained in \mathbb{R}^3 , these points being connected to each other by edges for to form faces (approximating the surface of the object). In the case of a mesh triangular, the faces are triangles whose each vertex corresponds to a point of the mesh.

In this case, the construction of the graph is implicit and is generally limited to consider the mesh as a graph and to keep the vertices and edges.

This type of construction is valid for any type of mesh, whether triangular or not, and whatever the size of the space containing it. It is shown in (FIGURE 2.9).

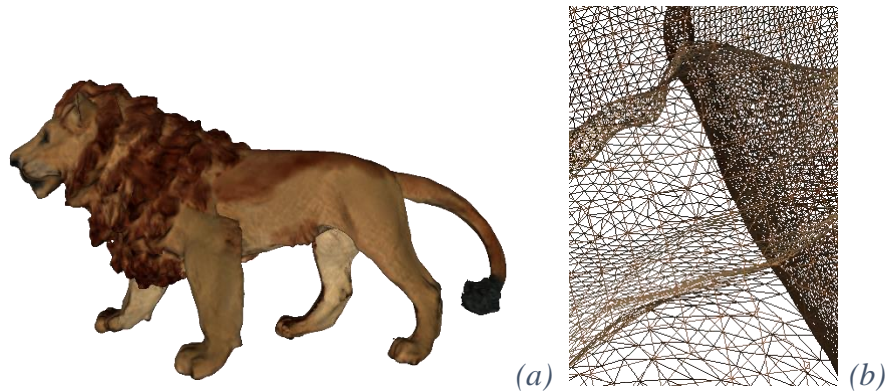


FIGURE 2.9 - Construction of a graph from a 3D mesh. (a) The 3D textured mesh. (b) The graph induced by this mesh

2.3. Conclusion

In this chapter, we presented some definition of knowledge discovery (KDD) and Data mining. Moreover, the different types of classification are presented and especially the decision tree approach with its different variations. Furthermore, we introduced in this chapter the different definitions, notations and concepts inherent to weighted graphs. In the next chapters, we will present our proposed methods based on the general concepts detailed in the current chapter.

Chapter 3

EFFICIENT 3D POINT CLOUDS CLASSIFICATION USING LINEAR PROGRAMMING AND DATA MINING

Summary

3. EFFICIENT 3D POINT CLOUDS CLASSIFICATION USING LINEAR PROGRAMMING AND DATA MINING	33
3.1. Introduction	34
3.2. Skin detection method.....	34
3.2.1. Learning dataset construction	35
3.2.2. Prediction rules generation.....	37
3.3. Skin Region Segmentation.....	39
3.3.1. Notations and definitions	39
3.4. Face detection	44
3.4.1. Refinement based on number of non-skin regions.....	44
3.4.2. Refinement based on size.....	45
3.4.3. Refinement based on the distance between face parts	46
3.5. Experimental results	47
3.5.1. 3D point clouds and 3D meshes	47
3.5.2. 2D images	52
3.6. Comparison with the state-of-the-art	54
3.7. Conclusion	55

Most of applications related to security and biometric rely on skin region detection such as face detection, adult 3D Objects filtering, and gesture recognition. In this chapter, we propose a robust method for skin detection on 3D colored point clouds. Then, we extend this method to solve the problem of 3D face detection. To do so, we construct a weighted graph from initial colored 3D point clouds. Then, we present a linear programming algorithm (LP) using a predictive model based on a data mining approach in order to classify and label graph vertices as skin and non-skin regions. Moreover, we apply some refinement rules on skin regions to confirm the presence of a face. Furthermore, we demonstrate the robustness of our method by showing and analyzing some experimental results. Finally, we show that our method deals with many data that can be represented by a weighted graph such as 2D images and 3D models.

3.1. Introduction

Nowadays, owing to the fast growing of digital field and informational data, 3D object processing, including skin and face detection, became an essential pre-processing phase for many applications such as human – machine interaction, gesture recognition, human identification, and robotics.

In this chapter, we propose to extend the skin detection method from 2D images to 3D colored point clouds using weighted graphs representation. In addition, defining operators on graphs leads to generalize the face detection method to any data that can be represented by a weighted graph. The first step in our approach consists of modelling the 3D surface using a weighted graph, and a predictive model based on a data mining approach is constructed. This model leads to define a statistical function that classifies graph vertices into skin and non-skin vertices. Then, the graph is segmented into regions using linear programming algorithm where each region is defined as a set of vertices of similar class. All skin regions are considered as candidate face regions and the presence of a face is confirmed by applying refinement steps such as face size. Our method is illustrated in the following flowchart (FIGURE 3.1).

The main goal of this chapter is to present an accurate method for skin region and face detection on 3D colored point clouds using weighted graphs representation. This model aims to extend skin models proposed for 2D images to any data that can be represented by a graph and accordingly with different types of visual data such as video and 3D objects.

3.2. Skin detection method

In this subsection, we detail the different phases of construction of our skin model, relying on data mining techniques and image analysis. The method uses data mining techniques to produce the prediction rules based on the RGB color spaces.

In order to elaborate the skin model, we proceeded in three steps:

- The first step is dedicated to the construction of our dataset, as well as the preparation of data for the learning phase.

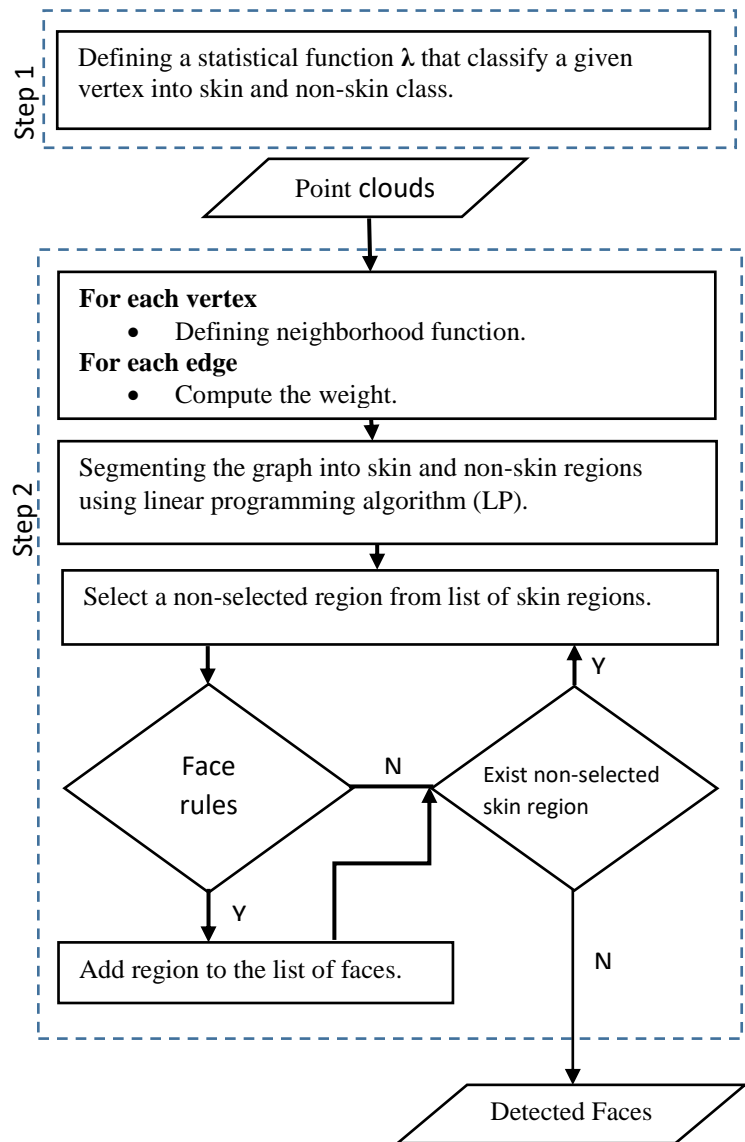


FIGURE 3.1- Flowchart of our method

- The second step is to find a prediction model associated with a representation space able to discriminate skin pixels from non-skin pixels. This representation space can be one of the classic color spaces; the chosen color space is the RGB color space. The tools of data mining then allow us to retain the appropriate decision rules.
- The last step is to define a statistical function based on the previous decision rules to be used later on in the phase of graph segmentation.

The following flowchart illustrates the prediction rules generation.

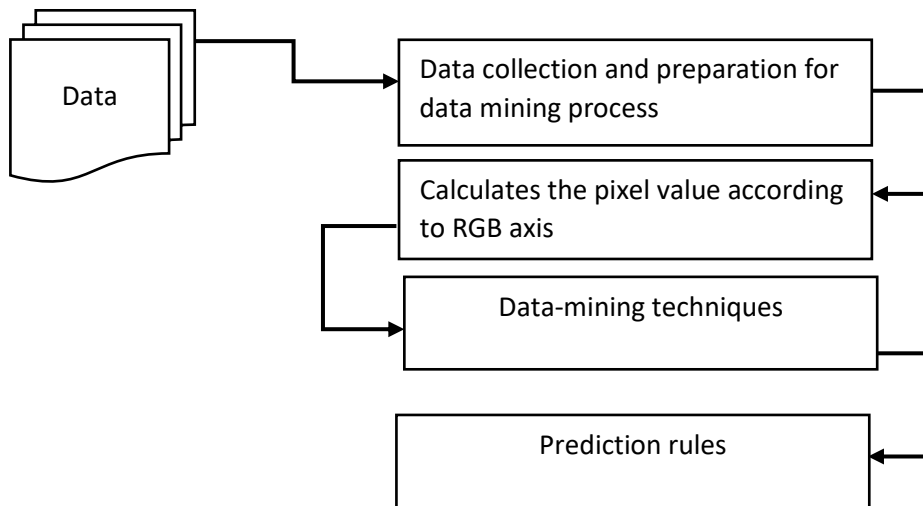


FIGURE 3.2 - Flowchart of prediction rules generation

3.2.1. Learning dataset construction

The construction of the learning dataset is an important element in the process of extracting knowledge from the data. For our problem of classification of pixels in pixels (vertices in case of Graph) of skin or not, the quality of the dataset can be judged on the following factors (The size of the database and the variety in the contents of images), which must be representative for the different sexes, races, and lighting conditions. Therefore, we used the existing images dataset “Helen dataset” [49] (FIGURE 3.3) that contains images with different lighting conditions, races, and sexes. This work led to a learning dataset consisting of 50,000 pixels of different classes (skin and non-skin).

Building tools

Building the learning dataset is not a simple task. Moreover, since the size of the database must be large, it is difficult to build it manually. Hence, the need for such software that helps us build this database.



FIGURE 3.3 - Extract from “Helen dataset”

For this reason, we developed a system that allows us to perform this task as shown in the following figure:



FIGURE 3.4 - Learning dataset creator

Learning data file

Building learning data file using our system occurs through two phases in order to generate prediction rules:

- Using the developed system to save pixels colors in a database (TABLE 3.1),
- Creating Weka data file (.arff).

TABLE 3.1 - Information extraction from pixels database: the rows in the table correspond to the representative pixels, while the columns correspond to the representation axes of the RGB (maximal values 255 255 255) space and the class (skin or no skin)

R	G	B	Class
X_1^r	X_1^g	X_1^b	X_1^c
.
.			
X_k^r	X_k^g	X_k^b	X_k^c
.
.			
.
.			
X_N^r	X_N^g	X_N^b	X_N^c

As shown in the previous chapter, Weka [48] is a set of tools for manipulating and analyzing data files, implementing most artificial intelligence and machine learning algorithms, including decision trees and neural networks. The Weka file contains information in the following form:

```
@relation data
@attribute R real
@attribute G real
@attribute B real
@attribute Skin {TRUE FALSE}
@data
```

```
206,159,149,TRUE
203,159,148,TRUE
202,159,150,TRUE
204,161,152,TRUE
205,163,151,TRUE
204,164,154,TRUE
204,164,154,FALSE
.....
```

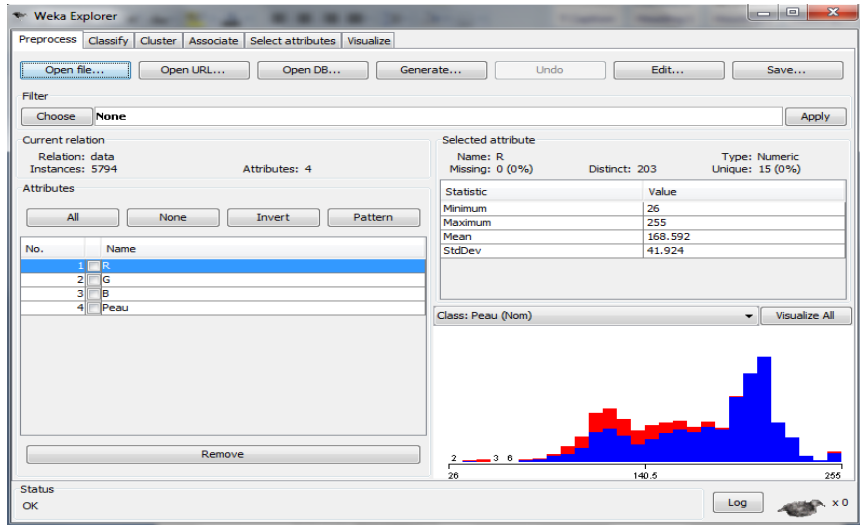


FIGURE 3.5- The Weka software

3.2.2. Prediction rules generation

The Weka program will then use this file (FIGURE 3.5) in order to generate the prediction rules. The decisions rules can be written in the form of the following rules:

If (Condition $1, 1$ and Condition $1, 2$ and Condition $1, 3$) => Skin=FALSE

.....

If (Condition $k, 1$ and Condition $k, 2$ and Condition $k, 3$) => Skin =FALSE

.....

If (Condition $N, 1$ and Condition $N, 2$ and Condition $N, 3$) => Skin =FALSE

Else => Skin=TRUE

where

Condition $i,1$ written as $(R > \dots \text{ ou } R < \dots)$,

Condition $i,2$ written as $(G > \dots \text{ ou } G < \dots)$,

Condition $i,1$ written as $(B > \dots \text{ ou } B < \dots)$.

$\forall i = 1..n$ (n is the number of conditions where the color of given pixel is interpreted as non-skin). In order to find the best prediction model, we tested several techniques based on the J48 [50] decision tree for classification rules and JRip induction graphs [50] using Weka 3.6.0 and we obtain the following results:

TABLE 3.2 - Testing results of J48 and JRip

Algorithm	Cross validation			Confusion Matrix	
Decision tree J48	Correctly Classified Instances	48177	96.1857 %	a	b <-- classified as
	Incorrectly Classified Instances	1910	3.8143 %	4651 101	a = TRUE
	Kappa statistic	0.8698		120 922	b = FALSE
	Mean absolute error	0.0535			
	Root mean squared error	0.1849			
	Relative absolute error	18.1438 %			
	Root relative squared error	48.1385 %			
	Total Number of Instances	50087			
Associations rules JRip	Correctly Classified Instances	47978	95.7887 %	a	b <-- classified as
	Incorrectly Classified Instances	2109	4.2113 %	4639 113	a = TRUE
	Kappa statistic ¹	0.8563		131 911	b = FALSE
	Mean absolute error	0.0642			
	Root mean squared error	0.1956			
	Relative absolute error	21.7589 %			
	Root relative squared error	50.9324 %			
	Total Number of Instances	50087			

The resulting prediction rules based on RGB component of the learning data file lead to a definition of function called lambda λ : $[0...255] \times [0...255] \times [0...255] \rightarrow \{0, 1\}$ as following:

$$\lambda(v_r, v_g, v_b) = \begin{cases} 1 & \text{if the vertex } v(v_r, v_g, v_b) \text{ has a skin color} \\ 0 & \text{otherwise} \end{cases}, \quad (3.1)$$

¹Kappa statistic measures the agreement of prediction with the true class

where v_r, v_g, v_b are the red, green and blue color components of a vertex v . This function takes as input the color components of a vertex and verifies whether this vertex has a skin or non-skin color based on the generated prediction rules.

3.3. Skin Region Segmentation

3.3.1. Notations and definitions

As we mentioned above, our method deals with any data that can be represented by a weighted graph such as 2D images, 3D point clouds, and 3D meshes. We present a review of some basic definitions and operators defined on weighted graphs.

Once the prediction rules are defined, we model the surface in order to treat it. Consider the general situation where a point cloud can be viewed as a weighted graph. An undirected weighted graph $G = (V, E, W)$ consists of a finite set of vertices v , a finite set of edges $E \subset V \times V$, and a set of weight functions W .

Let (u, v) be the edge that connects two vertices u and v . The set of adjacent vertices $N(u)$ of a vertex u shown in (FIGURE 3.6) is constructed using the k-nearest neighbor algorithm and defined as $N(u) = \{v / d(u, v) \leq k\}$, where $d(u, v)$ represents the three-dimensional Euclidean space distance between two vertices (u, v) and defined as:

$$d(u, v) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + (u_3 - v_3)^2}, \quad (3.2)$$

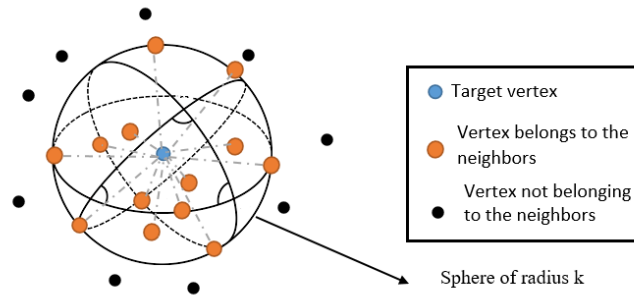


FIGURE 3.6 - Vertices neighbors' construction

The weight function $\omega: V \times V \rightarrow [0, 1]$ that represents a homogeneity measure between a vertex and its neighbors is defined as:

$$\omega(u, v) = \begin{cases} \frac{\nabla(N(u))}{\text{Card}(N(u))} & \text{If } \lambda(u_r, u_g, u_b) = 0 \vee \lambda(v_r, v_g, v_b) = 0 \\ 1 & \text{If } \lambda(u_r, u_g, u_b) = 1 \wedge \lambda(v_r, v_g, v_b) = 1, \end{cases} \quad (3.3)$$

where $\nabla(S)$ represents the cardinality of a skin vertices from a set S and defined as:

$$\nabla(S) = \text{Card}\{u \in S \text{ such as } \lambda(u_r, u_g, u_b) = 1\}. \quad (3.4)$$

Our method consists of classifying graph vertices into skin regions ($F(G)$) and non-skin regions ($B(G)$) using the generated predictive rules where

$$\circ F(G) = \{ v \in G \text{ such as } \lambda(v_r, v_g, v_b) = 1 \} \quad (3.5)$$

$$\circ B(G) = \{ v \in G \text{ such as } \lambda(v_r, v_g, v_b) = 0 \} \quad (3.6)$$

FIGURE 3.7 presents the graph vertices classification: Images (a and c) show the original 3D point clouds, images (b and d) present the detected skin regions where the black regions represent non-skin vertices.

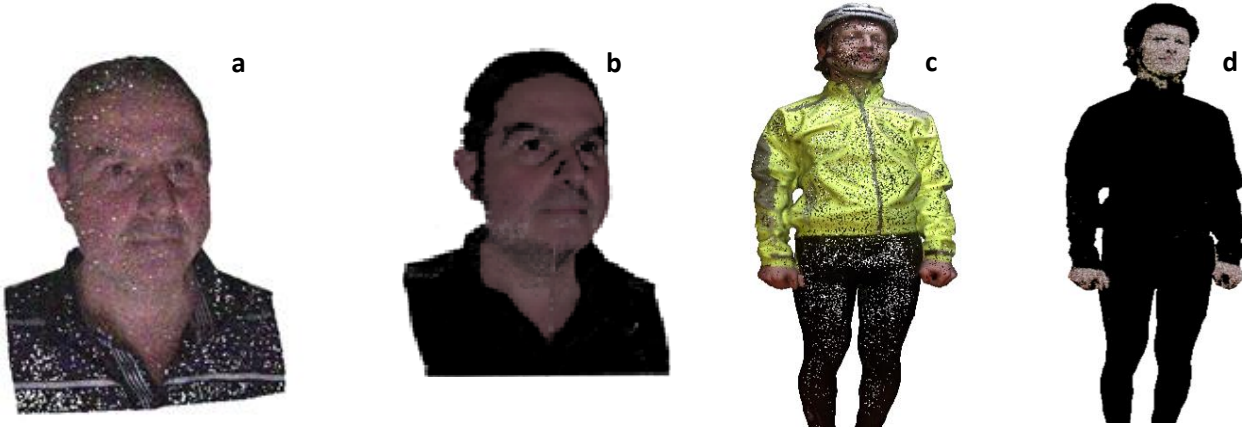


FIGURE 3.7 - Vertices classification

Vertices labelling once we classify the graph vertices as skin and non-skin, we define a set of regions by adopting a linear programming model that groups the set of vertices of the same class in regions by minimizing a cost function. Let us start with the combinatorial problem of labelling the graph vertices where we search to assign the same label to adjacent vertices belonging to the same class using the minimal number of labels.

We propose to formulate the *vertex-labelling* process as an integer linear problem **ILP**. To do so, we consider a set of vertices $V = \{ v_1..v_n \}$ and a set of n labels where the initial state of the problem considers that each vertex has its specific label.

We introduce two binary variables y_k and x_{ik} , $k = 1..n$. The first variable indicates whether a label k is used ($y_k = 1$) or not ($y_k = 0$), and the second one indicates if a given vertex v_i has received the label k . The model with all variables is established as following:

$$\text{Min } \sum_{k=1}^n y_k$$

Subject to:

$$(1) \quad \sum_{k=1}^n x_{ik} = 1 \quad \forall i=1, \dots, n$$

$$(2) \quad x_{ik} - y_k \leq 0 \quad \forall i, k=1, \dots, n$$

$$(3) \quad x_{ik} - x_{jk} \leq 0 \quad \forall k = 1, \dots, n; \forall (v_i, v_j) \in E; \quad v_i, v_j \in F(G) \vee B(G),$$

$$(4) \quad x_{ik} \in \{0, 1\}; y_k \in \{0, 1\}$$

The constraints (4) and (5) ensure that x_{ik} and y_k are binary variables. The constraints (1)-(3) guarantee (in this order) that each vertex is labelled; vertex v_i receives the label k only if this label is used, and any two adjacent vertices have different label if they belong to different classes. The optimal solution of such problem is a graph where each group of connected vertices belonging to the same class receives the same label.

We present a heuristic algorithm to solve the ILP problem. The idea of such algorithm is to start from an empty solution and construct a solution iteratively using expansion criteria that allows one to take a choice at each iteration subject to our problem constraints. The expansion criteria is controlled by the score function $Score(v_i, v_j)$ defined as:

$$Score(v_i, v_j) = \begin{cases} 1 & \text{if } v_j \in C(v_i) \\ 0 & \text{otherwise,} \end{cases} \quad (3.7)$$

where $C(v_i)$ is a set of vertices that combines the color properties and the similarity measure between vertices and defined as:

$$C(v_i) = \{ v_j \in N(v_i) \wedge v_j \in F(G) / \omega(v_i, v_j) \leq \alpha \}, \quad (3.8)$$

where $\alpha \in [0.5..1]$ based on experimental results observations. We introduced this set of vertices to handle the case where some vertices are interpreted as non-skin region due to lighting conditions and presence of small objects like hair, beauty marks, and moles as shown in (FIGURE 3.8).



FIGURE 3.8 - Influence of color information and similarity measure on vertices labelling process

The heuristic function to maximize is defined as:

$$H(v) = \sum_u Score(v, u) \quad (3.9)$$

ALGORITHM 3.1 describes the label propagation algorithm starting from v_i . This algorithm is illustrated in (FIGURE 3.9).

ALGORITHM 3.1: *LabelConnectedSimilar*

//This algorithm makes call to this procedure for each non selected vertex.

Procedure Propagation

Input

Vertex start

Integer K, NB_Label_Used / NB_Label_Used = NB of vertices as initial value*/*

Output:

Connected vertices have the same Label L

Let "[L1]" an empty list of vertices

Let "[L2]" an empty list of vertices


```

If (notSelected (start)) then
  Add start to "[L1]"
End if
While (true)
  Begin
    If ("[L1]" is  $\emptyset$ ) then
      Quit Procedure
    End if

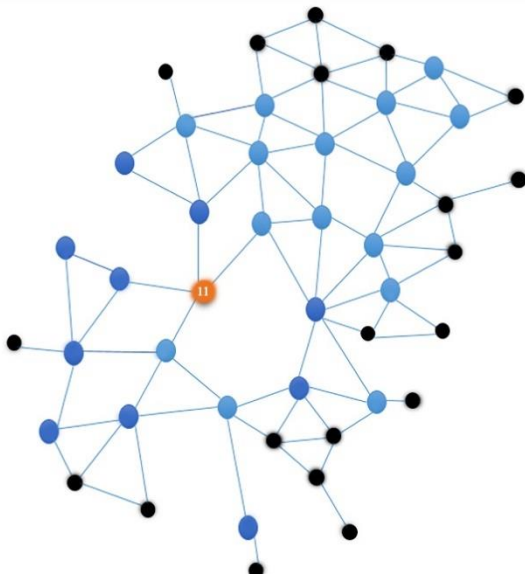
    For each (Vertex  $v_i$  in "[L1]")
      Begin
        Set  $v_i$  as Selected
        Assign  $k$  to  $v_i$ 
      End if

      For each (Vertex  $v$  in "[L1]")
        Begin
          For each (Vertex  $n$  in  $N(v_i)$ )
            Begin
              If (not Selected ( $v_j$ )) then

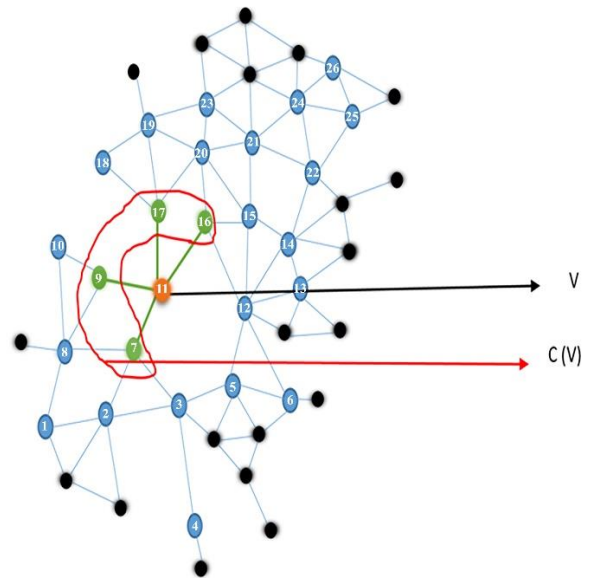
                If (Score( $v_i v_j$ )=1) then
                  Add  $v_j$  to "[L2]"
                  Remove duplicate ("[L2]")
                  NB_Label_Used = NB_Label_Used-1
                End if
              End if
            End for
          End for
        End if
      End if

      Clear ("[L1]")
      Replace "[L1]" by "[L2]"
      Clear ("[L2]")
    End While
  End Procedure

```



Iteration 1: $L1 = \{11\}$



Iteration 2: $L1 = \{11, 7, 9, 17, 16\}$

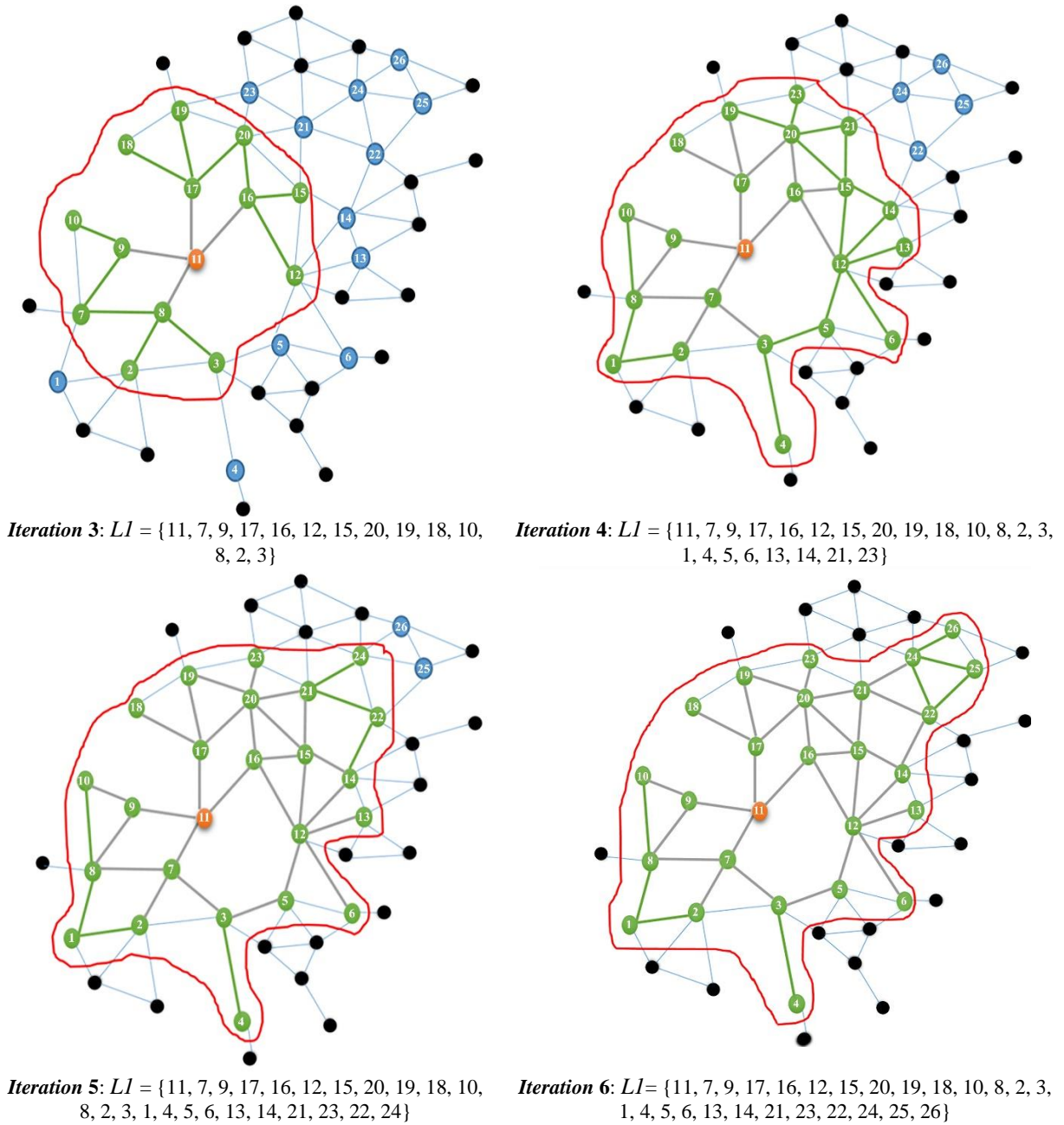


FIGURE 3.9 - Illustration of labelling process

ALGORITHM 3.1 results a graph where each set of vertices with the same properties has the same label. Finally, we traverse all graph vertices to assign a color for each label, and then we create a segment from each set of vertices having the same label as shown in the (FIGURE 3.10).



FIGURE 3.10 - Segmentation process

FIGURE 3.10 shows the set of segments produced by applying our method on 3D point cloud. image (a) shows the original point cloud, image (b) presents labelled graph where each group of connected vertices has a different color and image (c, d) shows the segments

3.4. Face detection

Once the graph vertices are labelled and segmented into skin and non-skin regions, we affirm whether a skin region has the face properties based on the most important characteristics of a human face such as eyes, nose, and mouth. We define these characteristics as a set of non-skin vertices surrounded by a set of skin vertices as shown in (FIGURE 3.11). To do so, we refine the list of faces candidates using some proposed refinement rules. These rules can be classified into three categories: refinement based on the number of non-skin regions contained in candidate region, refinement based on the size and ratios and refinement based on distances between face parts.

If the constructed graph is totally connected, the computational time complexity of face detection algorithm is $O(N^2)$, where N is the number of nodes in the graph.

3.4.1. Refinement based on number of non-skin regions

A skin region must contain a determined number of non-skin regions in order to be considered as a face. Therefore, candidate face regions are defined using the following conditions:

- For each face region R , \exists At least three non-skin regions B_1 , B_2 and B_3 such as $B_1 \sqsubset R$, $B_2 \sqsubset R$, $B_3 \sqsubset R$ and the number maximum of non-skin regions in R should be less than or equal of $[NBFormsMax]$.

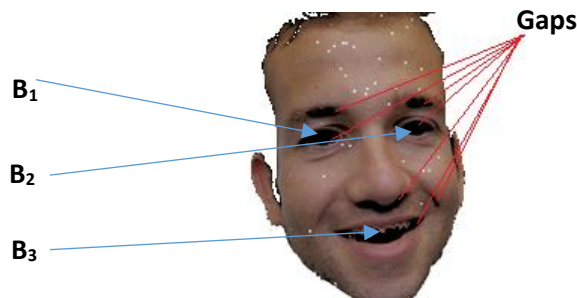


FIGURE 3.11 - Skin region-containing gaps

ALGORITHM 3.2 presents the procedure of defining the set of non-skin regions in a given candidate region.

Algorithm 2.2: *GapLocator*

Procedure *Check_Boundary*

Input:

Non-Skin Region S_b

Output:

Label of Skin region Container

Let " $[L]$ " an empty list of vertices

Let " $[R_v]$ " a list of vertices = $\{v\} / v \in S_b$ and $\nabla(N(v)) = 0$

While (*Card* (R_v) $\neq \nabla(R_v)$ or *Card* (R_v) increased)

Begin

For each (Vertex x in " $[R_v]$ ")

If ($x \in \mathbf{B}(\mathbf{G})$) **then**

 Add $N(x)$ to " $[L]$ "

if " $[L]$ " $\neq \emptyset$ replaces x with the vertices of " $[L]$ "

End if

End for

End While

If (*Card* (R_v) = $\nabla(R_v)$) **Return** Label of any element

Else return -1

End procedure

This algorithm traverses all gaps and specifies for each one the corresponding skin region.

3.4.2. Refinement based on size

The size of the human face has specific characteristics where there is a proportional relation between its height and width. This proportional relation obeys the golden ratio of human faces (FIGURE 3.13), which is approximately 1.6 [51]. Meanwhile, the candidate regions that are smaller than the minimum size are eliminated. In order to determine the size of a given candidate region, we define the minimum oriented bounding box of candidate region vertices as shown in (FIGURE 3.12). Once, the box width, height, and depth are determined, the following rules should be verified:

1. $GR - th \leq \frac{HB}{WB} \leq GR + th,$
2. $HB \geq h_{min}$ and $WB \geq w_{min},$

where HB is the height of bounding box, WB is the width and th is a parameter controlled by the user in order to increase the range of the face ratio and defined as an error margin.

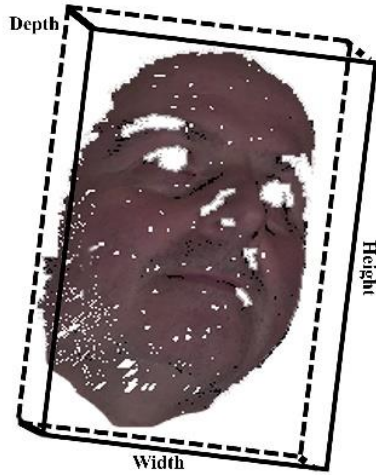


FIGURE 3.12 - Minimum bounding box of candidate region

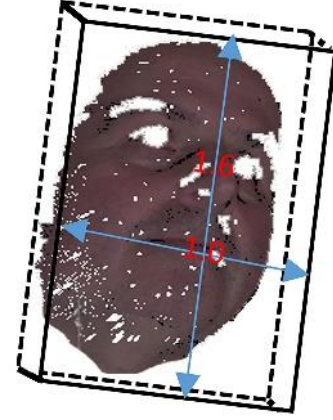


FIGURE 3.13 - Human face golden ratio

3.4.3. Refinement based on the distance between face parts

For each face region R , \exists At least three non-skin regions B_1 , B_2 and B_3 such as $B_1 \sqsubset R, B_2 \sqsubset R, B_3 \sqsubset R$ such as:

- $d(\theta(B_1), \theta(B_2)) - d(\theta(B_2), \theta(B_3)) \leq \varepsilon$,
- $d(\theta(B_1), \theta(B_2)) - d(\theta(B_1), \theta(B_3)) \leq \varepsilon$,
- The sum of ratios $\frac{\varphi(B_1)}{\varphi(R)}$, $\frac{\varphi(B_2)}{\varphi(R)}$ and $\frac{\varphi(B_3)}{\varphi(R)} \in [0.05, 0.2]$,

where $\varphi(R)$ represents the area of the region R , $\theta(R)$ is the gravity center of the region R , and ε is a distance threshold. FIGURE 3.14 shows the selection of face characteristic based on the distance between non-skin regions. Note that the values of interval $[0.05, 0.2]$ are chosen based on the observations done in our experiments.

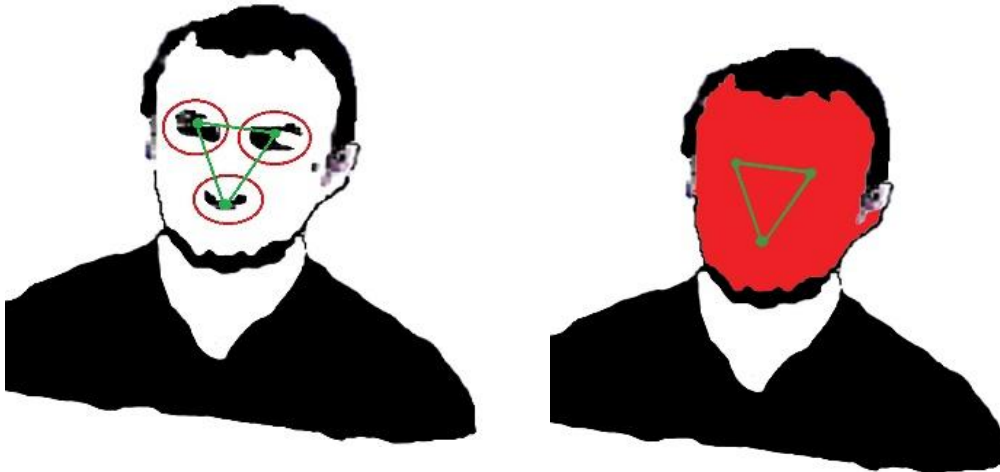


FIGURE 3.14 - Distance between face parts

3.5. Experimental results

In our experiments, we used a sample of 145 colored objects and 250 colored images from existing datasets such as “Helen” dataset [49], “Faces in the Wild” dataset [52], “Caltech” dataset [53] and others. The selected objects and images containing faces of different skin color, ages, expressions, and positions. In order to evaluate our method, we show visual and statistical results for both 3D point clouds, meshes and 2D images.

3.5.1. 3D point clouds and 3D meshes

In this section, we show the efficiency and robustness of our proposed method by applying it on many colored 3D point clouds models and 3D meshes. We will show and discuss the influence of different parameters on our approach such as minimal face size (h_{min} , w_{min}) presented in section 3.4.2, K -NN threshold k , the parameter $NBFormsMax$ existing in section 3.4.1, the parameter th defined in section 3.4.2, and the distance threshold ε presented in section 3.4.3.

FIGURE 3.15 presents the results of our skin detection model with a colored point cloud with the following parameters values ($k=0.01363$, $h_{min}=5$, $w_{min}=8$, $NBFormsMax=6$, $th=0.5$, $\varepsilon=0.6$). Images (a, c and e) show the original 3D point clouds, images (b, d and f) present the detected skin regions where the black regions are not skin.

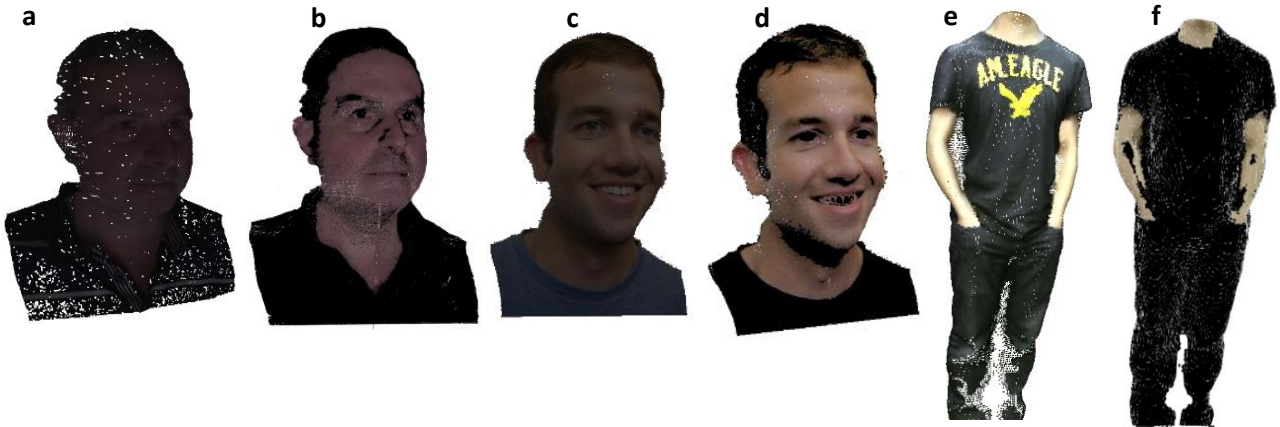


FIGURE 3.15 - Skin detection on 3D colored point clouds

FIGURE 3.16 presents the influence of the neighboring distance k (K -NN threshold) on 3D point clouds. Images (a) shows the original 3D point clouds. Image (b) shows skin regions containing gaps by using same parameters presented in (FIGURE 3.15). Image (c) presents the influence of k with the value 1. One can see that the number of gaps decreases when we increase the neighboring distance due to neighbors' information.

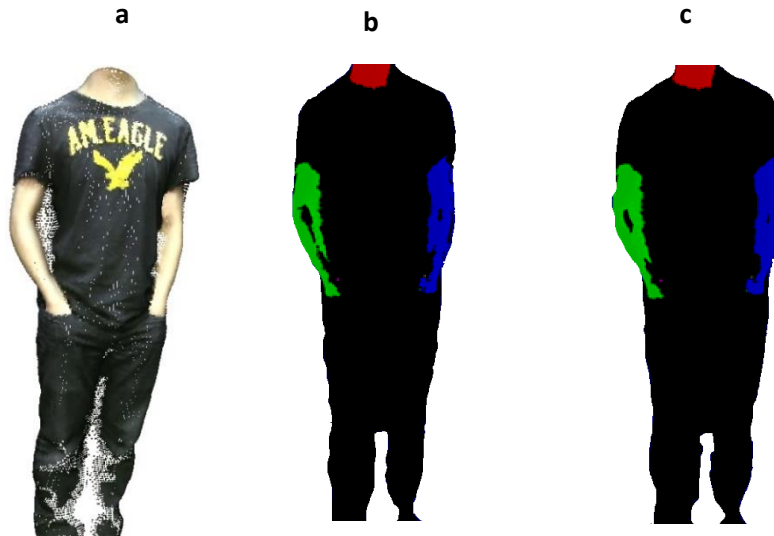
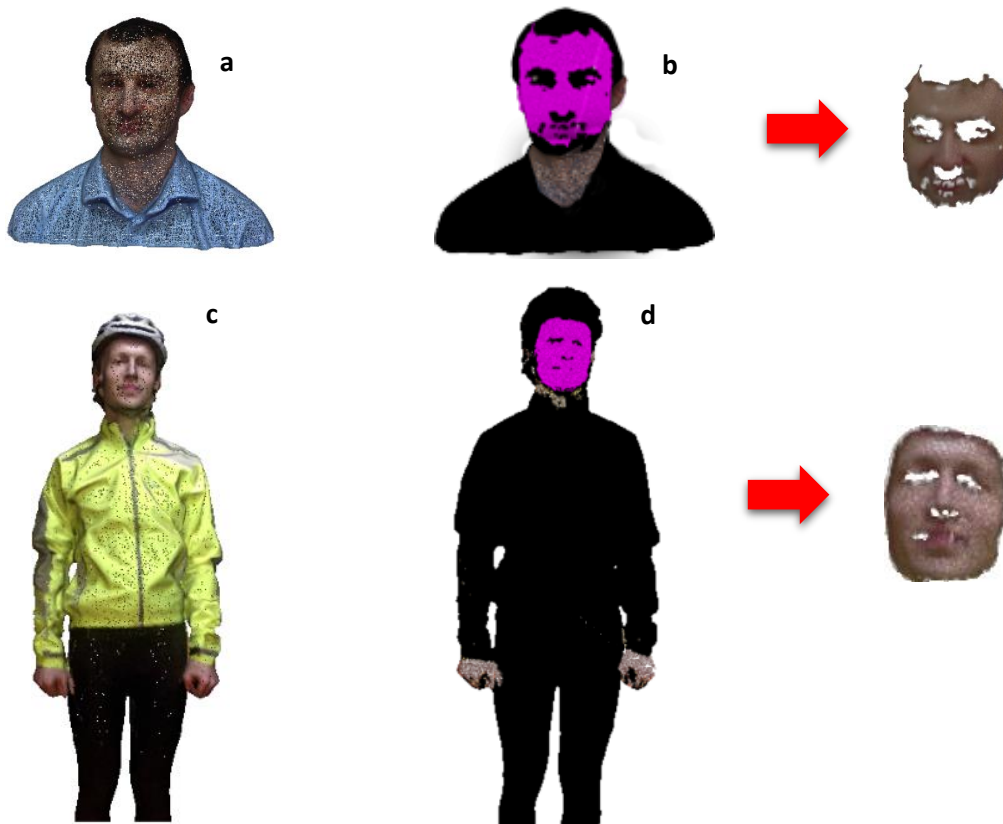


FIGURE 3.16 - Influence of vertices neighbors on labelling process

FIGURE 3.17 presents the results of our face detection method applied on a colored 3D point cloud. Images (a, c, e, g, i and n) show the original 3D point cloud, images (b, d, f, h, m, o and p) present the detected faces using our method with the following parameters ($k=0.01363$, $h_{min}=5$, $w_{min}=8$, $NBFormsMax=6$, $th=0.5$, $\varepsilon=0.6$). The fuchsia areas on the 3D point clouds object represent the detected faces. We can visually confirm the face detection. Images (L, m) demonstrate the efficiency of our method with different skin color. The image (p) presents an additional face detected, which is a failure case. This failure case is due to the characteristics of human face that are verified in this skin region.



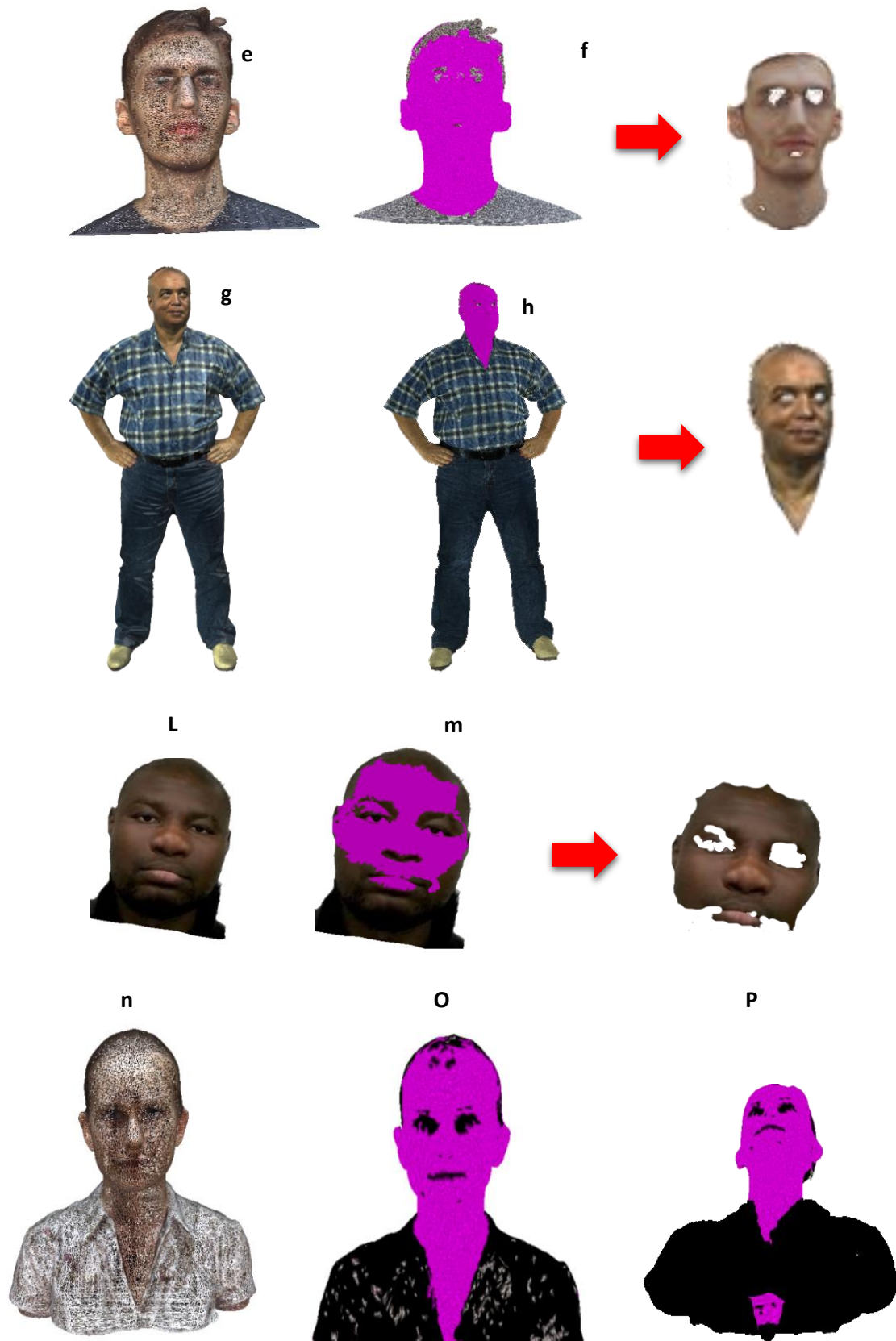


FIGURE 3.17 - Detected face on 3D colored point clouds

FIGURE 3.18 presents the result of our face detection method applied on a 3D colored point clouds with multi faces and expressions. We can visually confirm the detection of all faces in these point clouds.

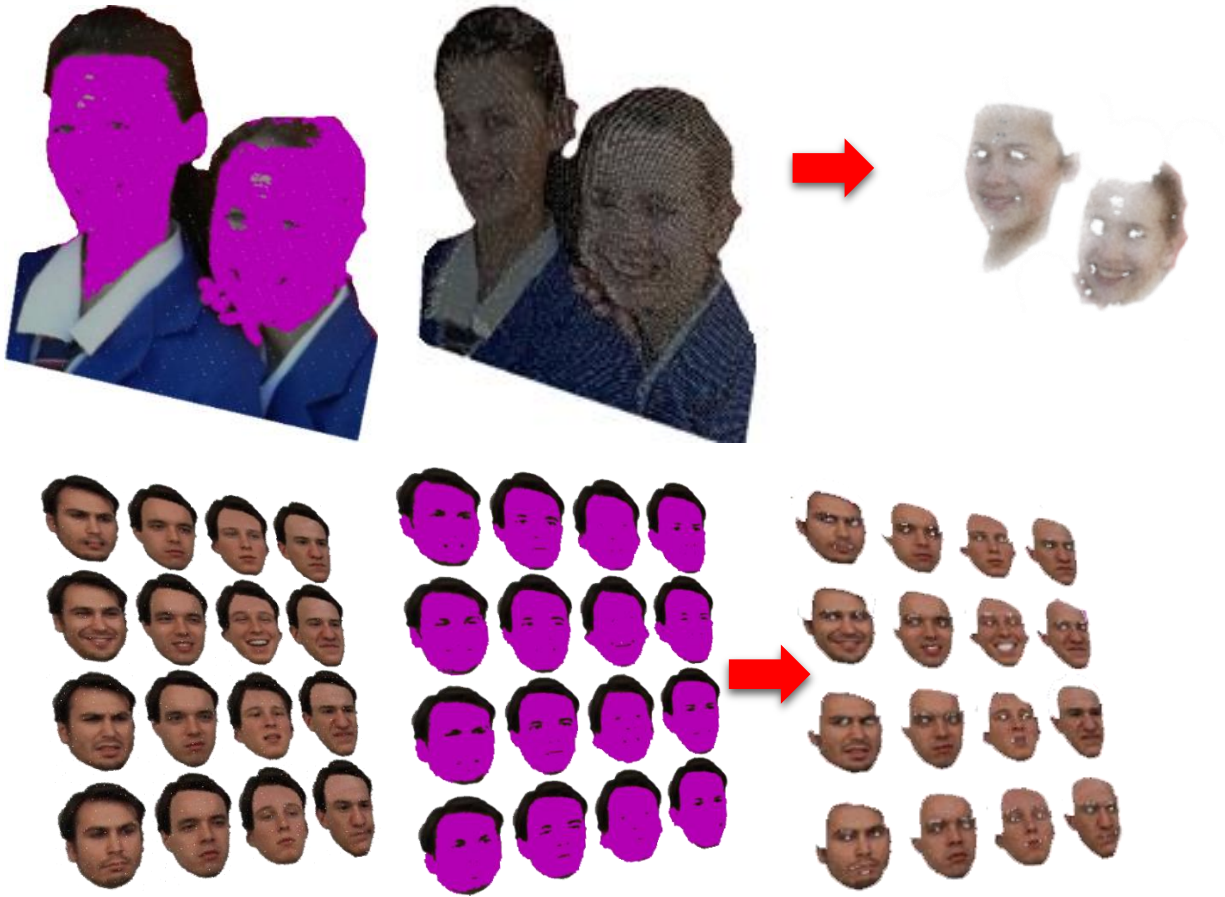


FIGURE 3.18 - Detected Faces with different expressions

FIGURE 3.19 presents the result of our face detection method applied on a noisy 3D colored point clouds. One can see that our method produces good results with noisy surfaces.



FIGURE 3.19 - Detected face on a noisy 3D colored point cloud

After presenting a sample of visual results of our proposed method on 3D meshes and 3D point clouds, the (TABLE 3.3) shows some statistical results of our experiments in term of detection accuracy.

TABLE 3.3 - Results obtained from 3D point clouds and 3D meshes

Objects count	145
Number of all faces	385
Number of truly detected faces	341
False positive rate (FPR)	8.8%
False negative rate (FNR)	3.63%
Accuracy	87.57%

The accuracy of our proposed method is measured as following:

$$Accuracy = 100 - (FPR + FNR), \quad (3.10)$$

where FPR (False Positive Rate) and FNR (False Negative Rate) can be computed as follows:

$$FPR = \frac{\text{number of false alarms}}{\text{number of detected faces}} \times 100, \quad (3.11)$$

and

$$FNR = \frac{\text{number of missed faces}}{\text{number of all faces}} \times 100. \quad (3.12)$$

In our experiments, we noticed that the variation in the parameters $NBFormsMax$ and th can affect the accuracy of our system. The following charts shows the influence of these parameters on detection accuracy observed on 50 objects of our dataset.

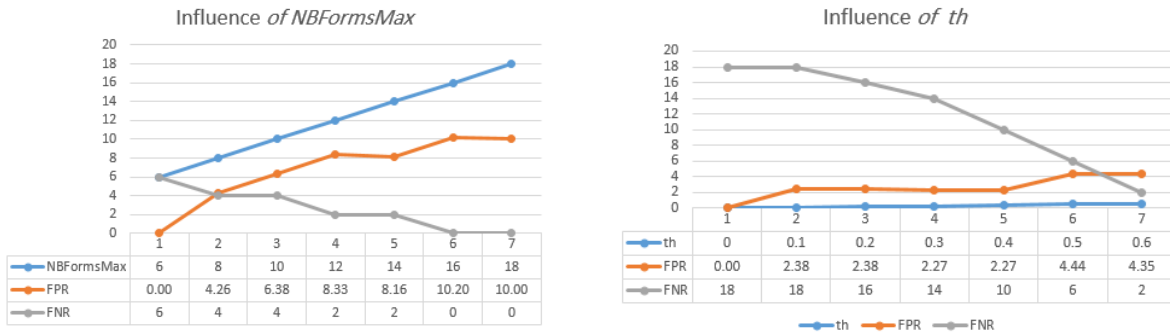


FIGURE 3.20 - Influence of parameters $NBFormsMax$ and th on accuracy

Furthermore, when the value of parameters (h_{min} , w_{min}) increase, the number of missed faces relatively increases. In contrast, when the value of parameter ε increases, the number of false alarms relatively increases.

3.5.2. 2D images

As mention above, our method can deal with any type of data that can be represented by a weighted graph such as 2D colored images and 3D colored point clouds. The figures below show the result of applying our method on 2D colored images. Note that a rectangle is drawn around the detected face region as shown in (FIGURE 3.21).

FIGURE 3.21 presents the results of our face detection method applied on 2D images with the following parameters values ($k=1$, $h_{min}=5$, $w_{min}=8$, $NBFormsMax=6$, $th=0.5$, $\varepsilon=0.6$). Images (a_1 , b_1 , c_1 , d_1 and e_1) show the original 2D images, images (a_2 , b_2 , c_2 , d_2 and e_2) present the detected skin regions and images (a_3 , b_3 , c_3 , d_3 and e_3) present the detected faces by our method. The image (b_3) presents a face detection in an image with a complicated background but image (d_3) shows a failure case where an additional face is detected due to the color of this region that is interpreted as skin color and verify all face rules.



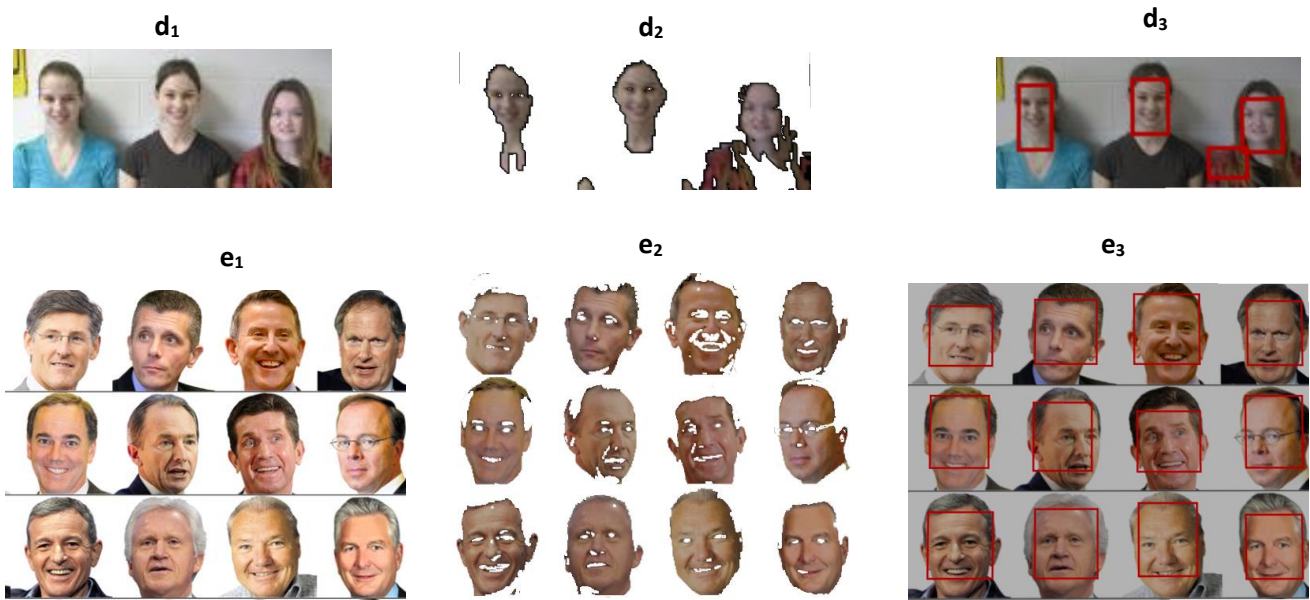


FIGURE 3.21 - Detected faces on colored 2D images

FIGURE 3.22 presents the results of our face detection method on an image containing many faces for a person with different expressions. Image (a) shows the original image and Image (b) presents the accuracy of our method where all faces are detected.

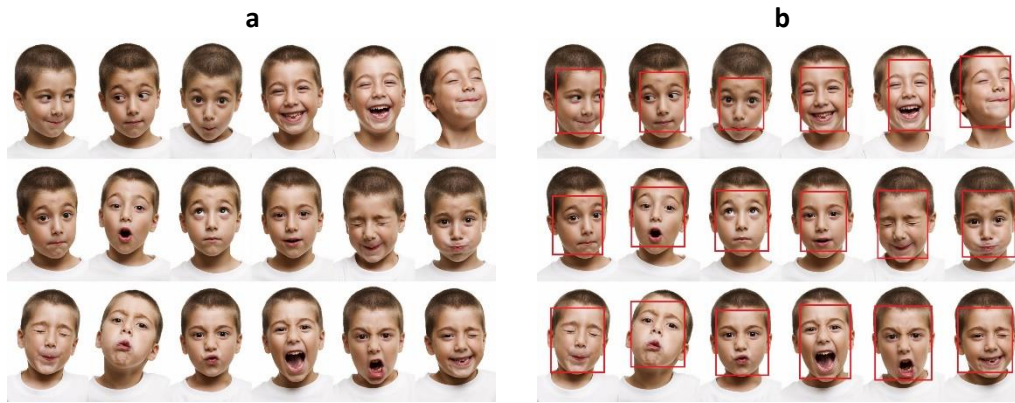


FIGURE 3.22 - Detected Faces from an image with different expressions

Finally, a numerical summary of our experiments shown in the (TABLE 3.4).

TABLE 3.4 - Results obtained from 2D image

<i>Images count</i>	250
<i>Number of all faces</i>	304
<i>Number of truly detected faces</i>	278
<i>False positive rate (FPR)</i>	7.01%
<i>False negative rate (FNR)</i>	1.97%
<i>Accuracy</i>	91.02%

3.6. Comparison with the state-of-the-art

In this section, we compare our method with some related works. Firstly, note that our method deals with 3D colored point clouds, 3D meshes and 2D images based on skin color. In contrast, most of the methods described in the state-of-the-art section deal with 2D images or 3D meshes that lie on geometry features. As another evaluation of our work, we compare the accuracy of our method to some methods described in the literature. The (FIGURE 3.23) shows the numerical data:

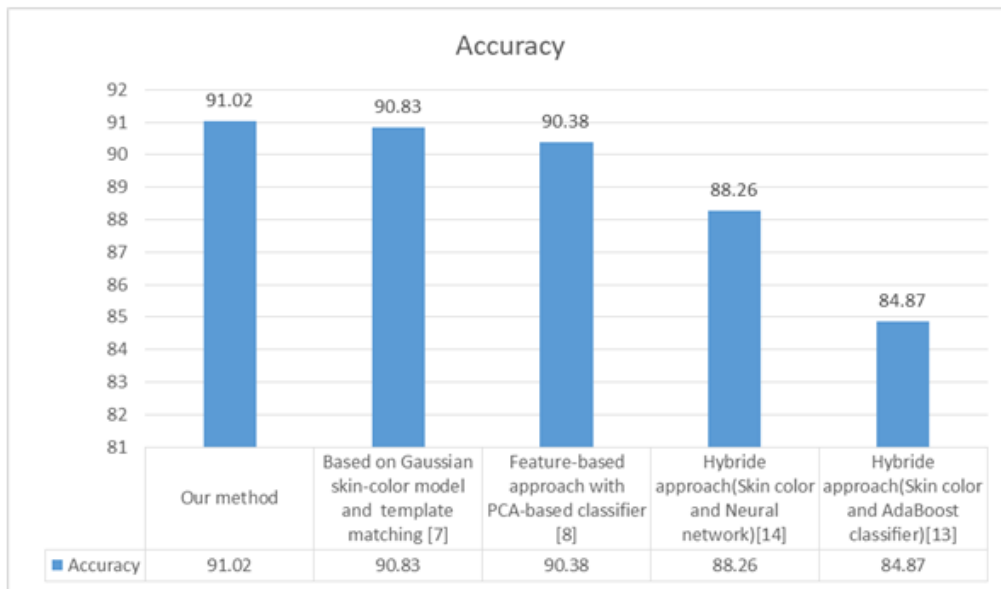
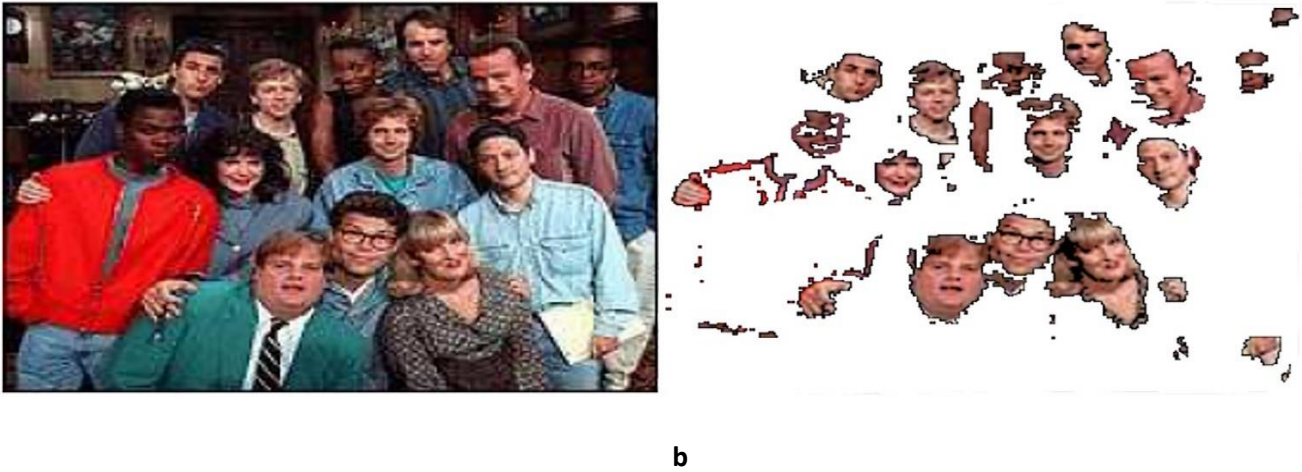


FIGURE 3.23 - Accuracy comparison with other methods

In addition to the statistical evaluation, we choose some visual examples of failure case presented by some related works and we apply our method on the same images. FIGURE 3.24 presents a comparison of the skin detection with [54]. One can see that the result of our method is better than the method proposed in [54] as shown in the (FIGURE 3.24). As shown in image (a) a non-skin color is interpreted as skin color. In contrast, it is detected correctly by our method as shown in image (b).



a



b
FIGURE 3.24 - Result obtained by [54] (a) and our method (b)

3.7. Conclusion

In this chapter, we have presented a new approach for skin and face detection on 3D colored point clouds using the weighted graph representation. To illustrate the robustness of our approach, we showed that our proposed method could be used to solve two signal-processing problems. The first one is related to the vertices classification based on the skin information. The second one is the face detection problem. We showed also that our method can deal with any data that can be represented by weighted graphs such as 2D images, 3D surfaces, and 3D point clouds. We presented then a comparison with some other related face detection works to prove the efficiency of our method. In the next chapter, we will present a new method for salient regions detection. The skin color detection and saliency detection will be used together in the last chapter in order to enhance the face detection process presented in in this chapter.

Chapter 4

EFFICIENT 3D MESH SALIENT REGIONS DETECTION

Summary

4. EFFICIENT 3D MESH SALIENT REGIONS DETECTION	57
4.1. Introduction	58
4.2. Saliency method	60
4.2.1. Surface modeling and normal estimation	60
4.2.2. Saliency first approach	61
4.2.2.1. Deviation factor	61
4.2.2.2. Local patch descriptor	62
4.2.2.3. Saliency computation	63
4.2.2.4. Non-local Saliency computation	64
4.2.3. Saliency second approach	65
4.2.3.1. Invariant patch descriptor	65
4.2.3.2. Single-scale saliency	67
4.2.3.3. Multi scale saliency	67
4.3. Experimental results	68
4.3.1. Comparison with the ground truth	68
4.3.2. Influence of different parameters	70
4.4. Comparison with the state-of-the-art methods	73
4.5. Comparison between the proposed approaches	76
4.6. Conclusion	76

Visual saliency is defined by the perceptual information that makes possible to detect specific areas that attract the human visual attention. In this chapter, we present two approaches for salient regions detection on 3D meshes using weighted graphs representation. In the first approach, we propose a novel 3D surface descriptor based on a local homogeneity measure. Then, we define the similarity measure between vertices using normal deviation similarities, a 2D projection height map, and the mean curvature. The saliency of a vertex is then evaluated as its degree measure based on the local patch descriptor and a height map. In the second one, we propose a novel 3D invariant surface descriptor based on Zernike coefficients to compute the shape saliency map. Then, a multi-scale saliency is calculated in order to improve the quality of the measured saliency and deal with noise. Furthermore, we show the robustness of our proposed methods through different experimental results. Finally, we present the stability and robustness of our methods with respect to noise.

4.1. Introduction

Salient regions are particular zones, which are distinguished from adjacent areas on a surface. Several saliency detection methods have been proposed for 2D images. However, 3D data acquisition technologies have known significant developments, which led to the acquisition of large amounts of data in the form of 3D meshes and point clouds.

Mesh saliency detection is an important pre-processing phase for many computer vision applications [55, 56, 57]. The segmentation of 3D objects based on salient regions detection has been widely used in many computer vision applications such as shape matching [55], objects alignments, 3D point clouds smoothing, and segmentation and recognition of face [56]. Among these applications, we can cite also the adaptive mesh simplification [57] that aims to maintain a better-perceived quality by simplifying regions with low saliency degrees.

The detection of saliency on 3D meshes has known a significant progress. Most of existing methods compute the saliency using the multi-scale computation [57] where they compute the saliency at multiple scale in order to use all of them to produce a robust saliency measurement and center-surround operators as explained in [55]. The challenge of these methods is to deal with noise. In [58], authors proposed to detect saliency using the local contrast and global rarity, which is robust against noise. They calculated square map of the height projections to describe the local shape surrounding a vertex. Then, local and global saliencies are combined together for each vertex to define the final visual saliency at different scales. However, the robustness of this method highly depends on parameter to obtain good results. [55] considered the regions as salient where the curvature of the vertex or patch is different from its neighbors. The curvature computation used in this method is sensitive to noise.

To handle the noise problem, we propose simple and efficient methods to detect salient regions on a 3D mesh using a weighted graph representation. The Surface roughness variation can be used as a metric for detecting salient region [59] where a vertex is considered as salient if it can be strongly differentiated from its neighbors.

Starting from this point, in the first approach, we propose to build a robust and rotational invariant surface patch descriptor based on the surface fluctuation information and a 2D height map to calculate the saliency degree at a given vertex. The combination of these two factors (deviation factor and height map) will enhance the saliency computation because it gives more information about the dissimilarity between vertices. To do so, we create a patch for each vertex according to its tangent plane. This patch is created by projecting vertices onto a 2D plane defined by the associated vectors. This projection leads to calculate a height map of the mesh vertices. To ensure that the saliency at a given vertex is calculated in term of all possible neighbor's information, we propose to compute a patch saliency in order to calculate the saliency degree at a given vertex. Finally, the saliency degree of a given vertex is computed from the patch saliency based on the entropies of the target vertex patch and patches associated to the neighborhood, and from a term containing the deviation factor, the height projections and the curvature. This procedure will help to assign comparable saliency values to geometrically similar vertices.

The novelty of our first method depends on the key points that follow. Firstly, a Delaunay graph based is constructed from the initial data, which guarantee a robust neighbouring modelling. A novel height map is generated by projecting the set of neighbours of a target vertex on a 2D plane. We define an efficient local patch descriptor that combines a height map with deviation factors representing the angles between the normal vectors of the vertices. We define also a new similarity measure between vertices according to the deviation factor and the projection height. Then, we compute a patch saliency according to its patches neighbouring based on the entropy measurement. According to this saliency with the similarity measure, we define the saliency degree at a given vertex. This computation will ensure that a vertex saliency degree on a given vertex depends on its neighbours and its patch neighbours. Finally, we

define and compute a non-local saliency measure by recursively increasing the number vertices neighbours. The following flowchart in (FIGURE 4.1) illustrates our method.

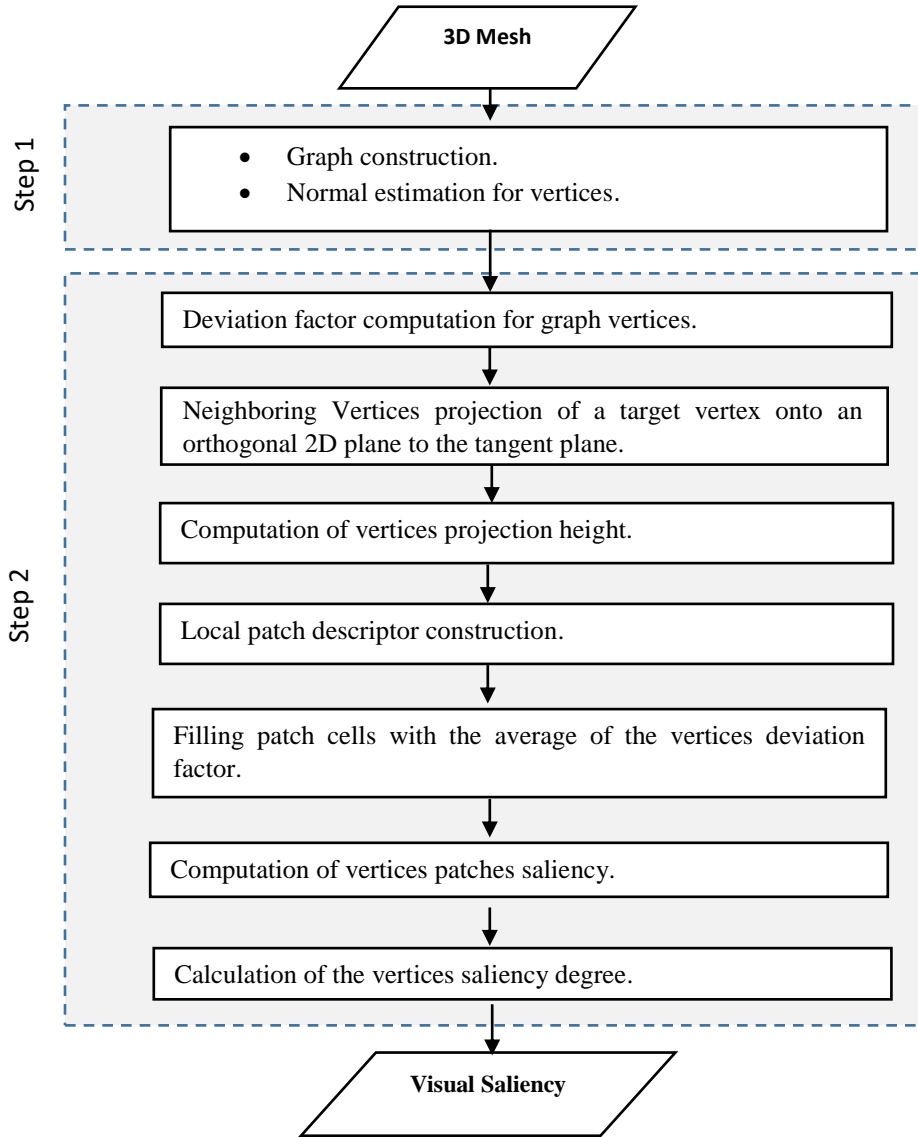


FIGURE 4.1 - Saliency computation steps – first approach

On the other hand, the novelty of the second method depends on the key points that follow. Firstly, a Delaunay graph based is constructed from the initial data, which guarantee a robust neighboring modelling. A novel 3D invariant surface descriptor based on Zernike coefficients is constructed. The Zernike polynomials are a set of functions that are orthogonal over the unit circle. They are useful for describing the shape of an aberrated wavefront in the pupil of an optical system. Several different normalization and numbering schemes for these polynomials are in common use [80]. In order to define this local patch descriptor for a given vertex v_i , we construct a rectangle of n cells centered at v_i of fixed length. Afterward, a height map is constructed by projecting v_i neighbors onto a 2D plan. Next, each cell of the patch is filled with the absolute value of the sum of the projections heights and zero if no projections occurs in it. Subsequently, the height map is converted into image. Hereafter, we compute the Zernike coefficients for the resulting image. Consequently, a vertex v_i is described by a list of Zernike coefficients.

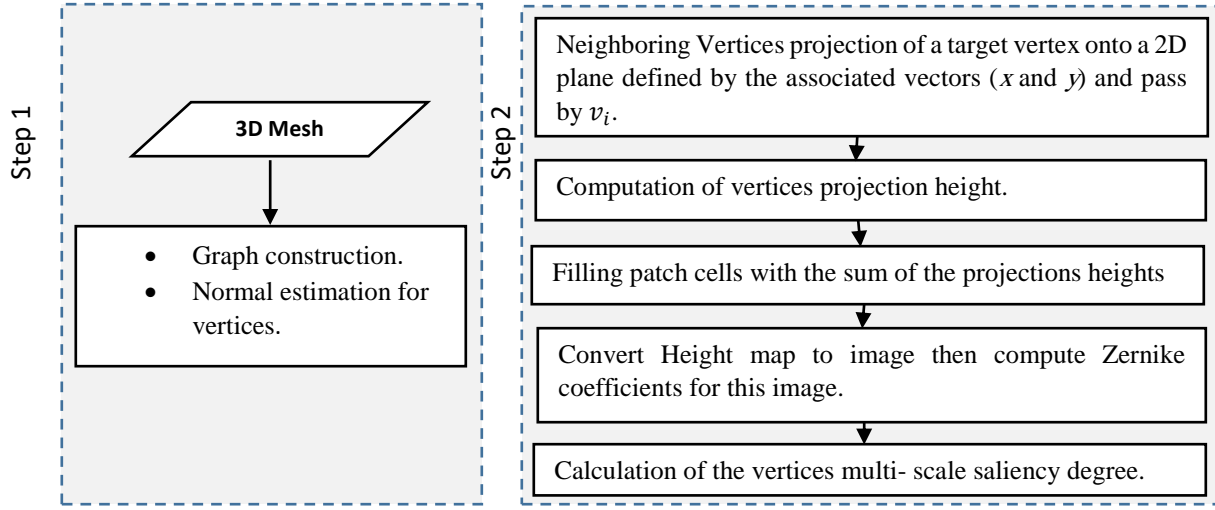


FIGURE 4.2 - Saliency computation steps – second approach

As we define the saliency computation on graph, our approaches can deal with any data that can be represented by a graph such as images, 3D point clouds, and 3D meshes.

4.2. Saliency method

4.2.1. Surface modeling and normal estimation

The first step in our approaches consists of modeling the 3D mesh surface. To do so, we consider the situation where a mesh can be viewed as a weighted graph $G = (V, E, W)$ where V is a finite set of vertices, E represents a finite set of edges $E \subset VXV$, and W a weight function that represents the similarity measure between vertices.

In order to construct the set of neighbors $D_n(v_i)$ of a given vertex v_i , we propose to use the concept of Delaunay triangulation [60]. A *Delaunay triangulation* for a set S of discrete points in a plane is a triangulation $DT(S)$ such that no point in S is inside the circumcircle of any triangle in $DT(S)$. Thus, working with the 3D mesh, we define circumscribed spheres from a given 3D surface point P_i such that no surface point is inside these spheres. Then, the set of points located on the circumscribed spheres borders P_j will be defined as the set of P_i neighbors. In other words, we define the list of P_i neighbors as the set of vertices located on the corners of all triangles (faces) sharing

P_i 's Neighbors as shown in (Figure 4.3). Thus, the set of neighbors $D_n(v_i)$ of a graph vertex v_i that represents a mesh point P_i is defined as all vertices v_j representing P_j .

Let (v_i, v_j) be the edge that connects two vertices v_i and v_j . Each vertex v_i is represented by its 3D coordinates $v_i = (x_i, y_i, z_i)^T$, its normal vector $N(v_i)$, and the directional vectors $x(v_i)$ and $y(v_i)$ that correspond to the 2D tangent plan estimated at v_i .

The normal $N(v_i)$ of a vertex v_i and the 2-directional vectors following the x - and y -axes are computed as the as the eigenvalues of the covariance matrix $Cov(v_i)$. To do so, we compute the gravity center and the associated covariance matrix at v_i .

- The gravity center is defined as:

$$\check{v}_i = \frac{1}{\text{card}(D_n(v_i))} \sum_{j \in D_n(v_i)} v_j \quad (4.1)$$

- Then, we define the covariance matrix at a vertex v_i as:

$$\text{Cov}(v_i) = \sum_{j \in D_n(v_i)} (v_j - \check{v}_i)(v_j - \check{v}_i)^T \in \mathbb{R}^{3 \times 3} \quad (4.2)$$

Moreover, $C_0(v_i)$ and $C_1(v_i)$ that estimate receptively the minor and major principal directions are computed using the covariance matrix.

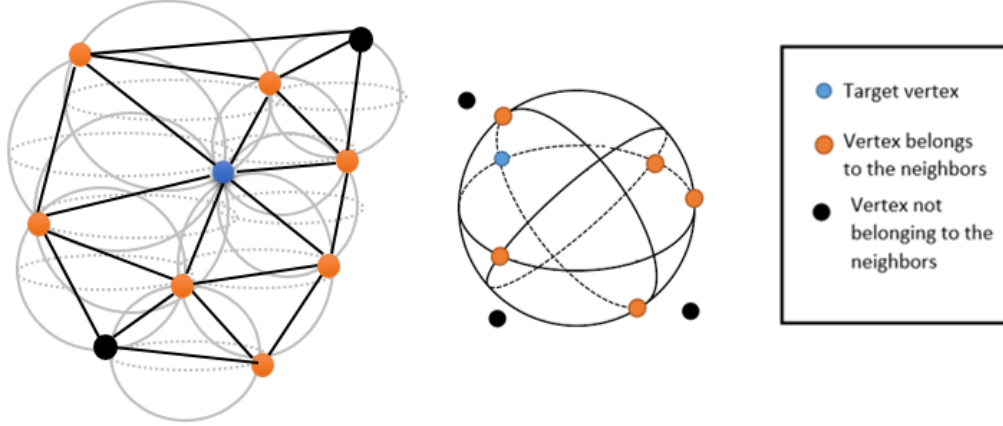


FIGURE 4.3 - Illustration of the vertex neighboring construction

4.2.2. Saliency first approach

After modeling the 3D surface and estimating the normal, the next step is to compute the deviation factor. Afterward, a local patch descriptor is constructed for each vertex. Successively, a height map is created. Finally, by computing the weight of edges, we can calculate the local saliency and the non-local saliency by changing the number of neighbors' recursively.

4.2.2.1. Deviation factor

Consider the 3D mesh surface modeled as a weighted graph. In order to compute the saliency at a vertex v_i , we propose to calculate the deviation factor between two vertices v_i and v_j that measures the fluctuation of the surface. Then, we construct patches by projecting the vertices onto a 2D plan and we fill the patches cells by the deviation factors values.

In order to measure the deviation factor $DF(v_i, v_j)$, we compute the angle between the vertices normal vectors using the following formula:

$$\alpha(v_i, v_j) = \arccos \left(\frac{(N(v_i) \cdot N(v_j))}{\sqrt{(\sum_{k=1}^3 N(v_i)(k) * N(v_i)(k)) * \sum_{m=1}^3 N(v_j)(m) * N(v_j)(m)}} \right) \quad (4.3)$$

Then, we define the deviation factor between v_i and v_j as following:

$$DF(v_i, v_j) = \frac{\angle(v_i, v_j)}{360} \times 100, \quad (4.4)$$

In the case where the deviation factor value is between two vertices is almost zero, we consider that these vertices are located on the same plane.

4.2.2.2. Local patch descriptor

Once we calculate the deviation factors between vertices, we build the patches that describe the local surface of vertices. These patches are constructed by projecting the given vertex neighbors onto an oriented 2D plan from principal directions. Thus, we present in this sub-section the patch construction process. Then, we compute the saliency degree at a vertex in term of these patches.

To define a patch for a given vertex v_i , we construct a square grid centered at v_i according to its tangent plane. The patch length $L(v_i)$ is then defined as [61]:

$$L(v_i) = \max_{v_j \in D_n(v_i)} (\|v_j - v_i\|_2^2), \quad (4.5)$$

where $\|\cdot\|_2^2$ represents the Euclidean norm.

Then, the set of vertices are projected onto a 2D plan defined by the associated vectors (x and y) as shown in (FIGURE 4.4). As a result, we obtain a list of vertices v'_i for each projected vertex v_i and defined as:

$$v'_i = [(v_j - v_i) \cdot x(v_i), (v_j - v_i) \cdot y(v_i)]^T, \quad (4.6)$$

Thus, the patch at a vertex v'_i is represented by a rectangle of n cells where the side length of each cell is $L(v'_i)/n$, with n a constant that depends on applications.

To enhance the saliency degree measurement, we propose to calculate the saliency at a vertex depending on a projection height $He(v_i)$ and its deviation factor. The projection height of v_i is defined as:

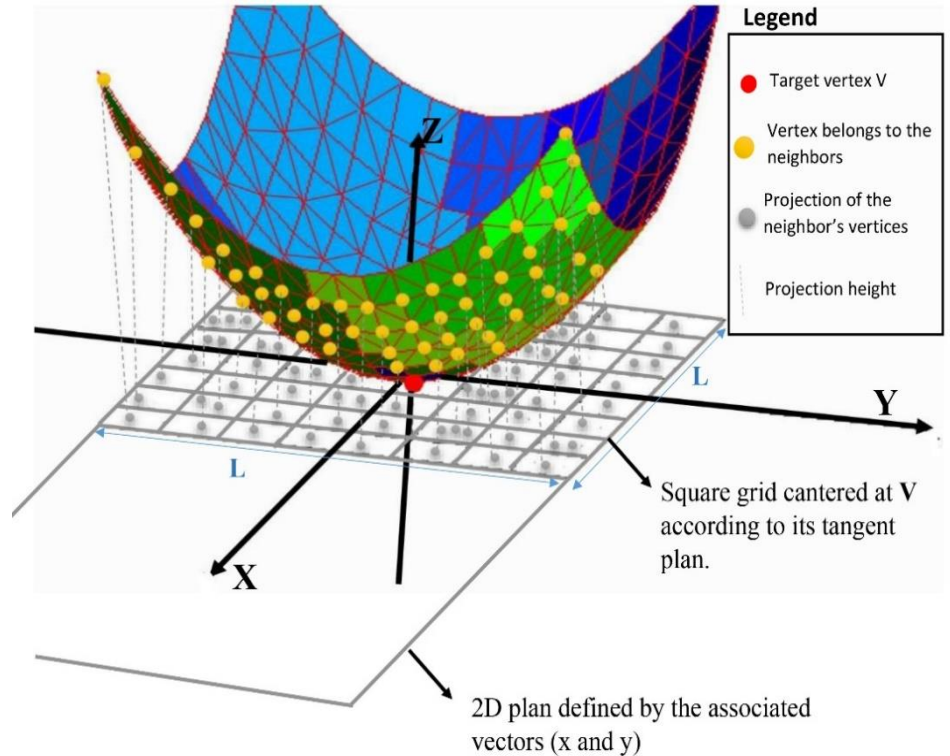


FIGURE 4.4 - Illustration of the patch construction and vertices projection. $c_0(v)$ and $c_1(v)$ are the principal directions at v . L is the patch length

$$He(v_i) = ||v_i - v'_i||_2^2, \quad (4.7)$$

4.2.2.3. Saliency computation

Once the patches are defined and filled, we compute the saliency degree at a vertex v_i . To do so, we compute the patch saliency $Patch_Saliency(v_i)$ using a similarity measure between the local patch descriptor $P(v_i)$ and the patches associated to v_i 's neighbors. Then, we calculate the saliency degree at v_i based on the $Patch_Saliency(v_i)$ and the weight functions of v_i and its neighbors.

Thus, we define the weight function $\omega: V \times V \rightarrow [0, 1]$ that represents the homogeneity measure between v_i and its neighbors as:

$$\omega(v_i, v_j) = e^{\frac{K(v_j) * DF(v_j, v_i) * (||He(v_j) - He(v_i)||)}{\sigma(v_i) * \sigma(v_j)}}, \quad (4.8)$$

where $K(v_j)$ represents the mean curvature [62] at v_j , the mean curvature of a surface describes locally the curvature of an embedded surface in some ambient space such as Euclidean space, and $\sigma(v_i)$ is the scale parameter and calculated as:

$$\sigma(v_i) = \max_{v_k \in D_n(v_i)} (||v_k - v_i||_2), \quad (4.9)$$

The saliency degree at v_i is computed based on the saliency of its patch $P(v_i)$. This saliency is calculated based on the entropy of $P(v_i)$. Thus, we propose to compute the probability $Prb(v_i)$ of a patch $P(v_i)$ as:

$$Prb(v_i) = \frac{C_x^i}{|V|}, \quad (4.10)$$

where C_x^i is the number of vertices of $P(v_i)$ with a deviation factor (the value of its corresponding patch cell) greater than x (x is a constant that depends on applications) and $|V|$ is the number of vertices belonging to $P(v_i)$ as shown in (FIGURE 4.5).

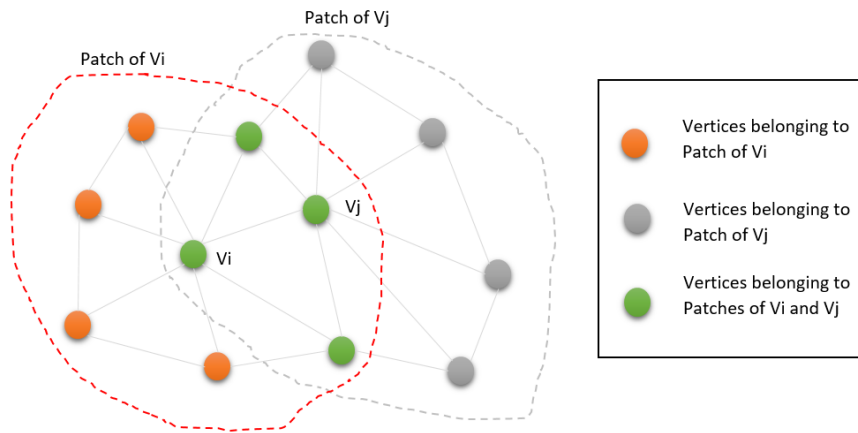


FIGURE 4.5 - Patch vertices.

The entropy of $P(v_i)$ that measures the dissimilarity between v_i and the set of vertices that belongs to $P(v_i)$ is calculated as following:

$$Entropy(v_i) = -Prb(v_i) \times \log_2 Prb(v_i), \quad (4.11)$$

Then, The saliency of $P(v_i)$ is computed as:

$$Patch_Saliency(v_i) = \frac{\sum_{j \in |V|} Abs(entropy(v_j) - entropy(v_i)) \times L(v_j)}{\sum_{j \in |V|} L(v_j)}, \quad (4.12)$$

where $L(v_j)$ represents the Euclidian distance between the neighboring vertex v_j and its 2D projection on the tangent plane at the vertex v_j .

Finally, we define the visual saliency at v_i as:

$$Saliency-degree(v_i) = \frac{\sum_{j \in |V|} Patch_Saliency(v_j) \times w(v_i, v_j)}{|V|} \quad (4.13)$$

The saliency degree is bounded between $[0, 1]$ where the degree 1 means that v_i is similar to its neighbors and 0 means the opposite.

Since the saliency measure at a vertex v_i is calculated in term of the patch saliency that contains v_i , this will ensure that all neighbors information of v_i are taken into consideration and will lead to a consistent saliency measure.

4.2.2.4. Non-local Saliency computation

In order to improve the quality of the measured saliency, we compute the saliency with a non-local graph where the neighboring function is calculated based on a multi-level Delaunay triangulation. Thus, to construct a non-local graph, we increase the number of neighbors of vertices. To do so, we recursively add to the list of v_i neighbors in $D_n(v_i)$ according to a level k that represents the neighbors of a vertex $v_j \in D_n(v_i)$. This nonlocal measure will ensure that the saliency at a vertex is computed using a larger number of neighbor's information resulting a consistent saliency measure. The term nonlocal means the non-locality of data defined on Euclidean domains (as images). This term is used to refer to the continuous case [63] where each element can interact with other elements in the domain (and not only adjacent ones). Thus, a graph constructed with two Delaunay levels will consider all gray vertices in the (FIGURE 4.5) as v_i 's neighbors.

Therefore, we can observe in (FIGURE 4.6) that a graph constructed with a small number of k Delaunay neighbor's leads to highlight large regions as silent, while a higher level will detect finest details. According to the ground truth map, we see that the non-local saliency detect finest regions better than the local one.

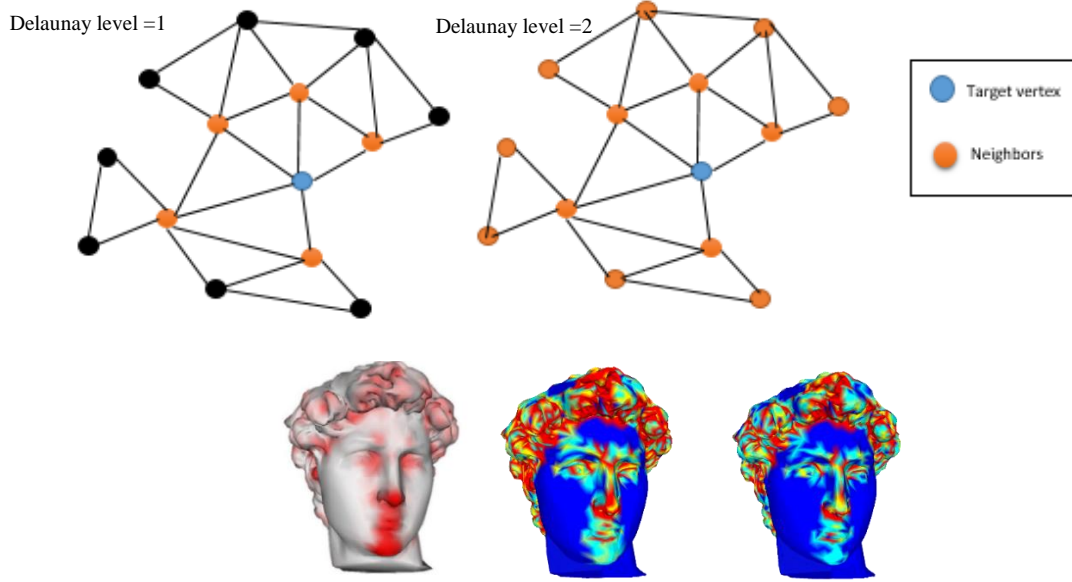


FIGURE 4.6 - Local and Non-local saliency. From left to right: ground truth map [64], saliency map with a local graph (level=1), non-local saliency map with a nonlocal graph (level=5)

4.2.3. Saliency second approach

In order to compute the saliency degree of vertices, we should first model the 3D surface and estimate the normal. Next, an invariant 2D descriptor is constructed for each vertex. Successively, a height map is created and the Zernike coefficients are computed based on this height map and the associated patch. Finally, by computing the weight of edges, we can calculate the single-scale saliency and the multi-scale saliency by changing the number of neighbors.

4.2.3.1. Invariant patch descriptor

Our main goal is to build patches that describe the local surface of vertices resulting in an approach of matching invariant under scaling, translation, rotation, and affine transformations. To do so, we construct these patches as following:

- First, to define a patch for a given vertex v_i , we construct a rectangle of n cells centered at v_i where the side length of each cell is $L(v_i)/n$, with n is a constant that depends on application and $L(v_i)$ is computed according to the bounding box of v_i neighbors using the formula (4.5).
- Afterward, a height map is constructed by projecting v_i neighbors onto a 2D plan defined by the associated vectors (x and y) and pass by v_i . As a result, we obtain a list of vertices v'_j for each projected vertex v_j by using the formula (4.6).
- In order to define the position of cells (in x - and y -axis) corresponding to the projected vertex v'_j of $v_j \in D_n(v_i)$, we use the following equation:

$$position^d(v'_j) = \left\lfloor \frac{v_j'^d}{\frac{L(v_i)}{n}} \right\rfloor \quad (4.14)$$

where $\lfloor \cdot \rfloor$ is the rounded integer and d represents the x or y coordinates.

At this stage, each cell of the patch is filled with the absolute value of the sum of the projections heights and zero if no projections occurs in it. The projection height of v_i is calculated using the formula (4.7).

Finally, the height map is converted into image where the value of each cell correspond to the intensity. Hence, the Zernike coefficients of which yield a rotationally invariant representation, we compute these coefficients for the resulting image. Consequently, a vertex v_i is described by a list of Zernike coefficients.

The Zernike polynomials constitute an orthogonal basis for functions defined on the unit disk. Each Zernike polynomial, V_p^q are defined over the domain $D = \{(p, q) | p \in \mathbb{Z}, q \in \mathbb{Z}^{\geq 0}, |q| \leq p, |p - q| \in \mathbb{Z}^{even}\}$ and has an associated order p and repetition q as following:

$$V_p^q(\rho, \theta) = R_p^q(\rho) e^{iq\theta} \quad (4.15)$$

where R_p^q is the radial polynomial defined as:

$$R_p^q(\rho) = \sum_{\substack{k=|q| \\ |p-q| \text{ even}}}^p \frac{(-1)^{\frac{p-k}{2}} \frac{p+q}{2}!}{\frac{p-k}{2}! \frac{k-q}{2}! \frac{k+q}{2}!} \rho^k \quad (4.16)$$

In order to get the Zernike coefficients of a function $f(x, y)$, we apply:

$$z_p^q(f) = \frac{p+1}{\pi} \iint_{x^2+y^2 \leq 1} (\overline{V}_p^q)(x, y) f(x, y) dx dy \quad (4.17)$$

where (\overline{V}_p^q) represents the complex conjugate of V_p^q .

The Zernike polynomials form a basis upon which an image f can be projected. The Zernike moments of the image is the result of this projection, the magnitudes of which are invariant to rotation. In practice, each of the Zernike basis functions is denoted as a set of discrete samples on a $k \times k$ grid. Thus, the projection of an image f onto the Zernike basis function \overline{V}_p^q is defined as:

$$z_p^q(f) = \frac{p+1}{\pi} \sum_{(x,y) \in S} (\overline{V}_p^q)[x, y] f(x, y) \quad (4.18)$$

where S is the region centered at $[C_x, C_y] = [\frac{k}{2}, \frac{k}{2}]$ and defined as :

$$S = \left\{ [x, y] \sqrt{(x - c)^2 + (y - c)^2} \leq \frac{k}{2} \right\} \quad (4.19)$$

4.2.3.2. Single-scale saliency

Once the patches are defined and the Zernike coefficients are computed, we compute the saliency degree at a vertex v_i . To do so, we compute the weight of edges connecting v_i with its neighbors and the angles between normal of v_i and its neighbor's normal.

Given two Zernike coefficient C_1 and C_2 . We define the distance *dit* between C_1 and C_2 , similar to Euclidian distance, as following:

$$dit(C_1, C_2) = \sqrt{(real(C_2) - real(C_1))^2 + (imaginary(C_2) - imaginary(C_1))^2} \quad (4.20)$$

Let $D_Z(v_i, v_j)$ be the distance between the two sets of Zernike coefficients describing the vertices v_i and v_j and defined as:

$$D_Z(v_i, v_j) = \frac{\sum_{k=1}^n dit(Z^k(v_i), Z^k(v_j))}{n} \quad (4.21)$$

where $Z^k(v_j)$ is the K^{iem} Zernike coefficient.

Thus, we define the weight function $\omega: V \times V \rightarrow [0, 1]$ that represents the homogeneity measure between v_i and its neighbors as:

$$\omega(v_i, v_j) = e^{-\frac{D_Z(v_i, v_j)}{\sigma(v_i) * \sigma(v_j)}} \quad (4.22)$$

where $\sigma(v_i)$ is the scale parameter and calculated using the formula (4.9).

Finally, we define the single-scale saliency at v_i as:

$$Single-Scale-Saliency(v_i) = \frac{\sum_{j \in |V|} \alpha(v_i, v_j) \times w(v_i, v_j)}{|V|} \quad (4.23)$$

where $\alpha(v_i, v_j)$ is the angle between normal of vertices v_i and v_j .

Since the saliency measure at a vertex v_i is calculated in term of Zernike coefficients describing v_i and the Zernike coefficients describing its neighbors and angles between normal, this will ensure that all neighbors information of v_i are taken into consideration and will lead to a consistent saliency measure.

4.2.3.3. Multi scale saliency

In order to improve the quality of the measured saliency and cope with noise, we compute the saliency at different scales. To do so, we vary the Delaunay neighboring level to increase the number of vertex neighbors. We consider four different level 2, 4, 6 and 8. Then, we calculate the single-scale saliency map for each considered level. Before merging the obtained saliency maps, we calculate the Pearson Correlation Coefficient that measures the strength and direction of the relationship between the saliencies on each map.

To do so, at a given scale k , we compute for a given vertex v_i the saliency value and the average of saliency degrees of its neighbors. Afterwards, the Pearson Correlation coefficient [81] known as a measure of the linear correlation between two variables is computed as following:

$$r = \frac{\sum_{k=1}^4 (S_k(v_i) - \bar{S}_k(v_i)) (AS_k(v_i) - \overline{AS}_k(v_i))}{\sqrt{\sum_{k=1}^4 (S_k(v_i) - \bar{S}_k(v_i))^2} \sqrt{\sum_{k=1}^4 (AS_k(v_i) - \overline{AS}_k(v_i))^2}} \quad (4.24)$$

where $S_k(v_i)$ is the Single Scale Saliency of v_i at scale k and $AS_k(v_i)$ is the average Single Scale Saliency of v_i neighbors.

By weighting the saliency of each node by the Correlation Coefficient, we can obtain a robust multi-scale saliency map that considers the disparity of the saliency at each scale. It is computed as follows:

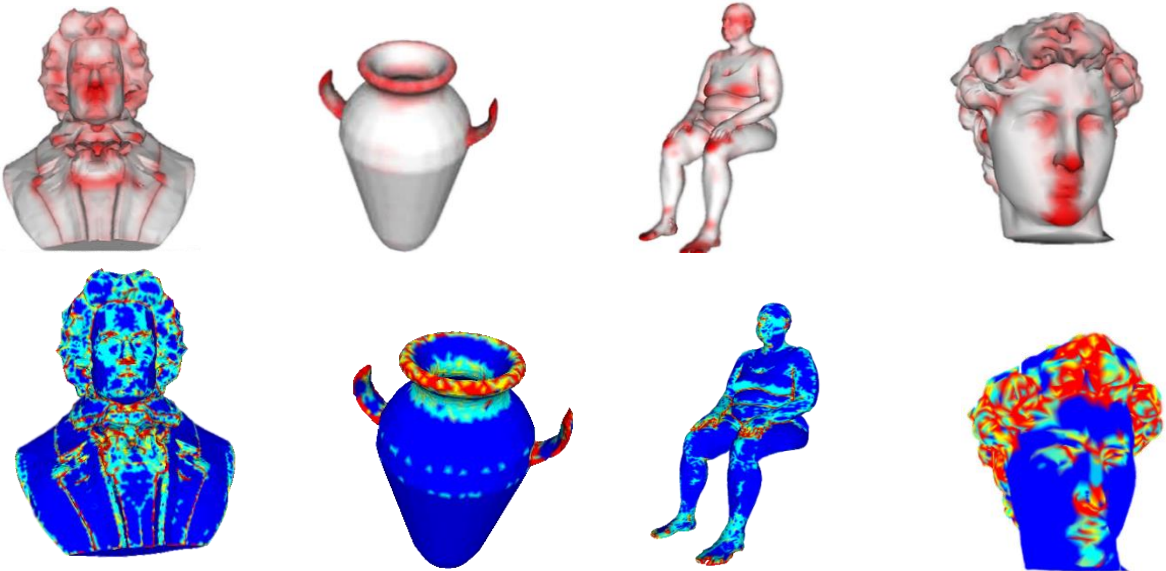
$$Multi - Scale - Saliency(v_i) = r * Single - Scale - Saliency(v_i) \quad (4.25)$$

4.3. Experimental results

In this section, we evaluate the performance of our methods by applying them on 3D meshes, 3D point clouds, and noisy 3D meshes. We show firstly the robustness of our approaches through a comparison with the saliency truth map. Then, We will show and discuss the influence of different parameters on our approach such as graph Delaunay-level k , the variable x in equation 3.9 that defines the deviation factor threshold in patches entropy calculation, and the number of patch cells n .

4.3.1. Comparison with the ground truth

To evaluate the performance of our method, we compare our results on several meshes from the SHREC 2007 with their ground truth saliency. This ground truth was obtained from Chen et al.'s user study [64] by asking users to select points that other users can select them. They applied then a regression analysis to construct an analytical model to predict which 3D point is salient. FIGURE 4.7 and 4.8 presents our results compared with ground truth maps. One can see that our results corresponds to the ground-truth saliency where we detect the most salient regions.



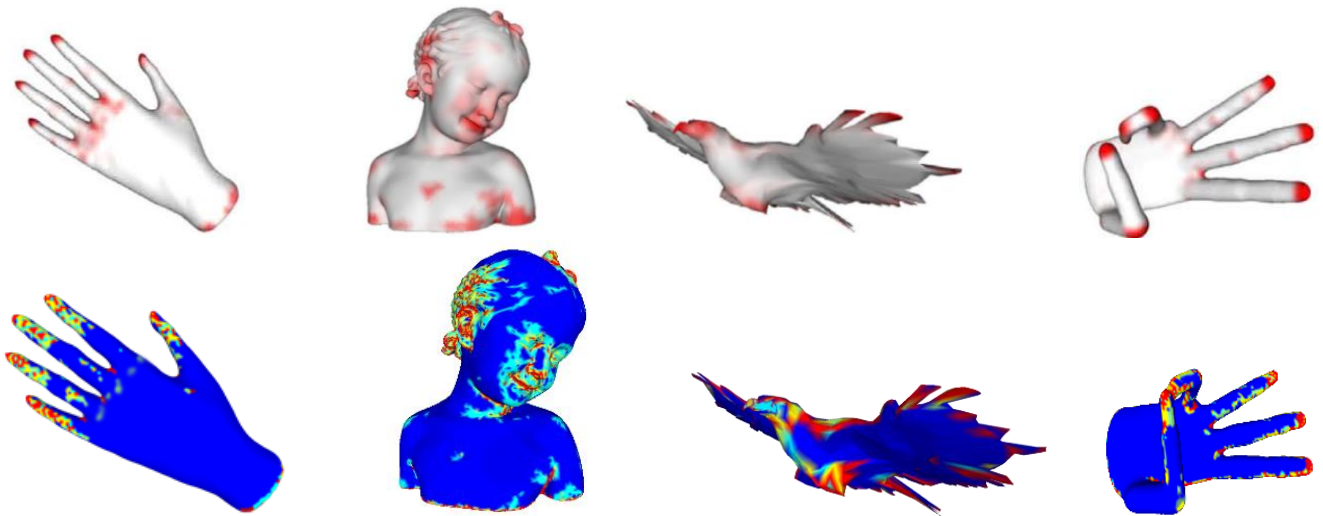


FIGURE 4.7 – Saliency first approach maps with ground truth maps

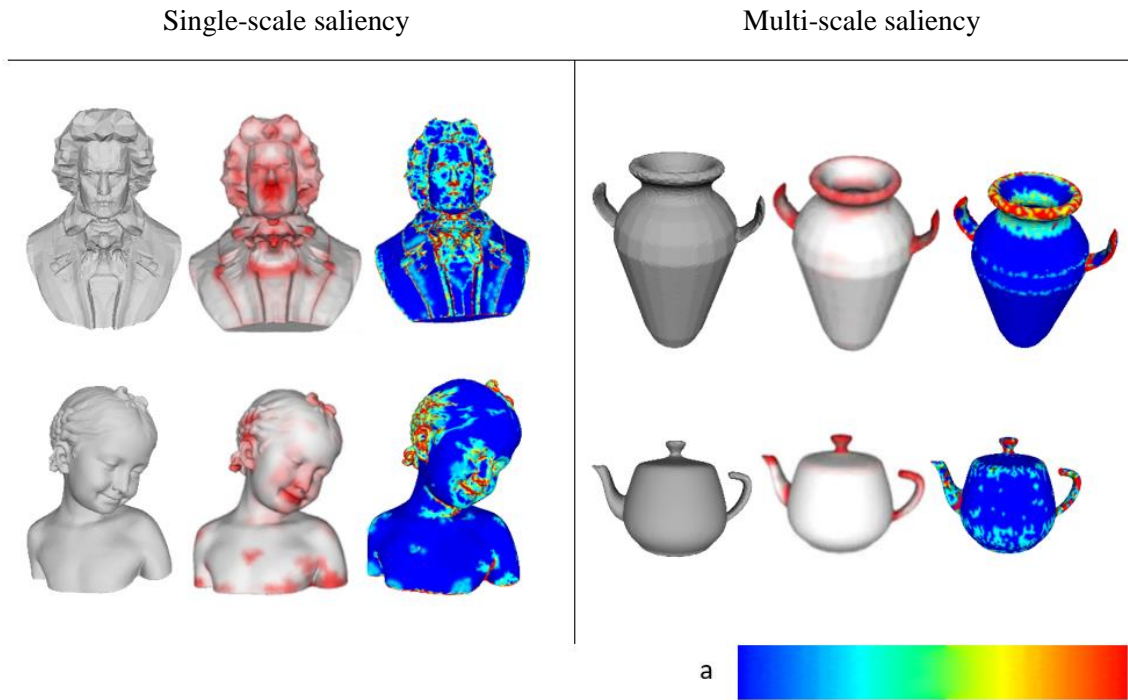


FIGURE 4.8. Saliency second approach map. From left to right: original 3D mesh, ground truth map, saliency map. Finally, (a) is the color map

4.3.2. Influence of different parameters

In (FIGURE 4.9), we present the results on some 3D meshes. Images (a_1 , b_1 and c_1) show the original 3D meshes, images (a_2 , b_2 and c_2) present the detected salient regions with $k=1$, $n=9$ and $x=9$. Images (a_3 , b_3 and c_3) display the salient regions detected using our method with $k=1$, $n=9$ and $x=12$, Images (a_4 , b_4 and c_4) display the salient regions detected using our method with $k=1$, $n=9$ and $x=15$. Image (a_0) shows the color map.

We notice that the number of salient regions increases when we decrease the deviation factor threshold. Practically, the similarity among the vertices relatively increases when the deviation factor threshold increases. We can observe clearly the influence of the deviation factor threshold in figures (b_3 , b_4) and (c_3 , c_4).

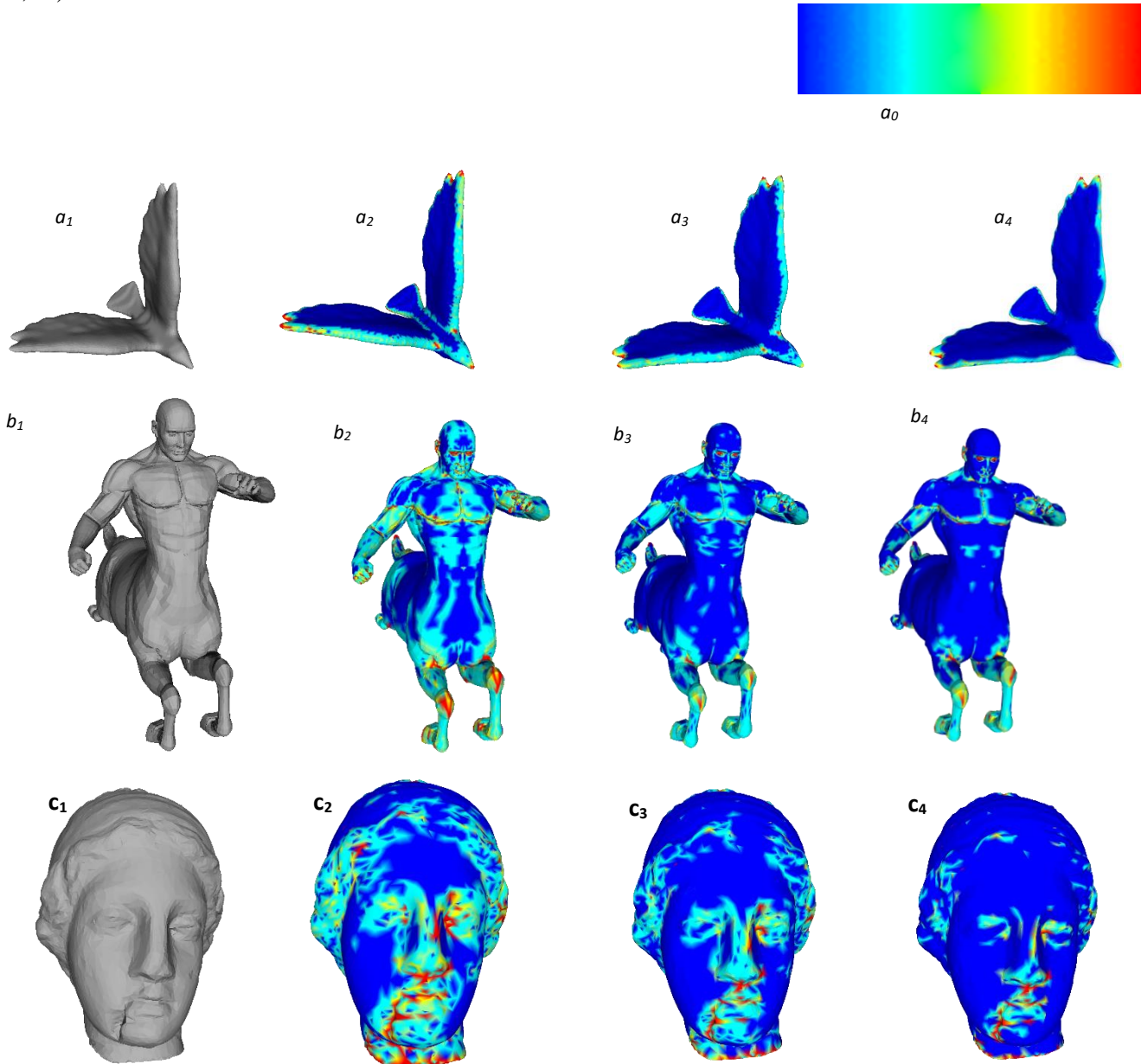


FIGURE 4.9 - Deviation factor threshold Influence

FIGURE 4.10 presents the result of our approach by construction a local graph ($k=1$) and a non-local graph with two values of k (4 and 8). These results show that the number of salient regions decrease when we apply the non-local saliency. The images (a1, b1 and c1) shows the original 3D meshes. Images (a2 and b2) show the results obtained by using a local saliency ($k=1$). In contrast, images (a3 and b3) show the results obtained by using non-local saliency ($k=4$) and images (a4 and b4) with ($k=8$). We notice that the number of salient regions with a local graph is greater than the number of regions detected with a non-local graph. Furthermore, we can notice that the variation in parameter n has a similar effects as k due to the larger number of neighbors' information. Image (c2) shows the results obtained by using ($k=2$, $n=9$), (c3) shows the results obtained by using ($k=2$, $n=12$) and (c4) shows the results obtained by using ($k=2$, $n=15$).

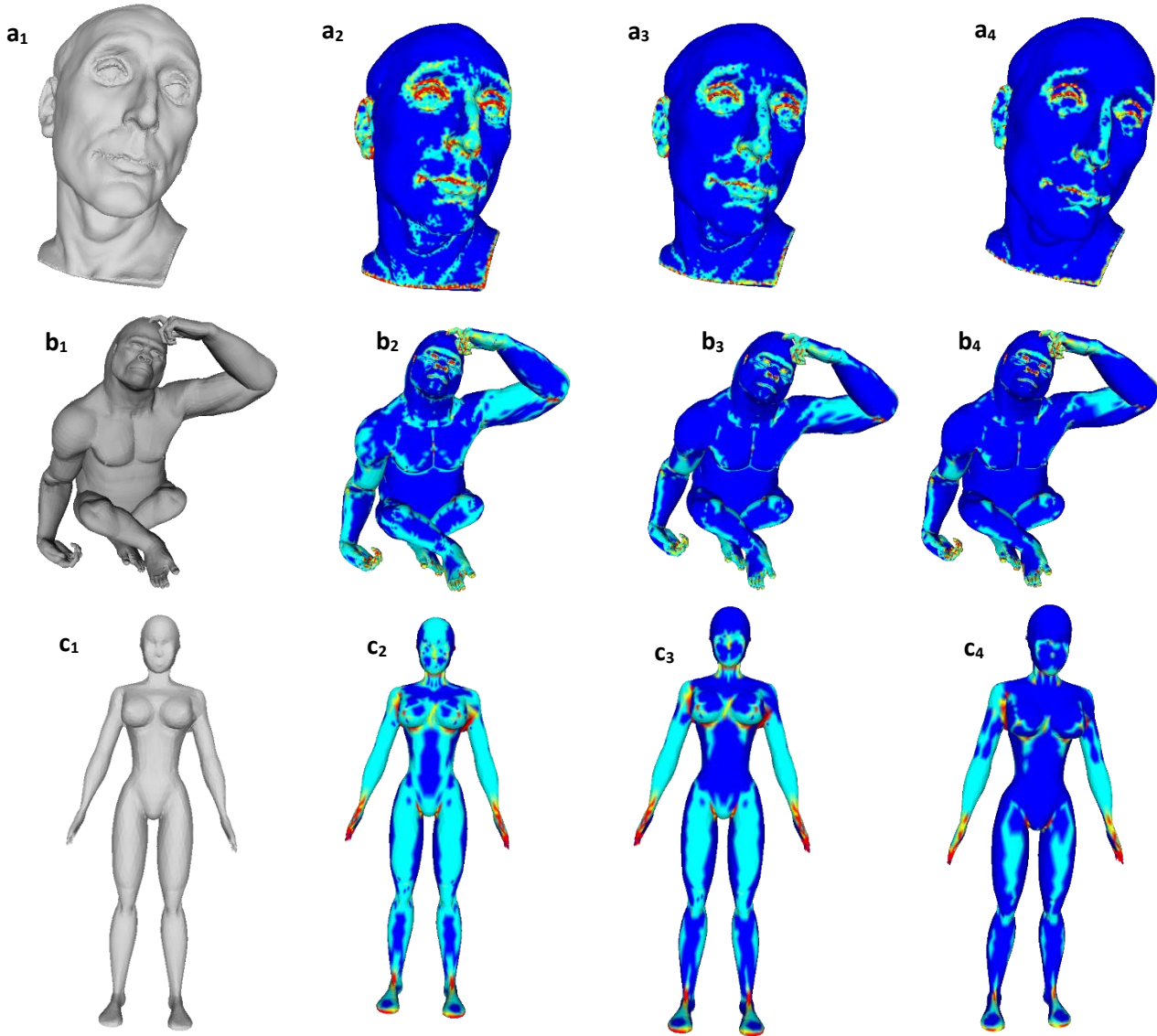


FIGURE 4.10 - Saliency map with a Local and non-local saliency computation

FIGURE 4.11 presents the results by applying our proposed method on 3D point clouds with and without noise. We construct a Delaunay graph using the concept of Delaunay triangulation from the given point clouds. Images (**a₁** and **a₃**) show the original 3D point clouds, images (**a₂** and **a₄**) present the detected salient regions with $k=1$, $n=9$ and $x=9$.

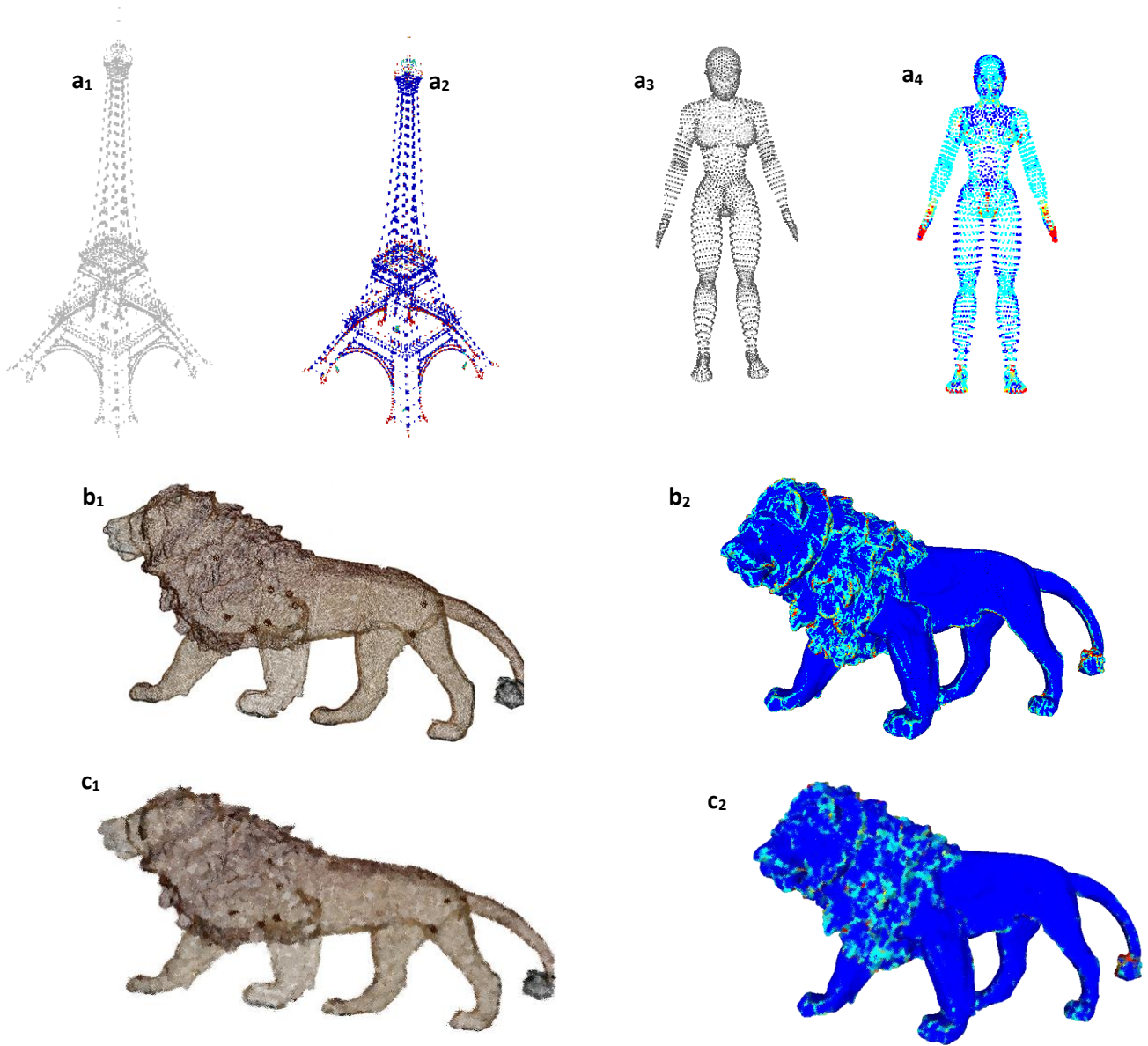


FIGURE 4.11 - Results of our method on 3D point clouds

The image **b₁** shows the lion 3D point cloud, image **c₁** shows the noisy lion 3D point cloud. Results obtained in (**b₂** and **c₂**) demonstrate that our method detects the same salient regions and deals with noise.

4.4. Comparison with the state-of-the-art methods

In this section, we compare our method (first approach) with some previous related works. We show in (FIGURE 4.12) a failure case obtained by the method proposed in [20] and the result of our proposed method, which is a failure case also, but better than the result obtained by [20].

It shows also a comparison of our method with some other methods. We can conclude visually that the best results are obtained by the method proposed in [20] and our method. Note that in (**b₃**) most of flat surfaces are considered as salient regions especially in legs.

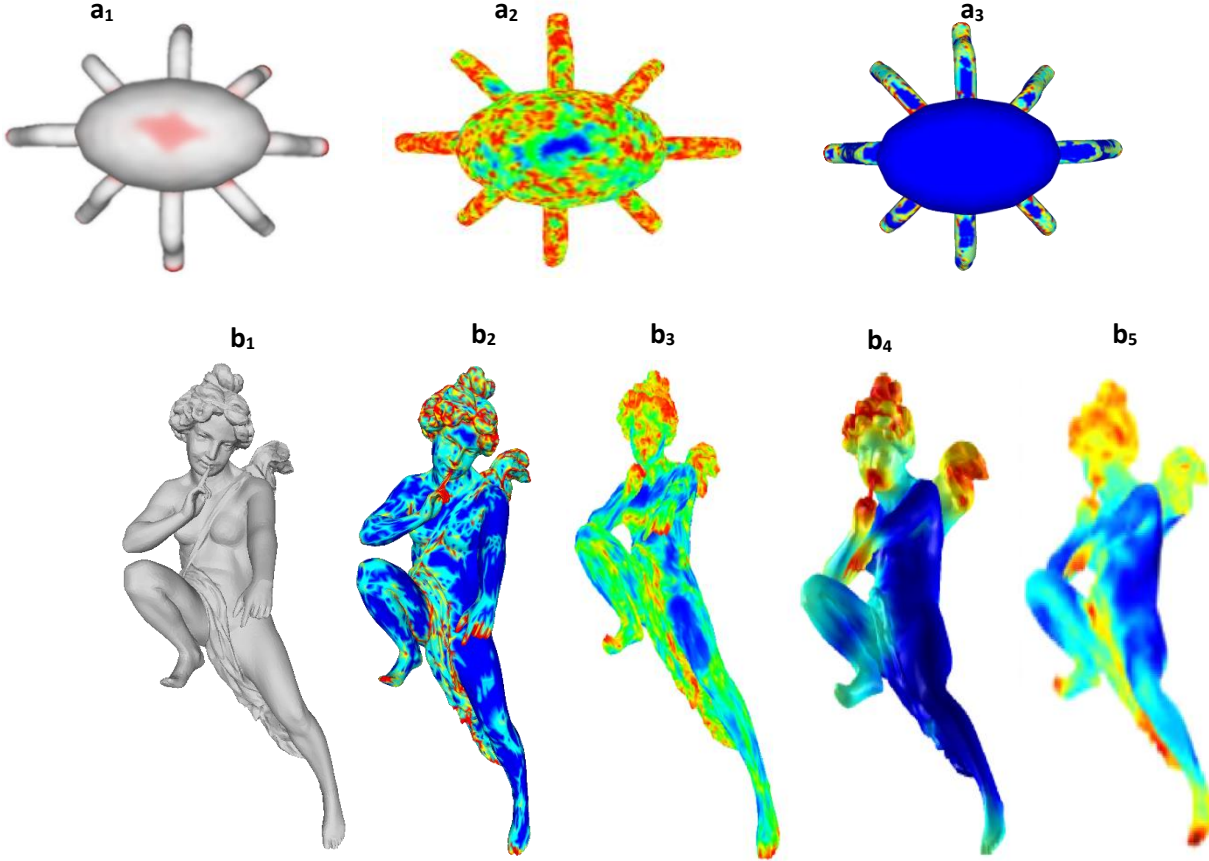


FIGURE 4.12 - Image (**a₁**) shows the ground truth spider 3D mesh, image (**a₂**) shows the saliency detected by the method proposed in [20] (**Multi-scale approach with adaptive patch size**). Image (**a₃**) shows saliency detected using our method ($k=1$ and $x=9$). Image (**b₁**) shows the original 3D mesh (14,227 vertices), image (**b₂**) shows saliency detected with our method ($k=1$ and $x=8$), image (**b₃**) shows the saliency detected in [20], image (**b₄**) shows the saliency in [65] (**detecting regions of interest of surfaces using Fast affine template matching**) and image (**b₅**) shows the saliency detected in [66] (**Saliency detection based on conditional Random Field (CRF) framework**)

FIGURE 4.13 shows a comparison of the detected saliency on the 3D mesh Dinosaur with some related works. As shown above, our method considers out-standing vertices in a flat surface as salient points and also those found on salient regions (fluctuations on surfaces naturally attract the attention of the human observation). Our result shown in (**a₂**) is similar to the result of the method proposed in [20] (**a₅**) but our method detects in addition all sharp regions in the two legs of the Dinosaur mesh. The result presented in [23] (**a₃**) considers the ribs of the 3D Dinosaur model as non-salient regions although the existence of fluctuations and sharpness in this region. In addition, the approach proposed in [23] (**a₃**) detects the flat

surface in the head as salient regions. In contrast, the method proposed in [66] (**a₄**) considers the most of object as salient regions even if those areas are flat.

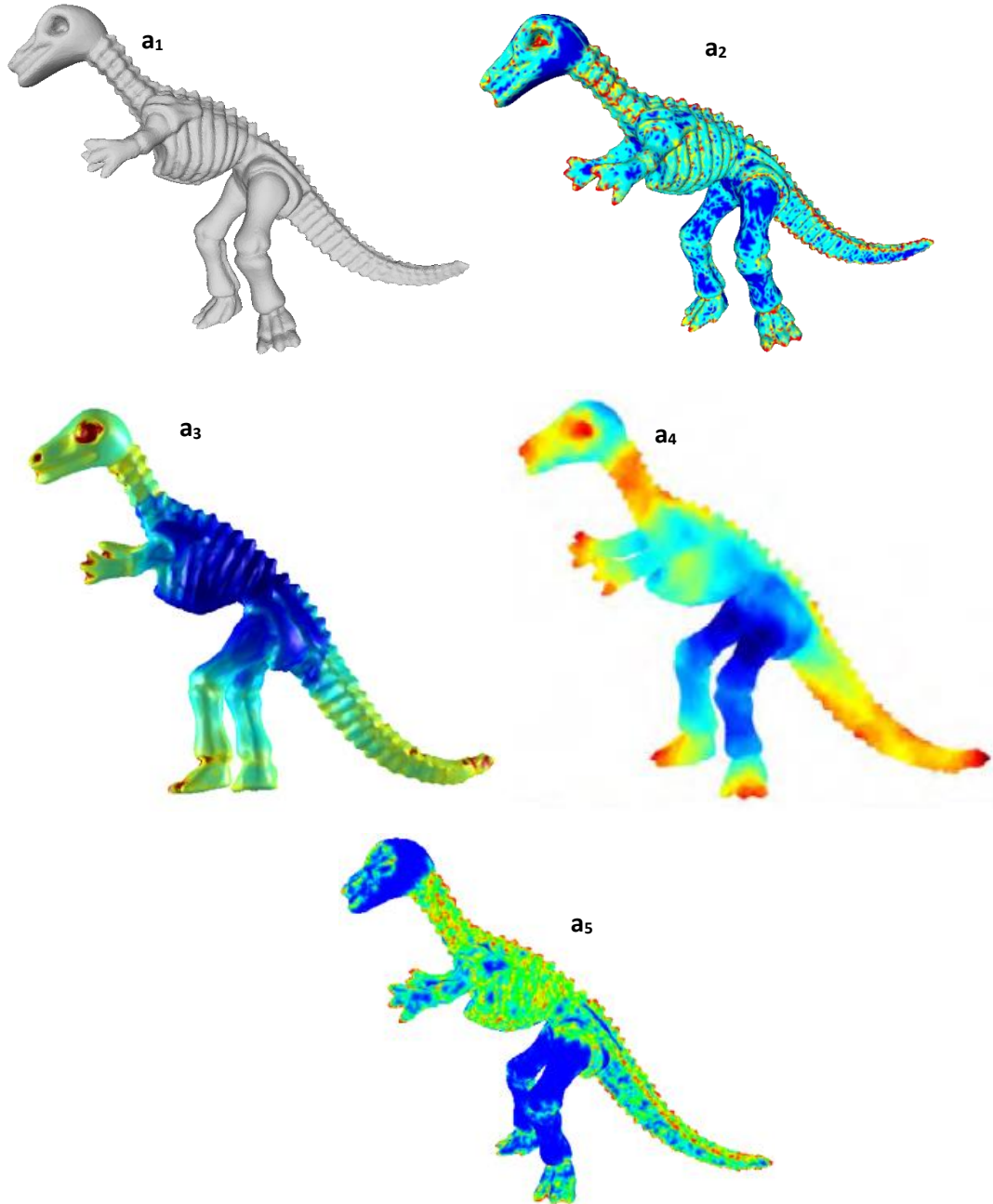


FIGURE 4.13 - Comparison with the state-of-the-art: image (**a₁**) shows original 3D mesh dinosaur (21,777vertices), image (**a₂**) shows saliency detected with our method (Level=1 and DF=9), image (**a₃**) shows the saliency detected in [65], image (**a₄**) shows the saliency detected in [66] and image (**a₅**) shows the saliency detected in [20]

FIGURES 4.14 and 4.15 present another comparison with [18, 20] according to the saliency map. We can observe that our method ($k=4$, $n=12$ and $x=9$) are more consistent with ground-truth maps.

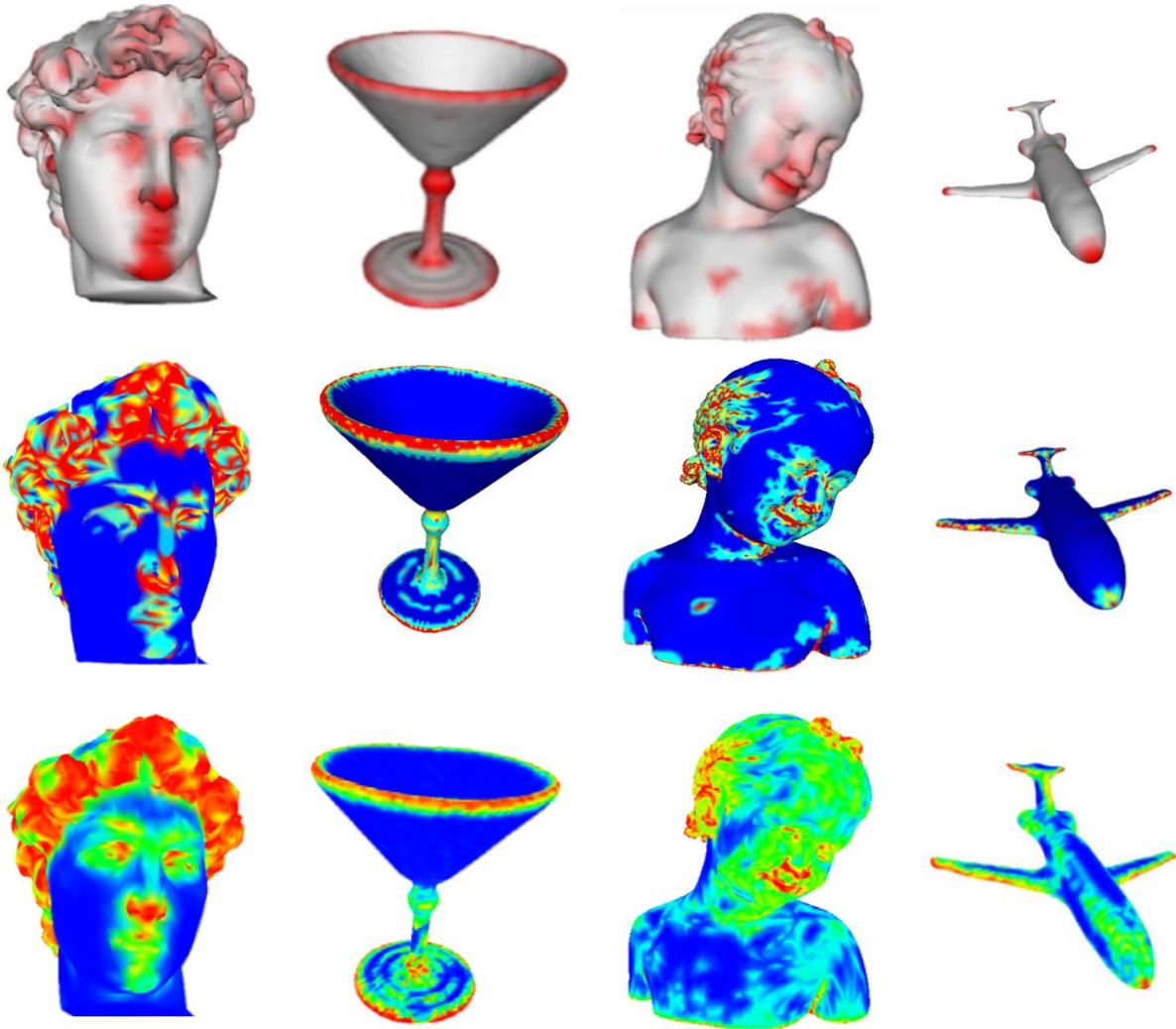


FIGURE 4.14 - Comparisons of our saliency map (second row) with Multi-scale mesh saliency [20] (third row) and ground truth [64] (first row)

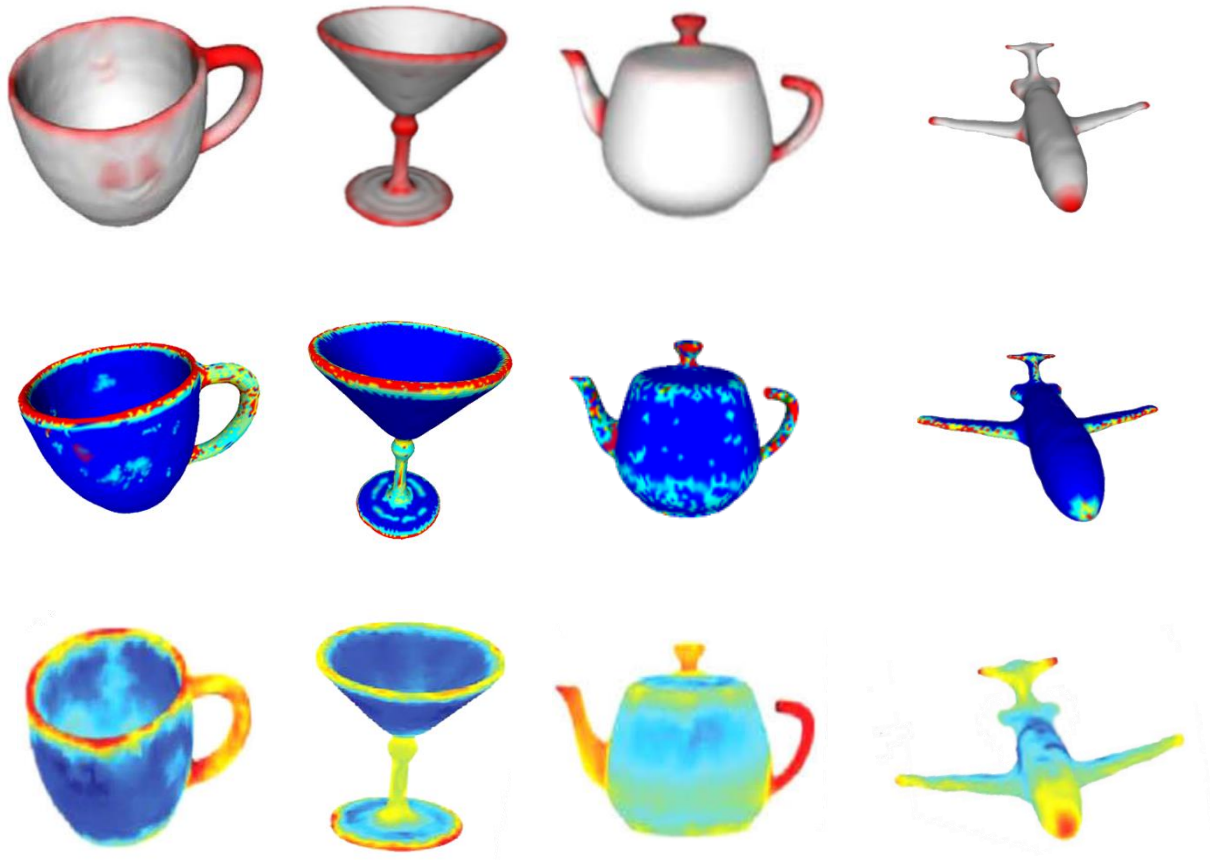


FIGURE 4.15 - Comparisons of our results (second row) with spectral mesh saliency [18] (the third row) and ground truth [64] (first row)

4.5. Comparison between the proposed approaches

By comparing the results produced by the two approaches with the ground truth maps, we can notice that the two approaches are similar and competent. However, the second approach presents better results than the first one when dealing with 3D matching application, which is discussed in the next chapter.

4.6. Conclusion

In this chapter, we have presented two novel approaches to detect salient regions on 3D meshes. In the first approach, a local patch has been created for each vertex where its cells are filled with deviation factors. The patch is then used as a local descriptor for the 3D meshes surface vertices. Furthermore, a similarity measure between patches descriptor and a height map have been calculated and in order to compute the similarity degree of vertices saliency measure. In the second approach, a novel 3D invariant surface descriptor has been created for each vertex based on Zernike coefficients. Furthermore, a single-scale-saliency degree is calculated, then, a multi-scale saliency map is computed in order to improve the

quality of the measured saliency and cope with noise. Finally, we showed the robustness and the efficiency of our approach by showing that our approach detects most of visual salient regions in a 3D mesh surface. Finally, we showed that our approach could detect salient regions on any data that can be represented by a weighted graph such as point cloud.

In the next chapter, we will present multiple applications based on our saliency methods in order to demonstrate the efficiency of our approaches.

Chapter 5

APPLICATIONS

Summary

5. Applications	79
5.1. 3D objects segmentation	80
5.1.1. Multi Search Hill Climbing – Algorithm	80
5.1.2. Heuristic algorithm	81
5.1.3. Experimental results	82
5.2. Feature points	83
5.2.1. Feature regions detection	83
5.2.2. Feature points detection	84
5.3. 3D point cloud simplification	84
5.3.1. Introduction	86
5.3.2. Contributions	87
5.3.3. Simplification process	87
5.3.4. Experimental results	88
5.3.5. Evaluation	91
5.4. 3D models matching	92
5.4.1. Template matching approach	92
5.4.2. Linear approach	93
5.4.2.1. 3D Object matching	94
5.4.2.2. Experimental results	96
5.4.2.3. Search experiments	98
5.4.2.4. Quantitative evaluation	99
5.4.2.5. Others applications	99
5.4.2.6. Comparison between the proposed approaches	100
5.5. Conclusion	100

Many 3D processing applications can benefit from salient regions detection on 3D objects or point clouds. In this chapter, we present multiple applications based on salient regions detection. First, a 3D objects segmentation is proposed, followed by feature points detection. Starting from these two applications, we present also two others, which are the 3D point clouds simplification, and the 3D objects matching.

5.1. 3D objects segmentation

The segmentation of 3D objects based on salient regions detection has been widely used in many applications of computer vision field such as shape matching, objects alignments, 3D point clouds simplifications, etc.

We propose to extend our methods proposed in the previous chapter to mesh region segmentation. To segment salient regions, we propose two algorithms. First algorithm is a custom version of Hill Climbing algorithm [67]. This algorithm uses the dilation morphological operator as defined in (EQUATION 5.4) to propagate throw weighted graph vertices and define regions (vertices), which are similar in term of saliency features. The second is a heuristic algorithm that solve a linear programming model for a graph-clustering problem. This problem derived from graph coloring problem.

5.1.1. Multi Search Hill Climbing – Algorithm

The main idea of this algorithm is to move to a list of vertices that maximizes better the solution $H_m(v_i)$ at each step rather than moving to one vertex. $H_m(v_i)$ is a function that measures the homogeneity between v_i and its neighbors. This algorithm is faster than the regular Hill Climbing algorithm since multi selection are performed at each level. Furthermore, it avoids the local maxima solution because more than one solution can be found in the same iteration.

Let $F(v_i, v_j): V \times V \rightarrow \{0,1\}$ be a function defined as:

$$F(v_i, v_j) = \begin{cases} 1 & \text{if } |SL(v_i) - SL(v_j)| \leq \sigma \\ 0 & \text{otherwise} \end{cases}, \quad (5.1)$$

where $SL(v_i)$ is the saliency degree of vertex v_i and σ is a parameter defining the point clouds density.

The homogeneity function $H_m(v_i): V \rightarrow \mathbb{N}$ can be defined as follows:

$$H_m(v_i) = \sum_{v_j \sim v_i} F(v_i, v_j), \quad (5.2)$$

Let HG a structuring element defined as following:

$$HG_v = \{v_j \in D_n(v_i) / F(v_i, v_j) = 1\}, \quad (5.3)$$

A dilation from a vertex v_i to v_j on a graph G using the structuring element HG is given by the following set of operations:

$$G \oplus HG = \{(v_i + v_j) | v_i \in G, v_j \in HG\}, \quad (5.4)$$

The multi search Hill Climbing algorithm steps can be summarized as follows:

- Pick a random vertex v_i from the weighted graph.
- Choose the list of neighbors v_j from $D_n(v_i)$ that maximizes better the function $H_m(v_i)$.
- Apply the dilation operator on v_i .
- Repeat step 2 and 3 for each element v_j in $D_n(v_i)$ until it becomes empty.

The following figure illustrates this algorithm.

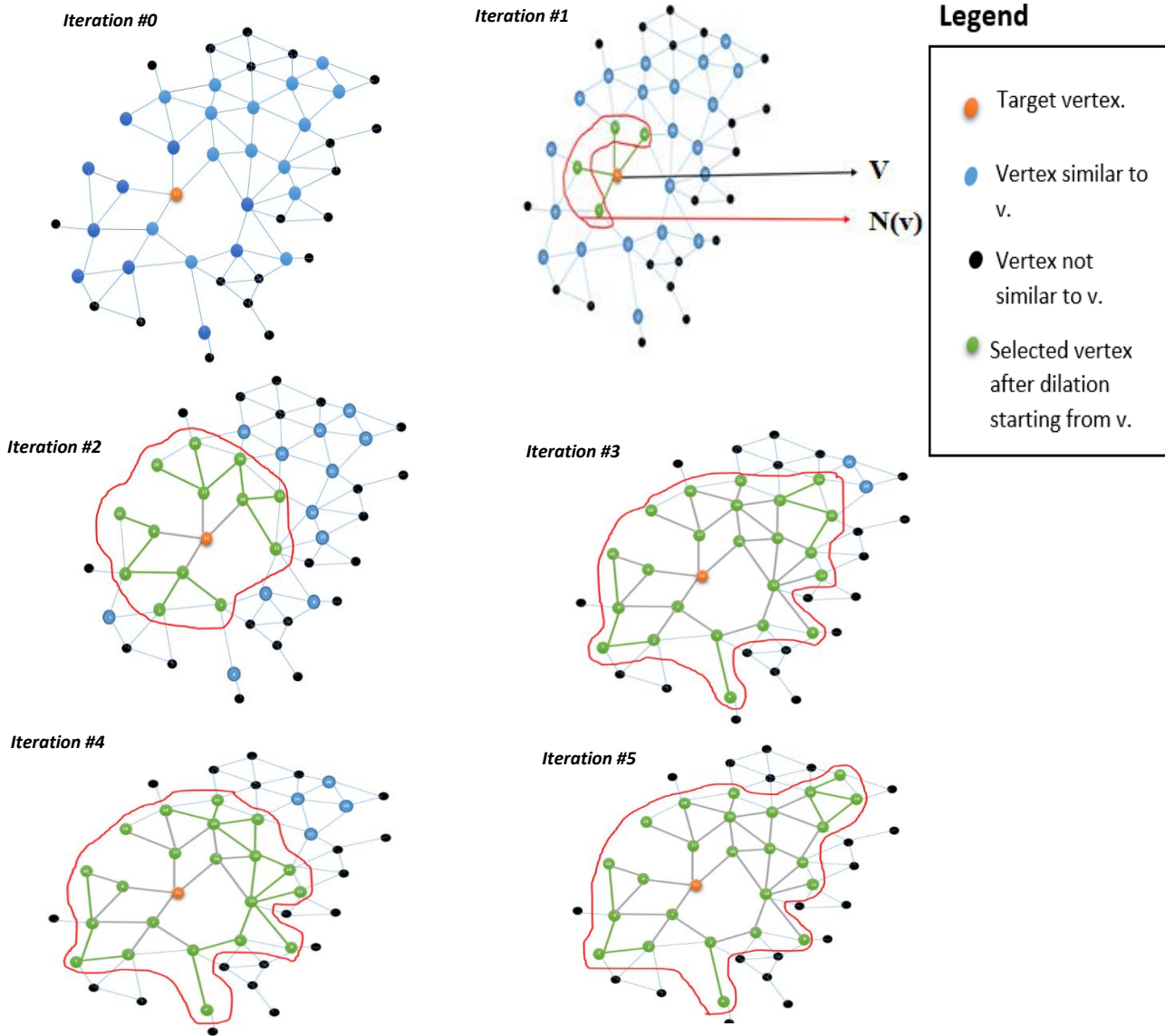


FIGURE 5.1 - Multi Search Hill Climbing – Algorithm

5.1.2. Heuristic algorithm

In order to classify the graph vertices into groups of vertices similar in term of saliency degree, we adopt a linear programming model that groups the set of vertices belonging to the same class by minimizing a cost function. Let us start with the combinatorial problem of labelling the graph vertices where we search to assign the same label to adjacent vertices belonging to the same class using the minimal number of labels.

We propose to formulate the *vertex-labelling* process as an integer linear problem **ILP**. To do so, we consider a set of vertices $V = \{v_1..v_n\}$ and a set of n labels where the initial state of the problem considers that each vertex has its specific label.

We introduce two binary variables y_k and x_{ik} , $k = 1..n$. The first variable indicates whether a label k is used ($y_k = 1$) or not ($y_k = 0$), and the second one indicates if a given vertex v_i has received the label k . The model with all variables is established as following:

$$\text{Min } \sum_{k=1}^n y_k$$

Subject to:

$$(1) \sum_{k=1}^n x_{ik} = 1 \quad \forall i=1, \dots, n$$

$$(2) x_{ik} - y_k \leq 0 \quad \forall i, k=1, \dots, n$$

$$(3) x_{ik} - x_{jk} \leq 0 \quad \forall k = 1, \dots, n; \forall (v_i, v_j) \in E; F(v_i, v_j) = 1,$$

$$(4) x_{ik} \in \{0, 1\}; y_k \in \{0, 1\}$$

The constraints (4) and (5) ensure that x_{ik} and y_k are binary variables. The constraints (1)-(3) guarantee (in this order) that each vertex is labelled; vertex v_i receives the label k only if this label is used, and any two adjacent vertices have different label if they belong to different classes. The optimal solution of such problem is a graph where each group of connected vertices belonging to the same class receives the same label.

We present a heuristic algorithm to solve the ILP problem. This algorithm is similar to that presented in (CHAPTER 3: *ALGORITHM 3.1*) where the score function $Score(v_i, v_j)$ is defined as:

$$Score(v_i, v_j) = \begin{cases} 0 & \text{if } v_j \in C(v_i) \\ 1 & \text{otherwise,} \end{cases} \quad (5.5)$$

where $C(v_i)$ is the set of vertices belonging to $D_n(v_i)$ and similar to v_i in term of saliency.

$$C(v_i) = \{v_j \in D_n(v_i) \text{ such as } F(v_i, v_j) = 1\} \quad (5.6)$$

5.1.3. Experimental results

In (FIGURE 5.2), we present the result using the proposed segmentation method based on the vertices saliency degree. Each group of similar connected vertices has a specific color and form a separated segment. From left to right, the first column is the original 3D object, the second column is the saliency map and the last column is the segmented object.

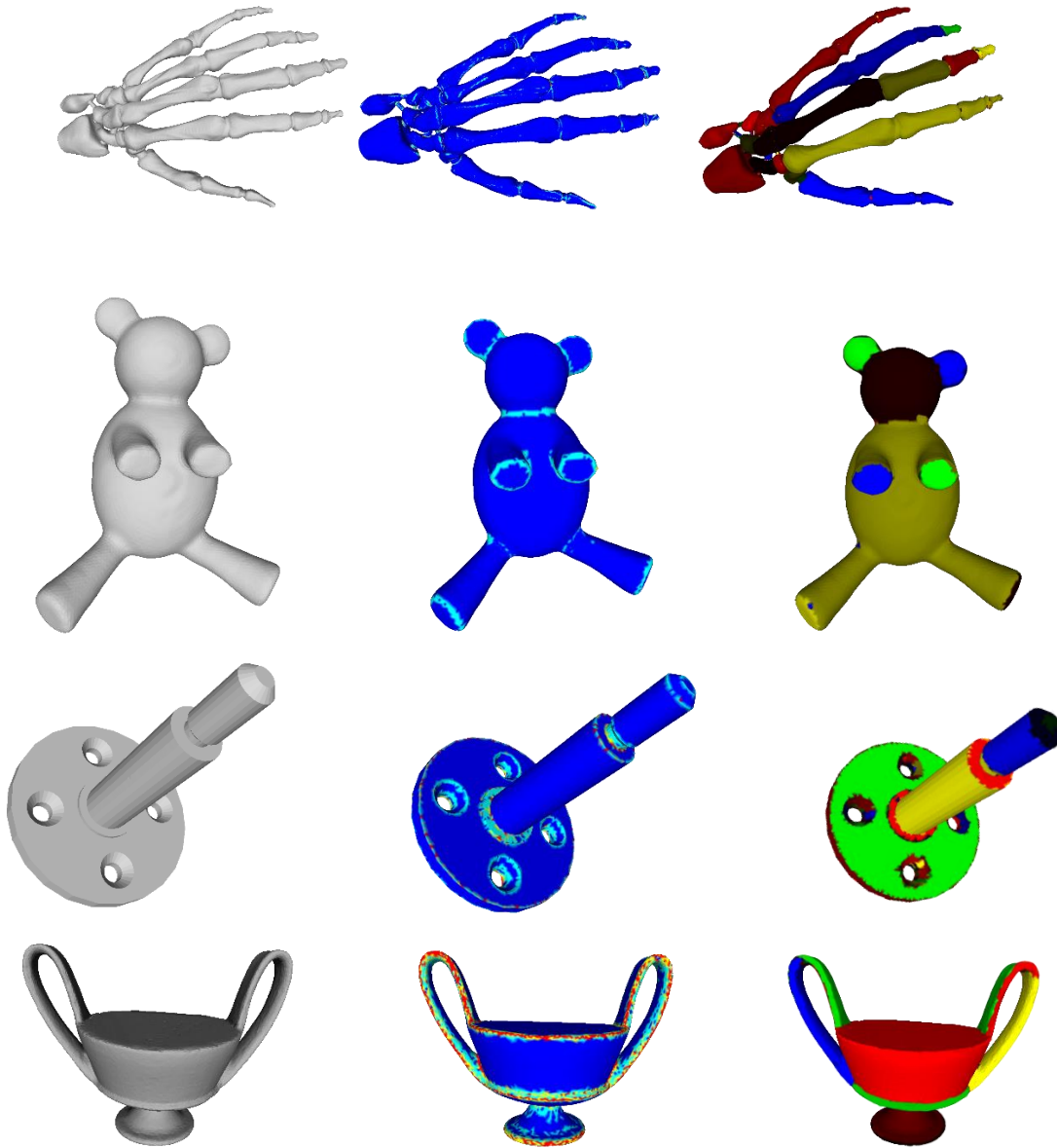


FIGURE 5.2 - 3D mesh segmentation based on saliency degree

5.2. Feature points

When we segment the 3D surface according to saliency degree of vertices, the tiny segments detected will form the features regions that describe the geometry of the surface and used later to identify the feature points.

5.2.1. Feature regions detection

In this subsection, we present a method to detect feature regions followed by detecting feature points on a 3D point cloud using the saliency degree of vertices. Firstly, we segment the 3D point cloud according to saliency degree of vertices. Then, we define the set of interest points (Feature points) I_p in each segmented region. The segmentation process is accomplished using one of the two algorithms proposed in the previous section.

FIGURE 5.3 summarizes the regions segmentation process:

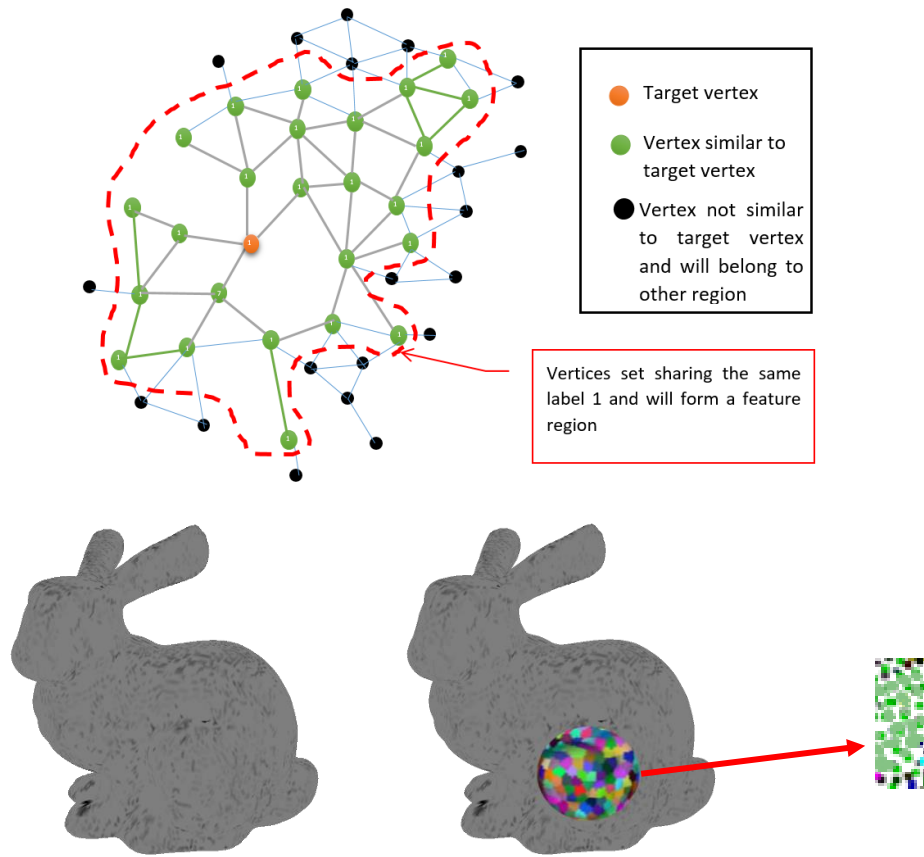


FIGURE 5.3 - Feature regions

5.2.2. Feature points detection

In this subsection, we describe the process of defining the set of feature points in each feature region. Note that an interest point can be located inside a feature region or on its boundaries as shown in (FIGURE 5.8).

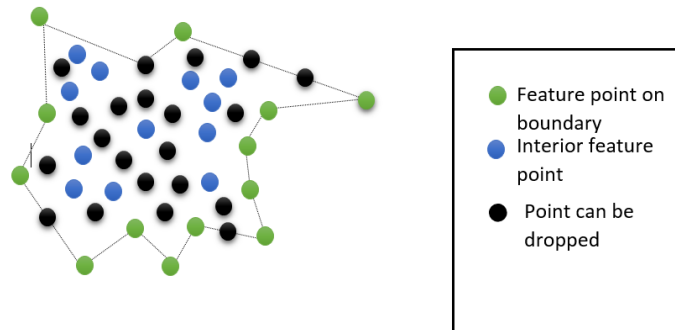


FIGURE 5.4 - Illustration of distribution of interest points

The boundary point set of a feature region is defined as its Concave Hull presented in (FIGURE 5.5).

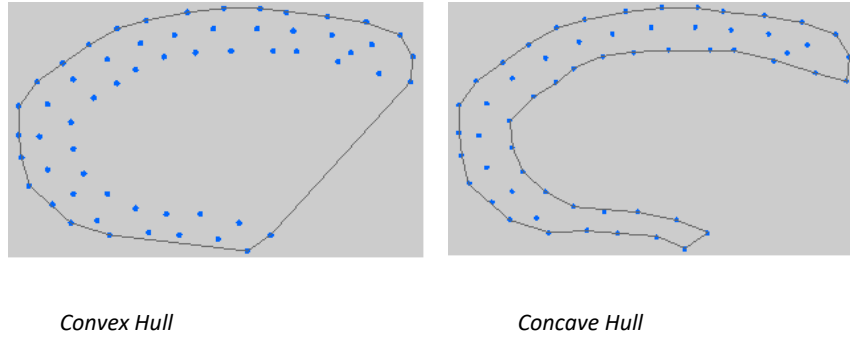


FIGURE 5.5 - Illustration of feature region boundary

The set of interest points are defined amongst the set of boundary points as the extremities of collinear point's sets. The following figure illustrates the detection of boundary feature points.

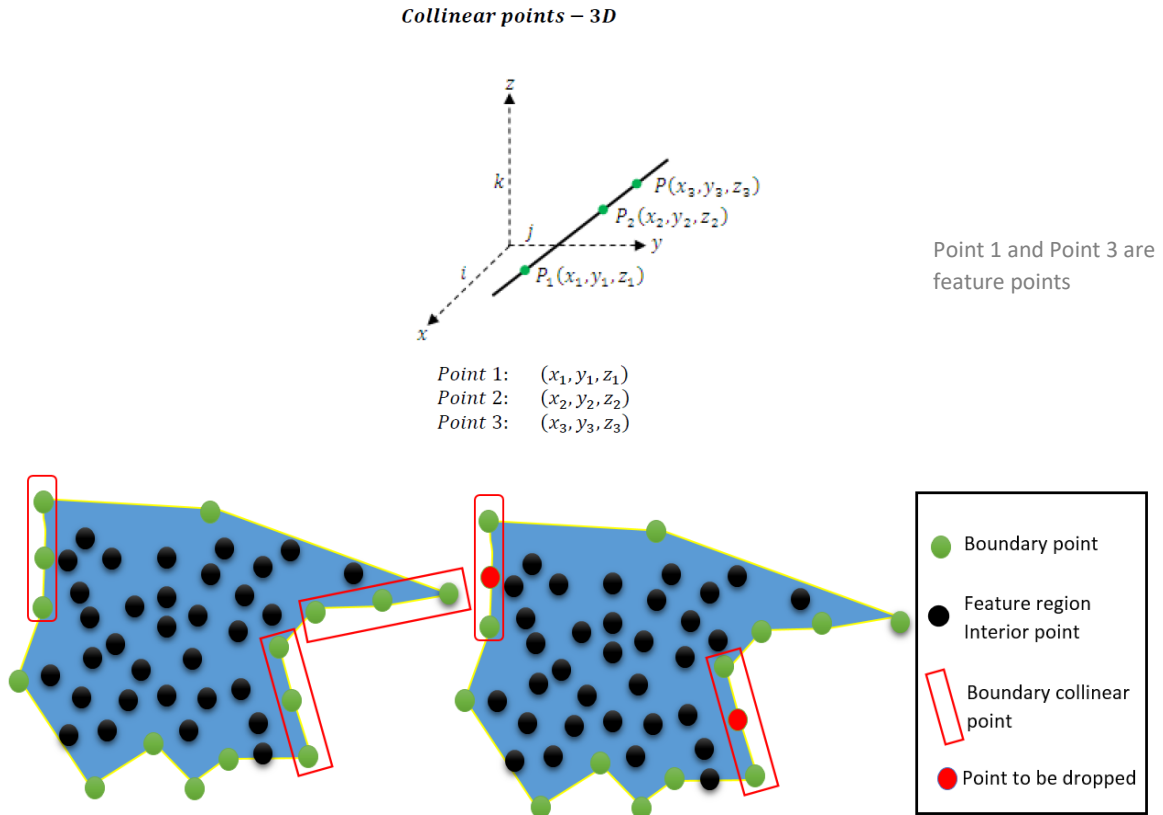


FIGURE 5.6 - Boundary feature points

The set of collinear points is defined in the form of triplet points. Therefore $P1(x_1, y_1, z_1)$, $P2(x_2, y_2, z_2)$, and $P3(x_3, y_3, z_3)$ are collinear if, the area A of the triangle composed by $P1$, $P2$ and $P3$ equals to zero. Let a , b and c be the sides of this triangle and defined as:

$$\begin{aligned}
a &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \\
b &= \sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2 + (z_3 - z_1)^2} \\
c &= \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2 + (z_3 - z_2)^2}
\end{aligned} \tag{5.7}$$

The area of the triangle s is calculated using Heron's formula:

$$A = \sqrt{s(s-a)(s-b)(s-c)}, \tag{5.8}$$

where s represents the half of triangle perimeter.

The set of interior points in each feature region are weighted by their saliency degree. Thus, we can classify these points as High salient or Low salient points. This classification is made according to a local mean saliency measure (LMS) and a global mean saliency measure (GMS) as following:

$$LMS(r) = \frac{\sum_{v_i \in r} SL(v_i)}{|r|}, \tag{5.9}$$

$$GMS = \frac{\sum_{v_i \in G} SL(v_i)}{|G|}, \tag{5.10}$$

where r represents the set of points in each feature region, and $|G|$ the number of graph vertices.

To check whether a point is an interest point, we define the function $ITP(v_i, r)$ as:

$$ITP(v_i, r) = \begin{cases} 1 & \text{if } (GMS - \alpha) \leq SL(v_i) \text{ or } (LMS(r) - \alpha) \leq SL(v_i) \\ 0 & \text{otherwise} \end{cases}, \tag{5.11}$$

where α is a threshold parameter.

Finally, the interest points located into feature regions and on boundaries can be considered as feature points.

5.3. 3D point cloud simplification

5.3.1. Introduction

3D content has an important role in many domains such as computer vision, architectural and industrial design, scientific visualization, animation films, and medical imaging. However, the performance of 3D scanning devices [68] has been enhanced year by year and modern scanners generate complicated and dense point clouds. This leads to a large data redundancy, which must be removed in order to limit the computing resources needed to analyze and represent the form. Thus, the objective of the simplification process is to eliminate duplicated and redundant points that does not affect the characteristics of the initial form [69]. Consequently, feature points that represent the geometry such as sharp features and boundaries are maintained.

The methods of point clouds simplification can be classified into three categories: clustering methods, methods coarse-to-fine, and iterative methods.

The first category (clustering methods) covers the algorithms which aim at subdividing the input point clouds into several surface patches (group of points), based on a certain criterion, then to replace every patch by its representative point (the centroid for example). In this context, Pauly et al. [29] proposed two different strategies, inspired by the methods of mesh simplification [70, 71], for the construction of patches. The first is incremental and uses an algorithm of regions growth. The second is hierarchical, and bases itself on a binary partition of the space. The number, as well as the size of patches, are controlled by means of the curvature. The authors mention that the methods of clustering are fast and effective regarding resource memory, but leads to point clouds presenting an important quadratic error.

The approaches of type coarse-to-fine extract randomly a set of points from the original points cloud then call on to a 3D Voronoï diagram in order to define implicitly a distance function. This last function is then used to refine the cloud until the error tolerated by the user is reached. Moenning and Dodgson [31] proposed an algorithm based on this principle.

The iterative methods correspond to the algorithms where the objective is to reduce in an iterative way the number of points of the original point clouds by using a decimation operator as the one proposed in [72].

In this section, we propose a new simplification method based on the saliency measure of point cloud vertices, due to the high saliency having sharp points and boundaries. Our method detect feature regions that represents the characteristic of the surface then removes the points having a low saliency degree from those regions.

5.3.2. Contributions

In this method, the point cloud is represented by a weighted graph, and the saliency degree of vertices is calculated. Then, the graph is segmented into feature regions that maintain the characteristics of the surface shape where each region is a set of similar vertices in term of saliency degree. Secondly, we define the set of feature points using the algorithm proposed in the previous section. Finally, we maintain the feature points and remove non-featured points. Our method is illustrated in (FIGURE 5.7).

5.3.3. Simplification process

The simplification process consists to define the set of interest points I_p in each segmented region that will represent finally the set of points to conserve in the simplified data. Finally, the resulting simplified 3D point cloud is composed by all feature points classified as high saliency point.

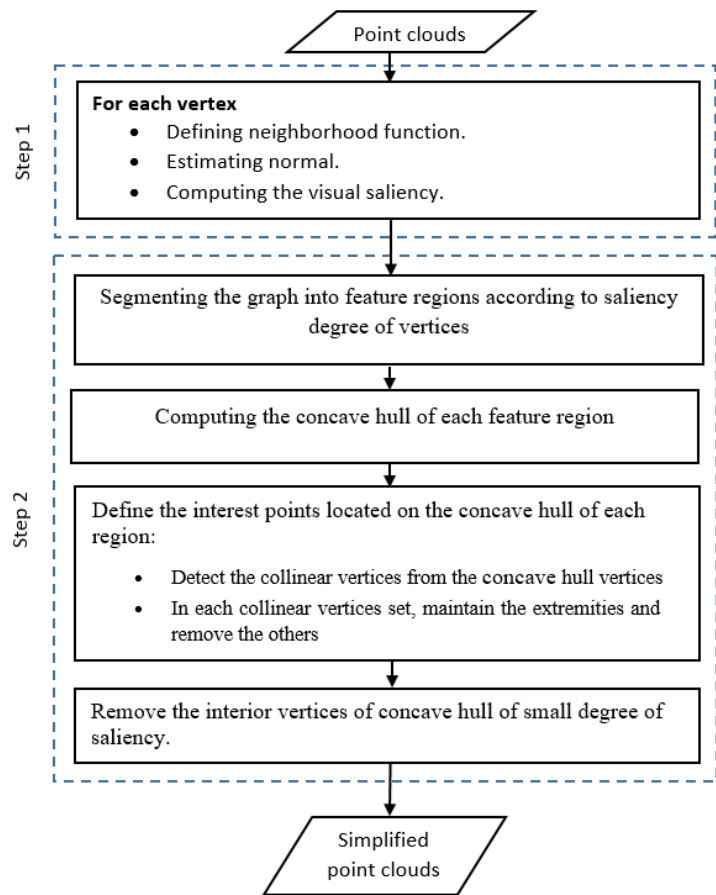


FIGURE 5.7 - Flowchart of our method

In some cases where the set of interior points is classified as low saliency points, the fact that produce gaps in the simplified 3D point cloud. To handle this case, we propose to simplify this set according to the following rules and illustrated in (FIGURE 5.8).

- Compute the gravity center GCS of region high salient points,
- Compute the midpoint of segment connecting each boundary interest point with the gravity center GCS ,
- Maintain the nearest low salient point of each midpoint.

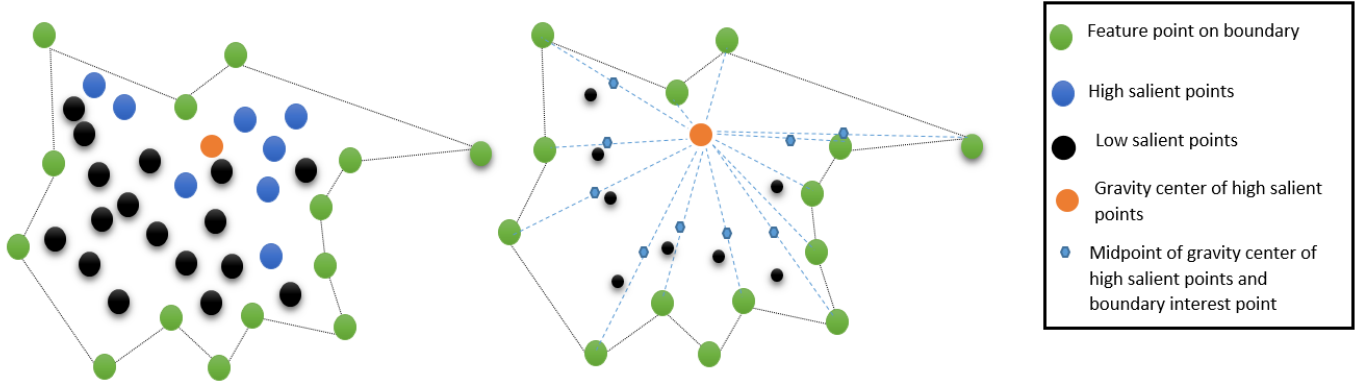
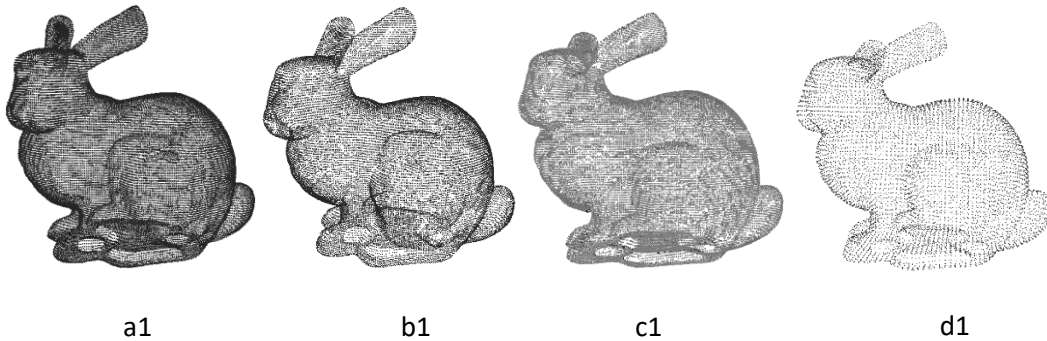


FIGURE 5.8 - Region gaps treatment

5.3.4. Experimental results

In this section, we show the efficiency and robustness of our proposed method on many colored 3D point clouds models and 3D meshes. In these experimental results, we present and discuss the influence of different parameters on our approach such as graph Delaunay-level k , the variable x in (EQUATION 4.10) that defines the deviation factor threshold in patches entropy calculation, threshold parameter α in (EQUATION 5.1) and the number of patch cells n in (SECTION 4.2.2.2).

In (FIGURE 5.9), we present the results of our simplification method applied on a point clouds. Images (a₁, a₂, a₃, a₄ and a₅) show the original 3D point clouds, images (b₁, b₂, b₃, b₄ and b₅) show the simplified point clouds with the following parameters values ($n=8$, $x=9$ and $\alpha=0$). Images (c₁, c₂, c₃, c₄ and c₅) show the simplified point clouds with ($n=8$, $x=9$ and $\alpha=30$). Images (d₁, d₂, d₃, d₄ and d₅) show the simplified point clouds with ($n=8$, $x=9$ and $\alpha=50$).



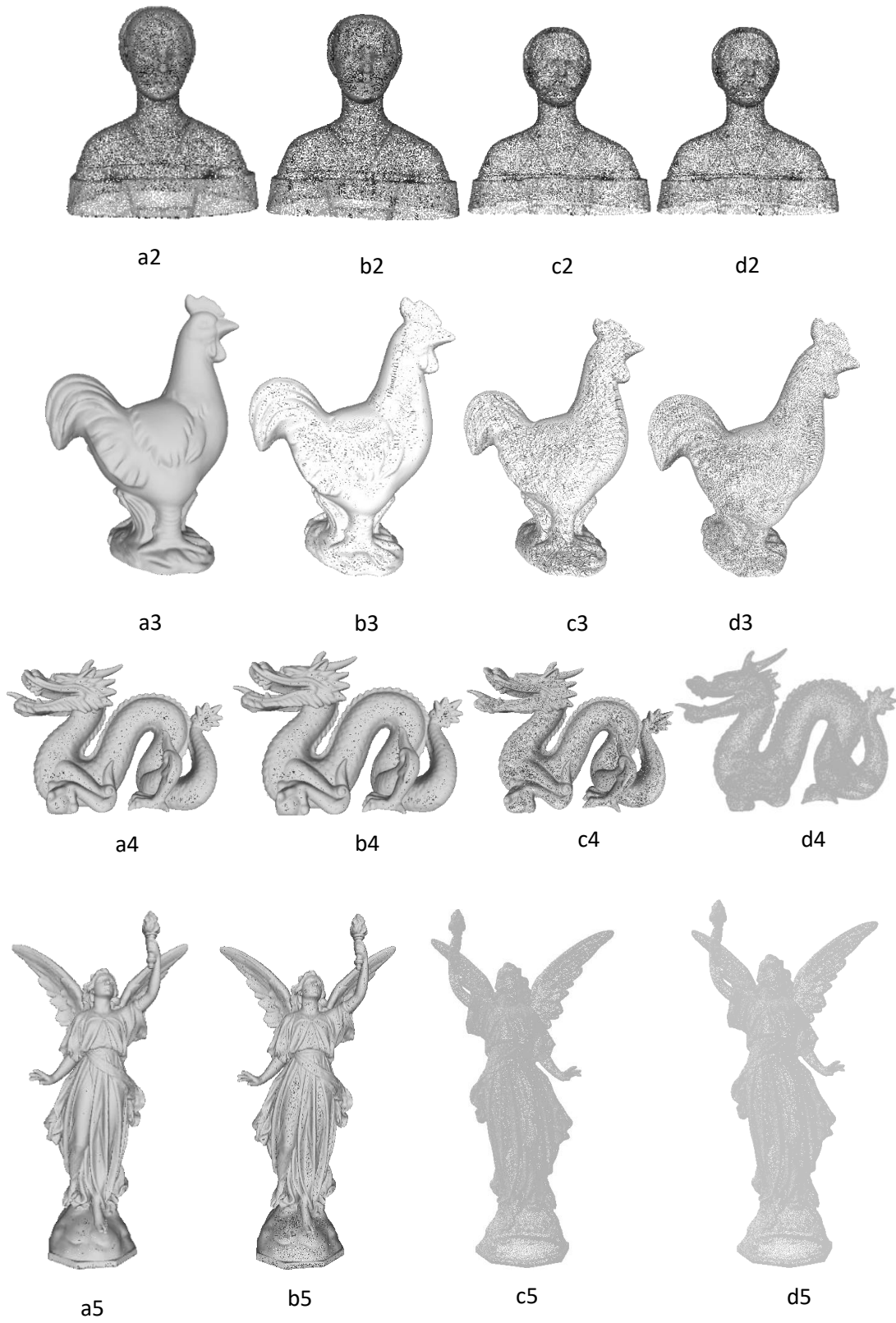


FIGURE 5.9 - Contribution of the factor α

In order to evaluate the quality of the simplified model generated by our method, we propose to measure the geometric error as the maximum error between the initial \mathbf{M} and the simplified \mathbf{M}' model, i.e. the hausdorff distance.

$$\Delta_{max}(\mathbf{M}, \mathbf{M}') = \max_{q \in \mathbf{M}} d(q, \mathbf{M}') \quad (5.12)$$

and the geometric average error:

$$\Delta_{avg}(\mathbf{M}, \mathbf{M}') = \frac{1}{|\mathbf{M}|} \sum_{q \in \mathbf{M}} d(q, \mathbf{M}') \quad (5.13)$$

TABLE 5.1 – Geometric error measurement

Model	Number of original points	Number of simplified 3D points			Geometric error	
		$\alpha = 0$	$\alpha = 30$	$\alpha = 50$	Δ_{max}	Δ_{avg}
Bunny	35947	30134	27316	16476	0.005151	0.000265
Laurana	27861	18819	17492	16241	0.003313	0.000014
chicken_high	135142	120415	118812	100616	0.050000	0.012888
Dragon	437645	387412	185620	33960	0.044323	0.021243
Lucy	262909	242308	212032	106007	0.064213	0.014888

FIGURE 5.10 presents the influence of the deviation factor threshold x on the result. Image a shows the original 3D point cloud. Image b, c, d, and e present the simplified point clouds with $(n=8, x=0 \text{ and } \alpha=0)$, $(n=8, x=5 \text{ and } \alpha=0)$ and $(n=8, x=9 \text{ and } \alpha=0)$. One can see that the sharp curves and boundaries of regions that define the characteristics of shape surface are preserved although a minimum value of deviation factor threshold is used (b, $x=0$).

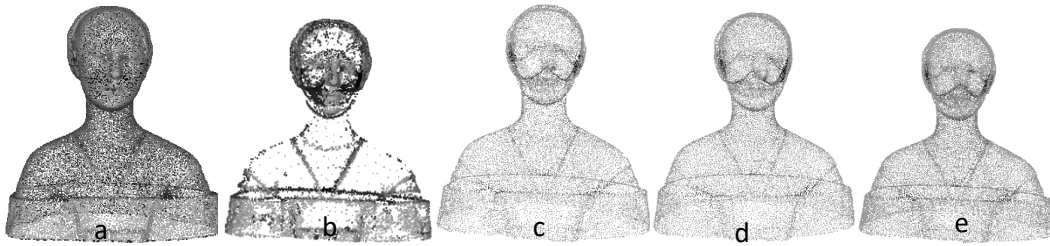
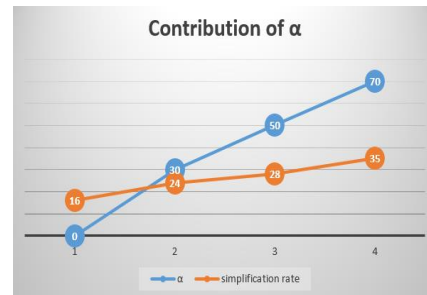
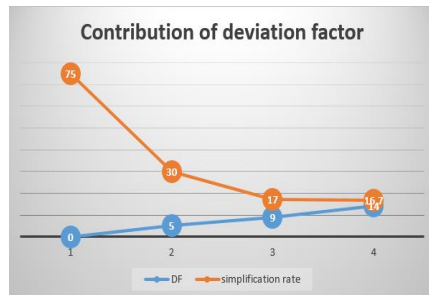


FIGURE 5.10 - Contribution of deviation factor threshold

TABLE 5.2 – Contribution of deviation factor threshold

Model	Number of original points	Number of 3D simplified points			
		$x = 0$	$x = 5$	$x = 9$	$x = 13$
Laurana	27861	9764	19636	22894	22929



5.3.5. Evaluation

In addition to the geometric average error measurement proposed in the previous section to evaluate the simplification quality, we compare our method with some related works in order to show its performance. First, note that our method has many advantages compared to some related work where the user can control the simplification rate. Furthermore, there are some cases where the simplification rate is very high [73] and some of sharp region points that conserve the characteristics of the surface shape will be removed resulting holes in the simplified 3D point clouds. In contrast, our method preserve the characteristics of the surface shape with a minimal configuration ($x=0, \alpha=0$) due to the technique we use where we maintain internal and boundary feature points. Figure 5.11 shows the simplification results proposed by [74] where the authors mention that their proposed method works only with models whose shape is symmetrical or spherical and may produces holes in others. In contrast, figure 5.9 and 5.10 shows that our method produce better results with models of different shapes (symmetric and no symmetric).

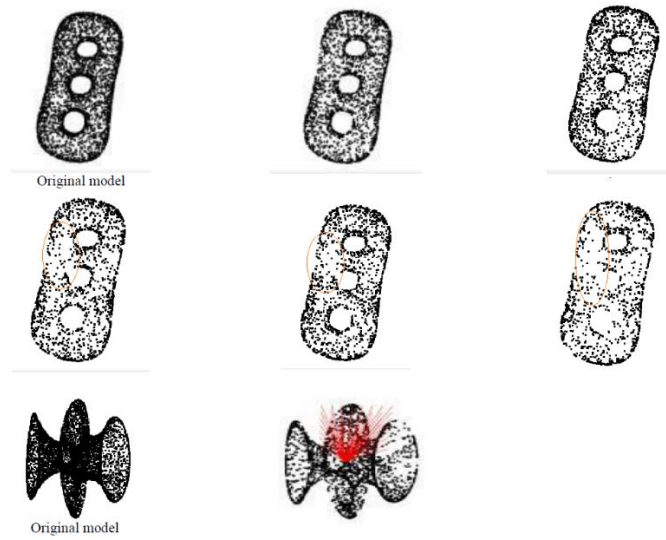


FIGURE 5.11 - Simplification results proposed by [74] show holes in the simplified point clouds

The table that follows shows a comparison of some results from [73, 74] and our method.

TABLE 5.3 – Comparison with some related works

Model	#Original points	Article	#Points of simplified model (related work)	#Points of simplified model (our method)	Screen shot in corresponding article
bunny	280 792	[35]	16 729	16476	Not available
Dragon	437,645	[74]	34,861	33960	Fig. 5.11

Finally, by comparing the geometric error on simplification results (Bunny model) between [35] and our method (Table 5.1, first row), we can notice that our method is better.

FIGURE 5.12 presents a comparison between our method and the method proposed in [74] on the dragon 3D point cloud.

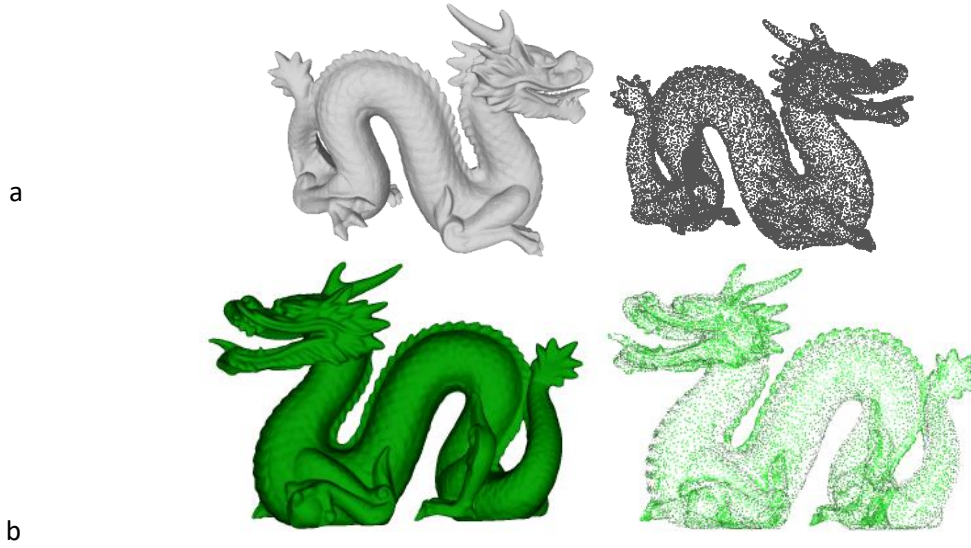


FIGURE 5.12 - Dragon model: (a) our method, (b) method in [74]

5.4. 3D models matching

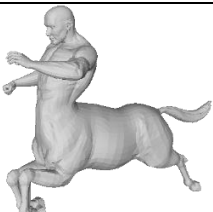
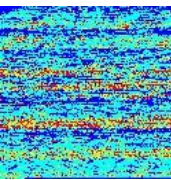

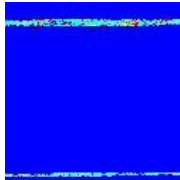
3D objects matching is a fundamental research problem in computer vision, with applications in computer graphics, medical imaging, molecular biology, and many other fields.

In this section, we describe our proposed approaches for 3D models matching using our saliency detection methods.

5.4.1. Template matching approach

The novelty of this method depends on the key points that follow. Firstly, we construct a features map from the given 3D model of n vertices with a size $m = \lfloor \sqrt{n} \rfloor$. The features map cells are filled with the saliency degree of vertices. Then, we apply the template matching method [75] between different 3D models features maps. We mention that using the saliency degree and the template matching algorithm, we can detect some parts of the object. The tables (TABLE 5.4 and TABLE 5.5) shows some experimental results.

TABLE 5.4 - Detecting parts of the object

Object 1	Features Map	Object 2	Features Map	Similarity %	Part of	Match, or No match
				90%	✓	


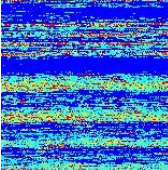

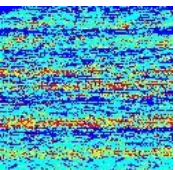

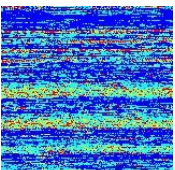

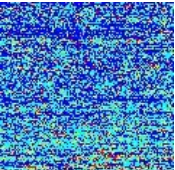

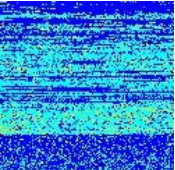

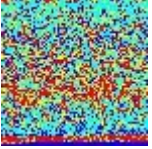

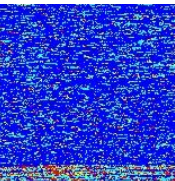
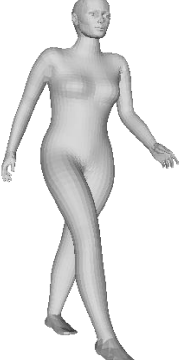
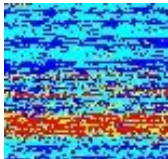

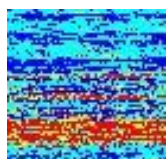
				90%	<	
---	---	---	--	-----	---	--

TABLE 5.5 - Results of different 3D models matching

				35%		Match
				50%		No Match
				50%		No Match
				98%		Match

5.4.2. Linear approach

In this section, we describe our proposed approach. We firstly compute feature regions (SECTION 5.1) then we compute the maximum matching between these regions. In order to define feature regions that represent the 3D Object surface, we compute a multiscale saliency map that describes the saliency degree of each vertex. This map is computed using multi-scale feature extraction based on rotationally invariant descriptor. Hence, the 3D object can be represented by a weighted graph, it is

sufficient to find a relational maximum subgraph matching of graphs representing the 3D objects. In this phase, we model the sub-graph isomorphism as a combinatorial optimization problem and we propose a heuristic algorithm to solve it.

5.4.2.1. 3D Object matching

In this section, we present the matching process as a combinational optimization problem using the relational maximum subgraph-matching notion and simulated annealing algorithm. The latter is defined based on the maximum graph isomorphism consisting of finding the maximum isomorphic pair of subgraphs where one is a part of the other. So, the matching problem can be seen as finding the largest common part between two shapes. We will present the graph isomorphism problem, and then we present the subgraph isomorphism to finally define the relational subgraph isomorphism.

The matching problem of two objects is formulated as the minimization of a cost function and a pruning criterion so that the objects can be mapped into a relational graph representation. Let $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ be two simple graphs with N vertices each and A and B the associated adjacency matrices. Our goal is to find a bi-continued one-to-one correspondence, Π between the set of graphs vertices, which minimizes the distance between these graphs. The classical graph-distance can be defined as:

$$J(\Pi) = \sum_{k=1}^N \sum_{l=1}^N (E_A(v_k, v_l) - E_B(\Pi(v_k), \Pi(v_l)))^2 \quad (5.14)$$

The graph matching problem can be derived from equation (5.14) using the Euclidean matrix norm as

$$J(P) = \|A - PB P^t\|^2, \quad (5.15)$$

where P is a permutation matrix and its coefficient is defined as :

$$P_{ki} = \begin{cases} 1 & \text{if } \Pi(V_k) = V'_i \\ 0 & \text{otherwise} \end{cases} \quad (5.16)$$

The graph isomorphism is reduced to the problem of finding the permutation matrix that minimizes (EQUATION 5.15).

To model the subgraph isomorphism problem, equation (5.15) can be rewritten as:

$$J(P) = \|A\|^2 - 2A \cdot PB P^t + \|PB P^t\|^2 = \sum_{k=1}^N \sum_{l=1}^N \sum_{i=1}^M \sum_{j=1}^M p_{ki} b_{ij} p_{lj} - 2 \sum_{k=1}^N \sum_{l=1}^N \sum_{i=1}^M \sum_{j=1}^M a_{kl} p_{ki} b_{ij} p_{lj}, \quad (5.17)$$

where N and M are the number of nodes of graph A and B respectively. As the coefficient of p can be rewritten using kronecker symbol, the cost function can be simplified as:

$$E = \sum_{k=1}^N \sum_{l=1}^N (1 - 2a_{kl}) b_{\Pi(k)\Pi(l)} \quad (5.18)$$

Finally, the relational subgraph isomorphism criteria is expressed as:

$$E = \sum_{s=1}^S w_s E^s, \quad (5.19)$$

where w_s is the weight a for each relationship s , E^s is defined by an equation as (EQUATION 5.18).

The pruning criteria is computed as:

$$D_k = \sum_{s=1}^S \left(\sum_{l \neq k, l=1}^N (W_A^s(V_k, V_l) - W_B^s(\Pi(V_k), \Pi(V_l)))^2 \right) \quad (5.20)$$

To apply the simulated annealing algorithm, we define states, state transition, the random generation of state transitions, and its associated energy changing.

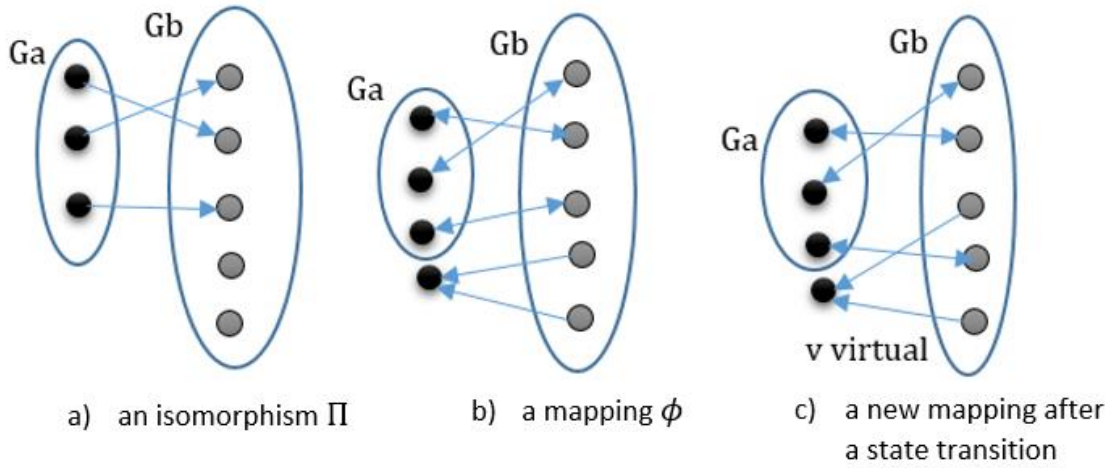


FIGURE 5.13 - An isomorphism and a slate transition to a new isomorphism

Given two graphs G_A and G_B where A is smaller than B , the matching problem between these two graphs is formulated as finding a subgraph of G_B isomorphic to G_A that minimizes the criterion of (EQUATION 5.19). To formulate the simulated annealing algorithm using the graph matching property, a state is considered as an isomorphism and the possible states space is the possible isomorphism set. Let Π be an isomorphism that maps G_A to a subgraph of G_B . The mapping ϕ is defined as:

$$\begin{cases} \phi(v_B) = v_A & \text{if } \Pi(v_A) = v_B \\ \phi(v_B) = v_{virt} & \text{otherwise} \end{cases} \quad (5.21)$$

This mapping is considered from G_B to $G_A \cup v_{virt}$ where v_{virt} is a node virtually added to G_A and not connected to the nodes of G_A .

A state transition is created by varying the correspondents of two vertices within a mapping to obtain a new one. The random state changing is generate as following:

1. Select a random vertex v_A in G_A with $\Pi(v_A) = v_B$ its correspondent in G_B .
2. Select a random vertex v'_B of G_B different than v_B .
3. Swap $\phi(v_B)$ and $\phi(v'_B)$ in $G_A \cup v_{virt}$.

The energy changing associated with the state transition is defined after some algebra as:

$$\Delta E = 2 \sum_{s=1}^S w_s \sum_{m=1, m \neq k, l}^N (b_{i\Pi(m)}^s - b_{j\Pi(m)}^s)(a_{km}^s - a_{lm}^s) \quad (5.22)$$

5.4.2.2. Experimental results

In this subsection, we have tested and evaluated our proposed approach using several 3D objects taken from the dataset “MeshsegBenchmark” [76] which contains about 400 model, covering a wide variety of 3D objects and conditions. The effectiveness of our approach is shown in the terms of distinctiveness, robustness and invariance.

In (FIGURE 5.14) we show the matchings calculated for objects of different articulations. Although the shapes is deformed due to the movement of arms and legs, the proposed approach produce a correct matchings.

As shown in the second row, one can see easily that the saliency maps for all shapes are approximatively similar.

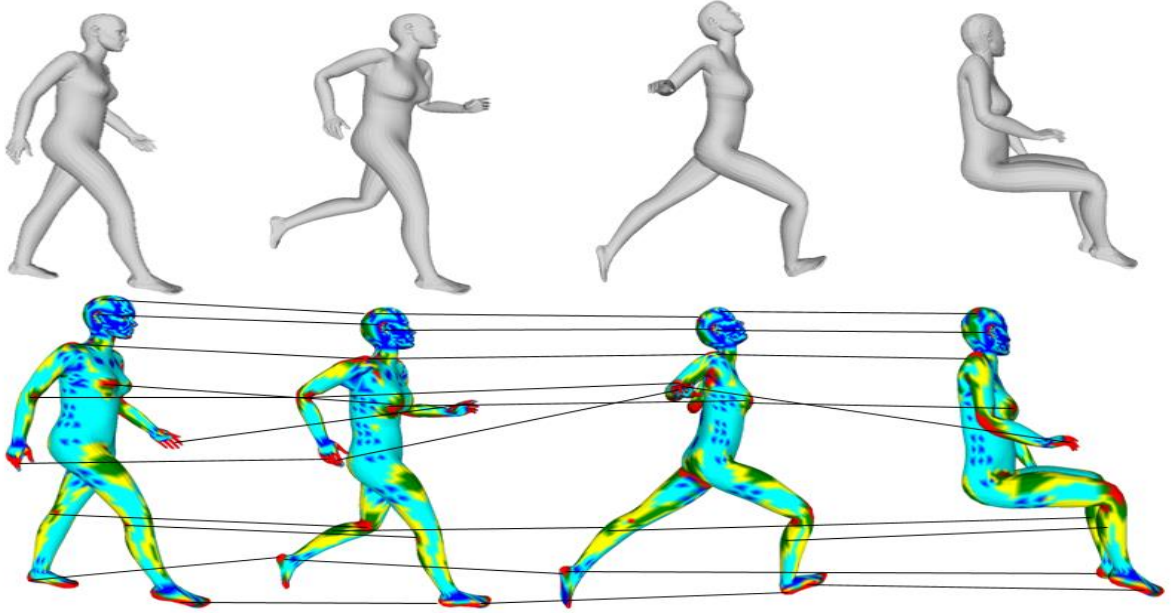


FIGURE 5.14 - Matching articulated shapes. The correspondences are imagined both by connecting lines and by showing saliency map

The figures (FIGURE 5.15 and FIGURE 5.16) demonstrate that our method is invariant to rotation, translation, and scaling. In (FIGURE 5.15), each image from the images (a, b, c, d and e) shows a matching between two articulated shapes.

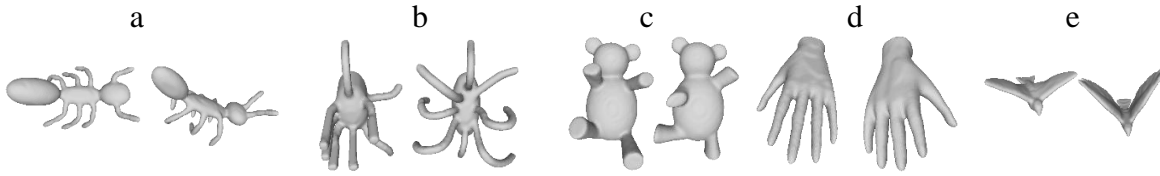


FIGURE 5.15 - Matching between articulated shapes

In (FIGURE 5.16), we show the matching results between “Gorilla” models under different transformations, with holes and simplification. Moreover, the second row shows that these transformations applied on the 3D Gorilla model, does not change significantly the saliency maps.

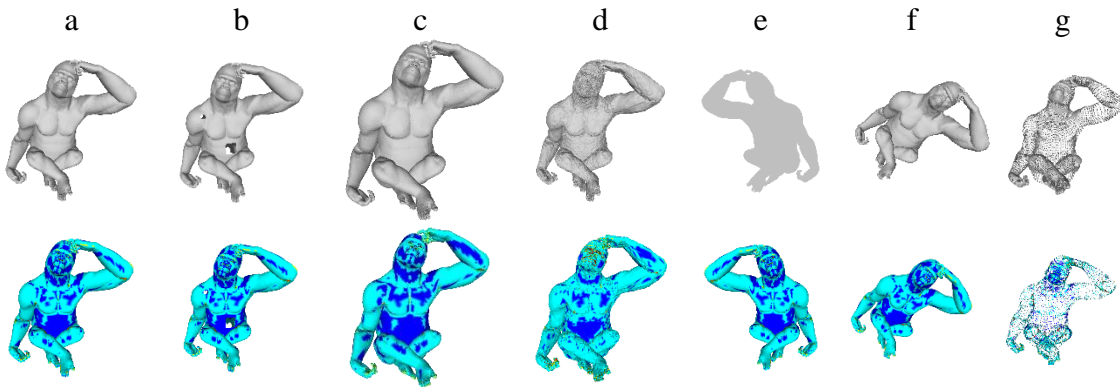


FIGURE 5.16 - Matching between Gorilla mesh with many transformation. The transformations are (from left to right): (b) holes, (c) scale, (d) noise, (e) flip x, (f) rotate and (g) simplified

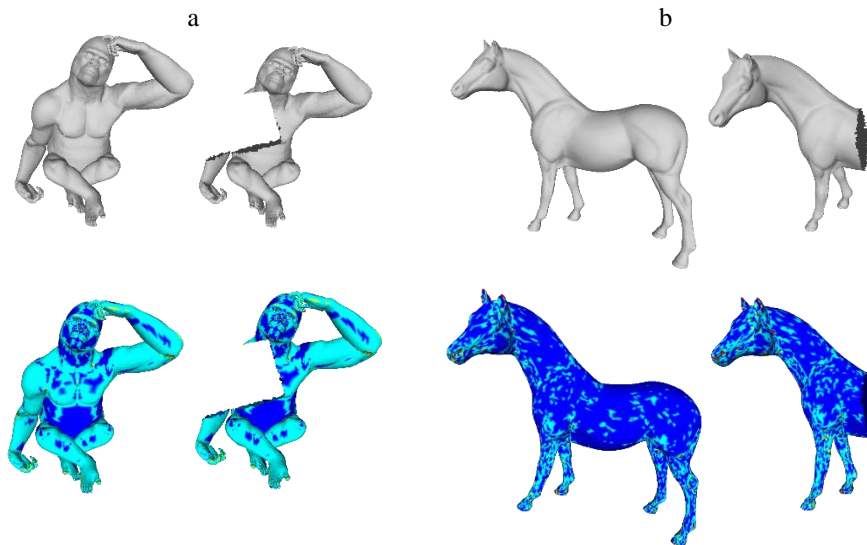


FIGURE 5.17 - Matching Detection of parts of 3D shapes



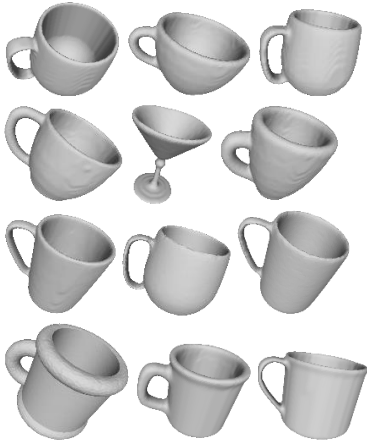

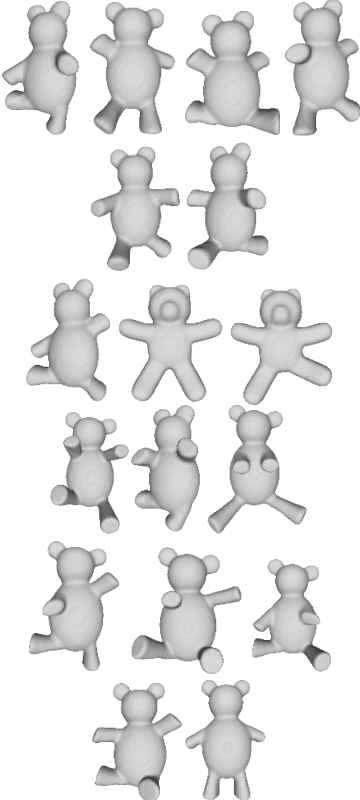

In (FIGURE 5.17), we show that our method can detect some parts of the object. In addition, we can observe that the saliency map of parts found in the saliency map of the whole object.

5.4.2.3. Search experiments

In this subsection, we present a new dynamic comparisons of 3D objects based on a multi-scale saliency map of the object. Consequently, a dynamic indexing of 3D objects allow searching for objects based on 3D object request.

Lastly, we performed an experiment to search for two objects from among 100 mesh models taken from [76]. In this case, the Saliency map for each of the 100 meshes were constructed in advance. In conducting the search, we select one model from the 100 models, then the similarities between it and the other models are calculated, and the models are shown if the resulting similarity is greater than 75%. The (TABLE 5.6) shows example results of the experiment.

TABLE 5.6: Retrieval results using the [76] Shape Benchmark. The query 3D objects are shown in the second row. The retrieved objects and missed objects are shown in row 4

Query			
			
Retrieved models	Missed models	Retrieved models	Missed models
			

5.4.2.4. Quantitative evaluation

In order to show the exact position of our method against the methods presented in the literature, we should find a quantitative evaluation of our method. In addition to presenting a sample of visual results, that shows the advantages of our proposed approach, the **Table 2** shows numerical results in term of accuracy.

In this chapter, we measure the accuracy as following:

$$Accuracy = 100 - (FPR + FNR), \quad (5.23)$$

where *FPR* (False Positive Rate) and *FNR* (False Negative Rate) can be computed as follows:

$$FPR = \frac{\text{number of false alarms}}{\text{number of correct matches}} \times 100, \quad (5.24)$$

and

$$FNR = \frac{\text{number of missed matching}}{\text{number of all models}} \times 100. \quad (5.25)$$

In our experiments, we select 50 mesh models taken from [76] to be used as query 3D object. Then the accuracy of our method is considered as the average of values of accuracy measured in each query.

TABLE 5.7: Quantitative evaluation of our proposed method in term of accuracy

3D models count	400
False positive rate (FPR)	5.29
False negative rate (FNR)	4.73
Accuracy	89.98

In contrast, each method of the methods presented in the literature is evaluated differently. Nonetheless, it shows that our proposed algorithm can compete with state-of-art matching and recognition algorithms in terms of accuracy.

5.4.2.5. Others applications

The 3D object recognition problem can be seen as a 3D matching problem. In the (SECTION 5.4.2.2), we showed that our method could detect parts of the object. Moreover, we presented a framework for searching 3D objects by query shapes.

In this context, we propose a Face recognition application, which was realized in two stages:

- First, we build a data bank of 3D faces regions where each face is a cut from human 3D model. Afterward, we assign for each face region a unique ID.
- Finally, by querying the dataset of faces using 3D human object as query shape, the system gives the ID of face that contains. Consequently, the face is recognized.

5.4.2.6. Comparison between the proposed approaches

In the previous chapter, we have presented two approaches for salient regions detections on 3D surface. Based on these approaches, two methods for 3D objects matching are released. Each of these methods is rely on a saliency approach.

During our experiments, we have noticed that the second method of 3D objects matching achieves good results with articulated objects comparing with first method. This superiority is due to the invariant descriptor used by the corresponding saliency approach.

5.5. Conclusion

In this chapter, we have presented several applications based on our saliency methods. Furthermore, we show the robustness of our proposed applications through different experimental results. Finally, we present the stability and robustness of our methods with respect to noise, scaling, translation, rotation, etc. In the next chapter, we will present a method for 3D face detection, which is an application based on our saliency and skin region detection methods.

Chapter 6

3D FACE DETECTION BASED ON GEOMETRICAL AND SKIN COLOR FEATURES

Summary

6. 3D FACE DETECTION BASED ON GEOMETRICAL AND SKIN COLOR FEATURES ..	101
6.1. Introduction	102
6.2. Surface modelling and normal estimation	104
6.3. Proposed method.....	104
6.3.1. Notations and definitions	104
6.3.2. Candidate face region	104
6.3.3. Saliency detection	104
6.3.4. Skin and saliency regions segmentation	105
6.3.5. Face detection method	106
6.4. Experimental results	107
6.4.1. 3D point clouds and 3D meshes	107
6.4.2. 2D images	113
6.5. Comparison with the state-of-the-art	115
6.6. Conclusion	118

Face detection has an essential role in many applications. In this chapter, we propose an efficient and robust method for face detection on a 3D point cloud represented by a weighted graph, which is another application based on our methods of skin color and saliency detection. This method is based on a data mining predictive model as proposed before. Then, the saliency degree of vertices is computed to identify the possible candidate face features. Finally, the matching between non-skin regions representing eyes, mouth and eyebrows and salient regions is done by detecting collisions between polytopes, representing these two regions. This matching process, lead to double face detection, one based on color and second based on geometry. This method extracts faces from situations where pose variation and change of expressions can be found. The robustness is showed through different experimental results. Moreover, we study the stability of our method according to noise. Furthermore, we show that our method deals with 2D images.

6.1. Introduction

Face detection becomes an important research field for computer vision applications such as human recognition and verification, analyzing emotions for multimedia tasks, and visual tracking. Nowadays, the development of biometric systems has become an important challenge for researchers. Fingerprints and iris techniques are the mostly used, but the face recognition seems to be the best approach especially in airports and critical zones.

Most of face detection approaches for 2D images present many drawbacks such as illumination variations, pose and facial expressions. In chapter 2, we have presented an efficient method for skin region detection on 3D point clouds. Then, a definition of human face based on skin color is proposed. The results produced by this method are good in term of detection rate. However, some skin regions are interpreted as human face and in fact are not. This false positive detection is occurred because the rules of human face are verified sometimes in other real faces regions. To deal with this problem and minimize the false positive detections we should add other rules based on geometrical features beside the skin color information. In this context, this chapter presents an accurate method for face detection on 3D colored point clouds due to their less sensitivity to lighting conditions, viewpoint and to the less false positive detection rate.

In order to define the geometrical features of a human face, we have benefit from the application of salient region segmentation proposed in the previous chapter. Furthermore, some parts of the human face can be seen as salient regions. Thus, we can define a human face as a skin region containing some salient regions. Consequently, this method confirm the presence of a face by double detections, one based on definition of human face based on skin color and second based on definition lies on salient regions.

Our main contribution in this work is to propose a new method to extract 3D face parts from a colored point cloud represented by a weighted graph. The face parts are seen as gaps (non-skin regions) and salient regions.

Our method steps can be summarized as follows:

- Modelling the 3D colored point clouds surface by an undirected weight graph.
- Estimating vertices normal.
- Identifying the skin vertices.
- Using a custom version of Hill Climbing algorithm to segment the graph vertices into skin, non-skin and salient regions.
- Detecting and locating gaps (non-skin regions) in skin regions.
- Defining the candidate human face region using skin and gap regions.
- Finding salient regions in each candidate skin region.
- Using the collision between polytopes representing gaps and salient regions, we confirm the face detection.

The following flowchart in (FIGURE 6.1) illustrates our method.

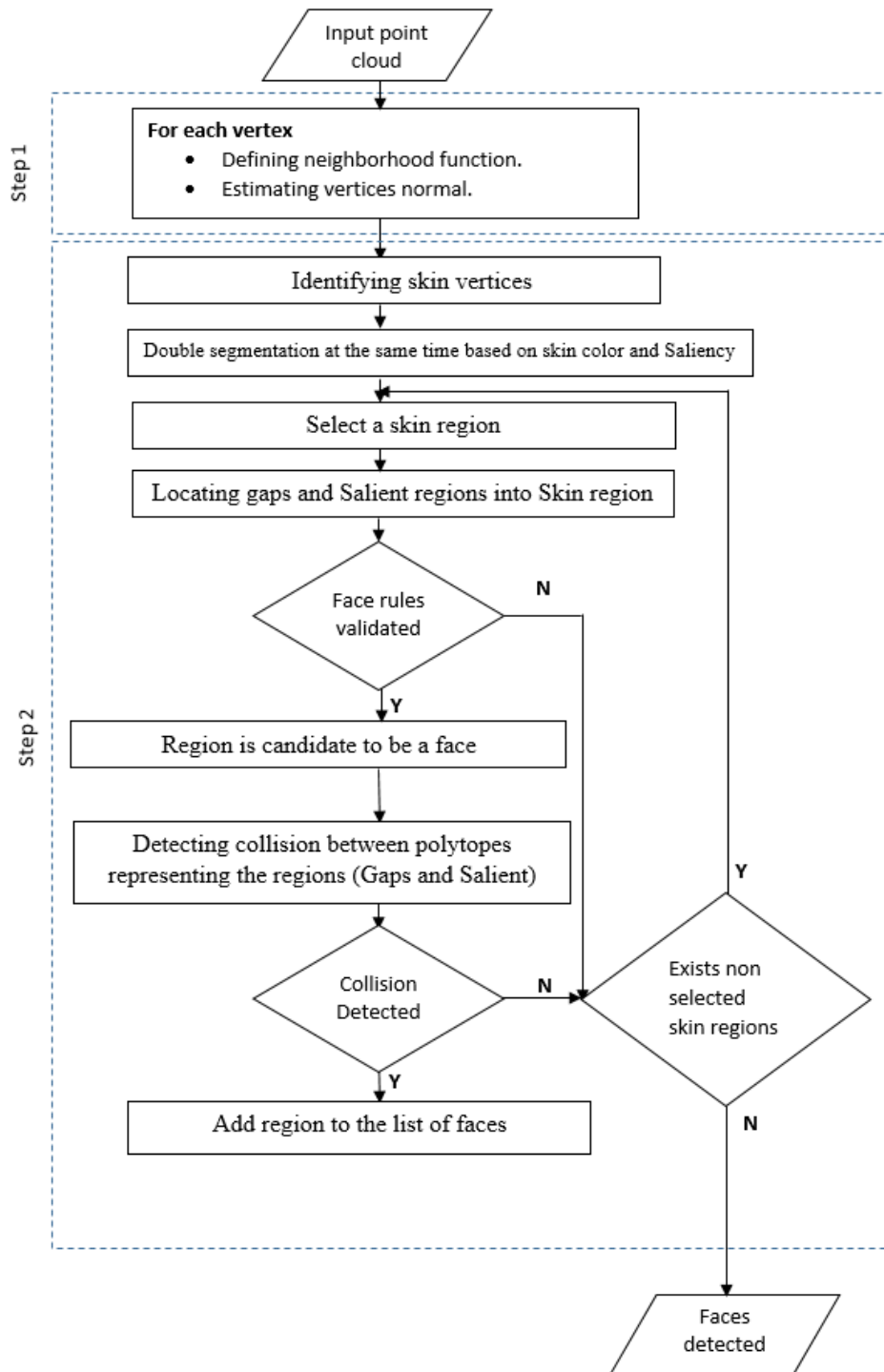


FIGURE 6.1 - Flowchart of our method

Finally, in this method, we convert the 3D colored point cloud into a weighted graph, and then we detect faces using the skin and saliency features of vertices. This proposed method deals with any data that can be represented by a weighted graph such as 3D point clouds, 3D meshes and 2D images.

6.2. Surface modelling and normal estimation

In this section, we adopt the same approach proposed in (CHAPTER 4) for modeling the 3D surface with a main difference concerning the edges weigh.

Let us consider the general situation where a point cloud can be viewed as an undirected weighted graph $G_d(V, E, W_d)$ where V represents a finite set of vertices and $E \subset V \times V$ a finite set of edges. Each edge $(v_i, v_j) \in E$ that connects two vertices v_i and v_j has a set of weights W_d that contains two weight functions, W_c the weight associated to color information and W_g the weight associated to compute saliency.

The set of neighbors $D_n(v_i)$ of a given vertex v_i is defined using the concept of Delaunay triangulation. Then, we compute the normal vector $N(v_i)$ and the 2-directional vectors for each vertex v_i following the x - and y - axes using the Eigen-values of the covariance matrix.

6.3. Proposed method

Our method consists of detecting a human face using skin and geometry information. In this section, we present firstly some useful notations to generate face candidate regions. Then, we show the saliency degree calculation of vertices in the extracted candidate regions. Finally, we present the method of matching candidate regions based on colors and saliency measure.

6.3.1. Notations and definitions

We begin this subsection by a brief review of some notations and definitions involved in this section.

Let $E_q(v_i, v_j): V \times V \rightarrow \{0,1\}$ be a function that can be defined as following:

$$E_q(v_i, v_j) = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ are considered skin} \\ 0 & \text{otherwise} \end{cases} \quad (6.1)$$

We define a function $H_m(v_i): V \rightarrow \mathbb{N}$ that measures the homogeneity between v_i and its neighbors as following:

$$H_m(v_i) = \sum_{v_i \sim v_j} E_q(v_i, v_j), \quad (6.2)$$

6.3.2. Candidate face region

The first phase in our method consists of defining a set of face region candidates based on skin information. To do so, we adopt the same approach detailed in (SECTION 3.4).

6.3.3. Saliency detection

Once we define the set of face candidate regions, we calculate the saliency degree of vertices (CHAPTER 4) in these regions to confirm the presence of a human face.

6.3.4. Skin and saliency regions segmentation

In this subsection, we introduce a new version of “**Multi Search Hill Climbing – Algorithm**” presented in the previous chapter. The main key feature of this version is the double segmentation of the graph based on color and geometric characteristics of vertices. In this context, during the segmentation process, a vertex will have two label holders, a label for the segmentation based on skin colors and a second label for the segmentation based on the saliency measure. The first step in our method consists of assigning the same label in the first and second label holder to each group of connected vertices that belongs to the class $F(G_d)$ and the similar vertices in terms of saliency. This step can be performed using a custom version of Hill Climbing algorithm. This algorithm uses a dilation morphological operator to propagate through weighted graph vertices and label homogenous regions in terms of skin and saliency similarities, respectively. Then, the algorithm traverses all graph vertices and creates a segment from each set of vertices having the same label. As a result, this algorithm produces two lists of segments (list of skin segments and list of salient segments) where a vertex can belong to skin and salient segment at the same time.

Multi Search Hill Climbing (Version 2) – Algorithm

Let $C'(v_i)$ be a set of neighboring vertices of a vertex v_i and similar to v_i in term of saliency. We define a vertices labeling function $E'_q(v_i, v_j): V \times V \rightarrow \{0,1\}$ as following:

$$E'_q(v_i, v_j) = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ are similar in term of saliency} \\ 0 & \text{otherwise} \end{cases} \quad (6.3)$$

We define a function $H'_m(v_i): V \rightarrow \mathbb{N}$ that measures the homogeneity between v_i and its neighbors as following:

$$H'_m(v_i) = \sum_{v_i \sim v_j} E'_q(v_i, v_j) \quad (6.4)$$

The custom version of Hill Climbing algorithm steps are listed as following:

- Pick a random vertex v_i in the weighted graph.
- Consider all the neighbors $D_n(v_i)$ of the current vertex v_i .
 - Choose the set of neighbors $C(v_i)^2$ from $D_n(v_i)$ that maximize the function $H'_m(v_i)$.
 - Apply the dilation operator from v to all vertices in $C(v_i)$.
 - Choose the set of neighbors $C'(v_i)$ from $D_n(v_i)$ that maximize the function $H'_m(v_i)$.
 - Apply the dilation operator from v_i to all vertices in $C'(v_i)$.
- Repeat 2 for each element in $C(v_i)$ and $C'(v_i)$ until $C(v_i)$ and $C'(v_i)$ are empty.

² For more details, see (CHAPTER 3 - SECTION 3.3)

FIGURE 6.2 illustrates the behavior of the precedent algorithm.

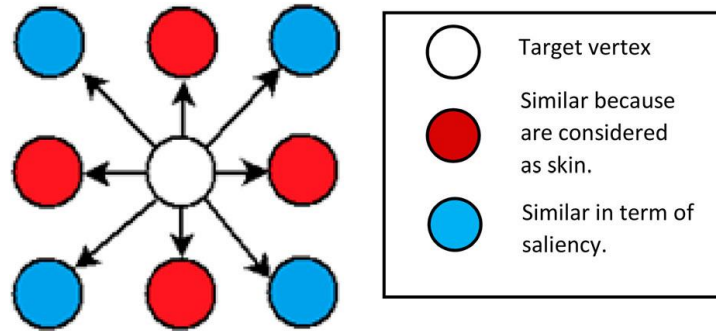


FIGURE 6.2 - Illustration of Multi Search Hill Climbing algorithm step

6.3.5. Face detection method

In this subsection, we describe the process of matching candidate regions based on colors and saliency measure.

The definition of candidate face regions is based on gaps features (non-skin regions) located in skin regions. Therefore, each non-skin and salient region can be considered as a convex polytope, because it contains a finite number of vertices, it is bounded and closed [77]. In (FIGURE 6.3), we present some examples of polytopes. The face detection process consists of checking whether there is a matching between the gap regions defining the face candidate regions and those of salient regions. The matching process is done by detecting collisions between polytopes representing these regions (gaps and salient regions).

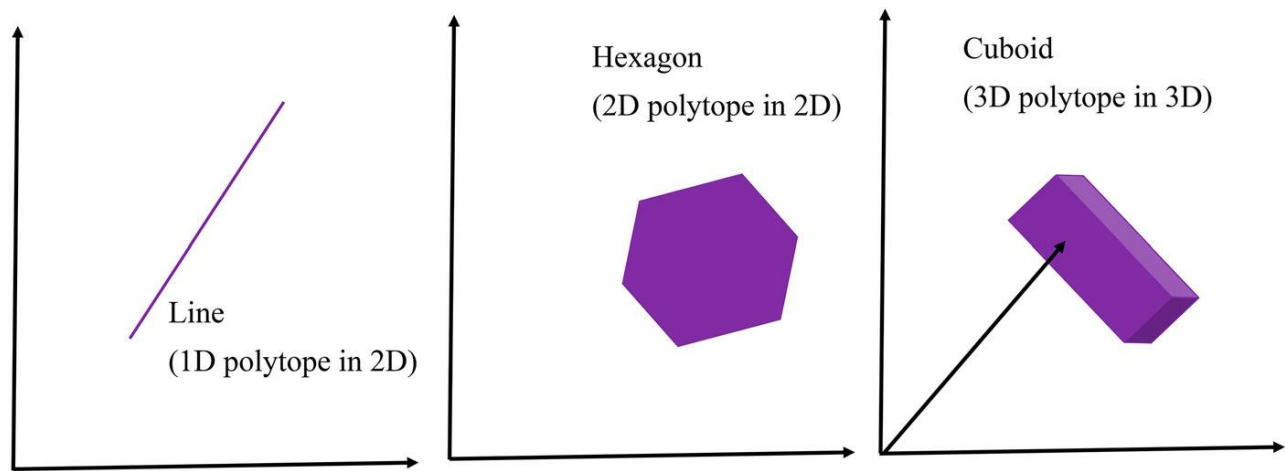


FIGURE 6.3 - Examples of polytopes

Let R be a region in G_d (Set of connected vertices), and $C_o(R) \subset \mathbb{R}^n$ where $n \in \mathbb{N}$ the set of polytope corners corresponding to R . According to Minkowski Theorem of [78], a bounded and closed convex polytope $K \subset \mathbb{R}^n$ is the convex hull of its external points or corners.

Let S be the $C_o(R)$ of a region R . The convex hull of S is the set defined as:

$$\text{Conv}(S) = \left\{ \sum_{i=1}^{|S|} \alpha_i x_i \mid (\forall i: \alpha_i \geq 0) \wedge \sum_{i=1}^{|S|} \alpha_i = 1 \right\}, \quad (6.4)$$

Let V and W denote $C_o(X)$ and $C_o(Y)$ of the two regions X and Y respectively.

The Minkowski sum and the Minkowski difference of V and W are defined as following:

$$V+W = \{v + w \mid v \in V, w \in W\}, \quad (6.5)$$

$$V-W = \{v - w \mid v \in V, w \in W\}, \quad (6.6)$$

In (FIGURE 6.4), we show a graphical example of the Minkowski sum of two sets in 2-dimensional real space.

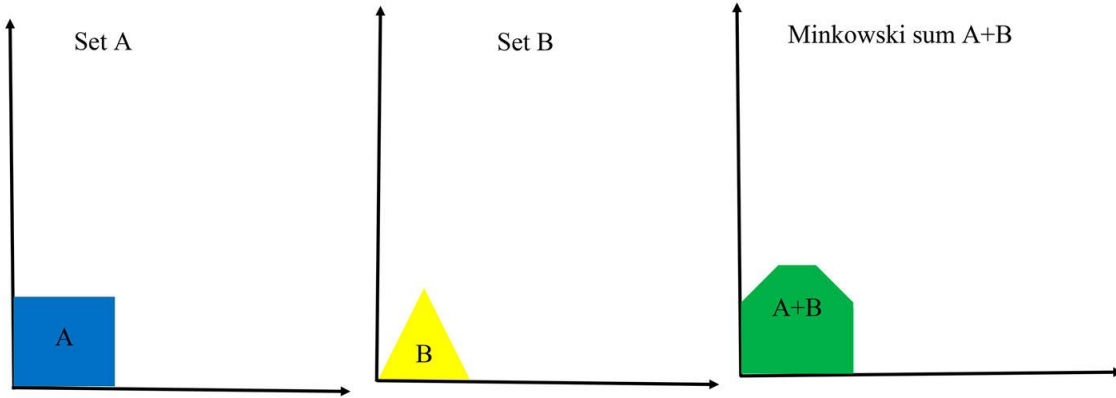


FIGURE 6.4 - Examples of Minkowski sum of two sets in two-dimensional real space

Let $D_s(S_1, S_2)$ be the function that represents the distance between two non-empty sets S_1 and S_2 which can be defined as following:

$$D_s(S_1, S_2) = \inf \{d(x, y) \mid x \in S_1, y \in S_2\}, \quad (6.7)$$

where d is a metric defined on \mathbb{R}^n and \inf is the infimum.

To detect the collision between two polytopes U and V , we compute the distance between the convex hulls of both two polytopes. Thus, two polytopes U and V are collide if:

$$D_s(\text{Conv}(C_o(U)), \text{Conv}(C_o(V))) = 0, \quad (6.8)$$

where $0 \in \mathbb{R}^n$.

As we mentioned before, both sets U and V can be represented by their convex hulls $\text{Conv}(C_o(U))$ and $\text{Conv}(C_o(V))$ of their corners, respectively. Therefore, formula (6.8) can be rewritten as

$$D_s(\text{Conv}(C_o(U)), \text{Conv}(C_o(V))) = 0$$

$$\Leftrightarrow 0 \in (\text{Conv}(\text{Co}(U)) - \text{Conv}(\text{Co}(V)))$$

$$\Leftrightarrow 0 \in (\text{Conv}(\text{Co}(U)) + \text{Conv}(-\text{Co}(V)))$$

To find a representation of $0 \in \mathbb{R}^n$ created from the sum of two convex combinations of the same point where one combination composed of points in $\text{Co}(U)$ and the other one of points in $-\text{Co}(V)$, we propose to use a linear system that can be written as following:

$$\begin{aligned} \hat{\mathbf{A}}\mathbf{x} &= 0 \\ x &\geq 0, \sum_{i=1}^{|\text{Co}(U)|} x_i = 1, \sum_{i=|\text{Co}(U)|+1}^{|\text{Co}(U)|+|\text{Co}(V)|} x_i = 1, \end{aligned}$$

with two additional constraints:

$$a_{n+1}x = 1, a_{n+2}x = 1,$$

where $\hat{\mathbf{A}} \in \mathbb{R}^{n \times (|\text{Co}(U)| + |\text{Co}(V)|)}$ that contains the elements of $\text{Co}(U)$ and $-\text{Co}(V)$, and

$$a_{n+1,i} = \begin{cases} 1 & \forall 1 \leq i \leq |\text{Co}(U)| \\ 0 & \forall (|\text{Co}(U)| + 1) \leq i \leq (|\text{Co}(U)| + |\text{Co}(V)|) \end{cases}$$

$$a_{n+2,i} = \begin{cases} 0 & \forall 1 \leq i \leq |\text{Co}(U)| \\ 1 & \forall (|\text{Co}(U)| + 1) \leq i \leq (|\text{Co}(U)| + |\text{Co}(V)|) \end{cases}$$

Finally, the linear system can be solved using the standard method of simplex and can be rewritten as:

$$\text{Min } f_A(x) = e^{T(b-Ax)}$$

Subject to:

$$\begin{aligned} \mathbf{Ax} &\leq \mathbf{b} \\ \mathbf{x} &\geq 0 \end{aligned}$$

$$\text{where } \mathbf{A} \in \mathbb{R}^{k \times m}, k=n+2, m=(|\text{Co}(U)| + |\text{Co}(V)|), \mathbf{e} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} \in \mathbb{R}^k, \mathbf{b} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 1 \end{pmatrix} \in \mathbb{R}^k, k, m \in \mathbb{N}.$$

Note that if the above linear program has a valid solution then the two polytopes U and V are collide.

In order to check the presence of collision between two polytopes U and $V \subset \mathbb{R}^n$, we present the following algorithm:

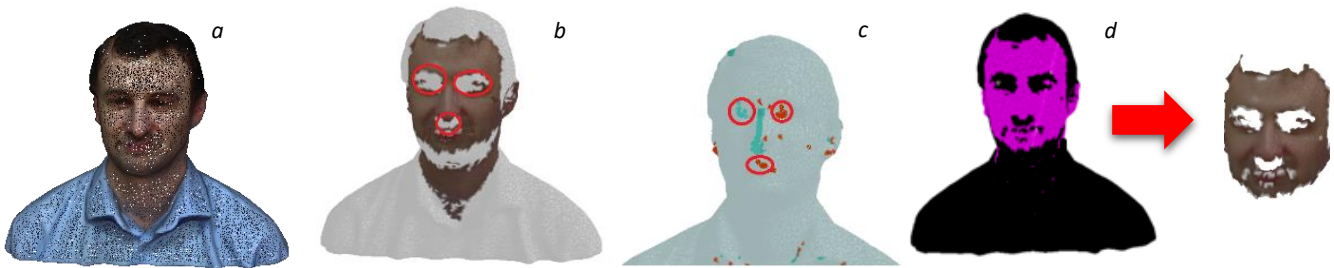
ALGORITHM 6.1: *Check collision*

- U, V : **Polytopes**
- Get the list of two polytopes corners.
- Check If $\text{Card}(\text{Co}(U)) = 0$ or $\text{Card}(\text{Co}(V)) = 0$ then there is **no collision**.
- Check if $\text{Card}(\text{Co}(U)) = \text{Card}(\text{Co}(V)) = 1$ and the two points are equals then the two polytopes are **colliding**.
- Check if $\text{Card}(\text{Co}(U)) > 1$ or $\text{Card}(\text{Co}(V)) > 1$
- Solve the linear program come from formula (26).
- If the obtained minimal value is **zero** value then the two polytopes **are colliding**.

*End***6.4. Experimental results****6.4.1. 3D point clouds and 3D meshes**

In this section, we show the efficiency and robustness of our method by applying it on 3D meshes, and 3D point clouds. In our experiments, we used a sample of 140 colored objects containing faces of different color of skin, ages and positions. We will show and discuss the influence of different parameters on our approach such as graph Delaunay-level k , the variable x in (EQUATION 4.10) that defines the deviation factor threshold in patches entropy calculation, and the number of patch cells in (SECTION 4.2.2.2).

In (FIGURE 6.5), we present the results of our face detection method applied on a colored point clouds. Images (a, e and k) show the original 3D point clouds, images (b, f and l) present the face regions candidates where the gaps defining the face are circled in red. Images (c, g and m) show the salient regions detected in the candidate face, using our method with the following parameters values ($k=1$, $m=9$ and $x=12$). Images (d, h and n) display the detected faces by our method. The fuchsia areas on the 3D point clouds object represent the detected faces. We can visually confirm the face detection in these point clouds.



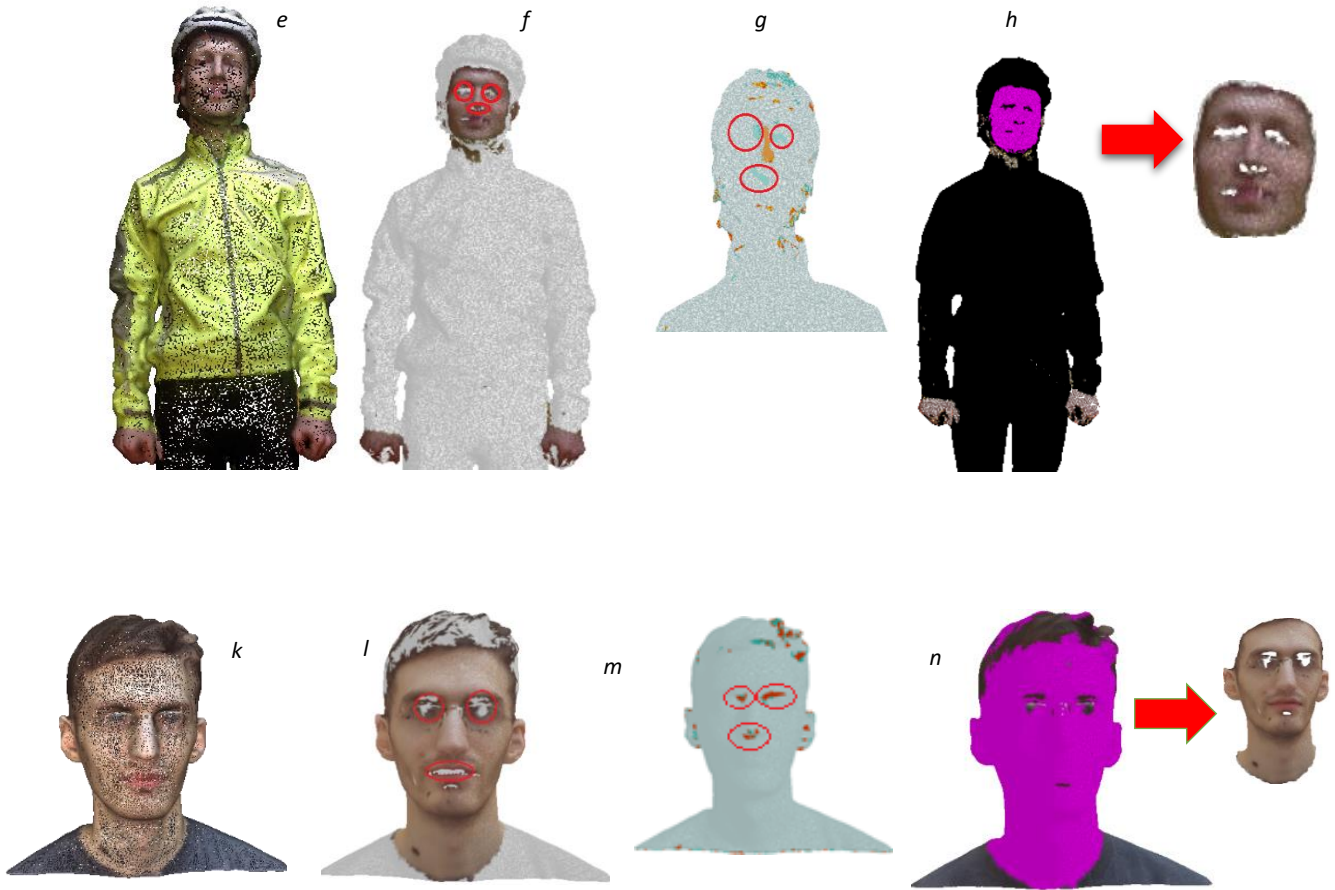


FIGURE 6.5 - Results of applying our proposed model for face detection on some 3D colored point clouds

In (FIGURE 6.6), images (a and c) present the original 3D colored point cloud. Images (b and d) shows the results obtained by applying first phase of our method (identifying the candidates regions). As shown below, two candidates regions are identified in the same object. The gaps defining the face are circled in red.





FIGURE 6.6 - Two candidates regions are identified by applying first phase of our proposed method

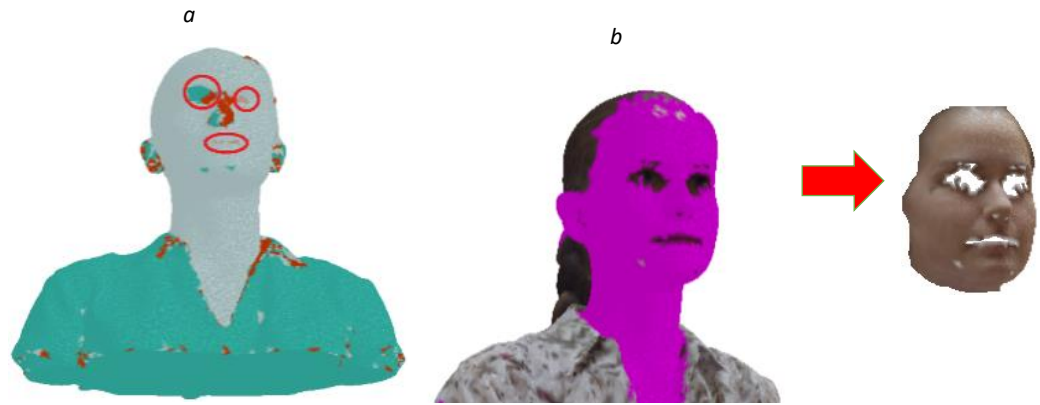


FIGURE 6.7 - Two candidates regions detected by first phase and a single face recognized after applying second phase

In (FIGURE 6.7), we show the result of applying the second phase of our method on the 3D point clouds presented in the previous paragraph (FIGURE 6.6). Image (a) displays the salient regions. The salient regions that collide with the gaps defining the candidates regions are circled in red. Note that there are salient regions located at the border of the second candidate region far than the gaps. Consequently, this candidate region is ignored.

FIGURE 6.8 presents the results of our method with a deviation factor $x = 9$.

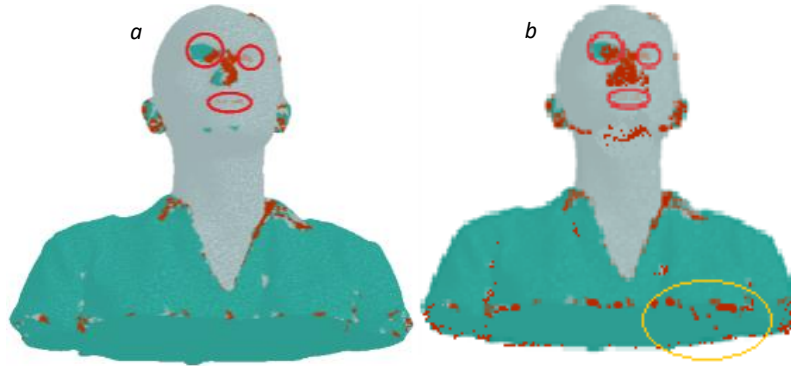


FIGURE 6.8 - Influence of parameter deviation factor threshold x : image a shows the salient regions detected with $x=12$ and image b shows the salient regions detected with $x=9$

As shown in (FIGURE 6.8), we notice that the number of salient regions increases when we decrease the deviation factor threshold. Practically, the similarity among the vertices relatively increases when the deviation factor threshold increases. This increasing in the number of salient regions lead to interpret the region candidate in the (FIGURE 6.6) (d) as human face where salient regions are detected near the gaps and collide with them.

Furthermore, we notice that the number of salient regions with a local graph is greater than the number of regions detected with a non-local graph due to the larger number of neighbors information.

FIGURE 6.9 shows the result of applying our method on 3D colored point clouds and meshes containing multi faces. Images (a, c) show the original point clouds and images (b, d) show the results where the region colored with fuchsia are the detected faces. Image (e) shows the original 3D mesh and image (f) shows the detected faces.

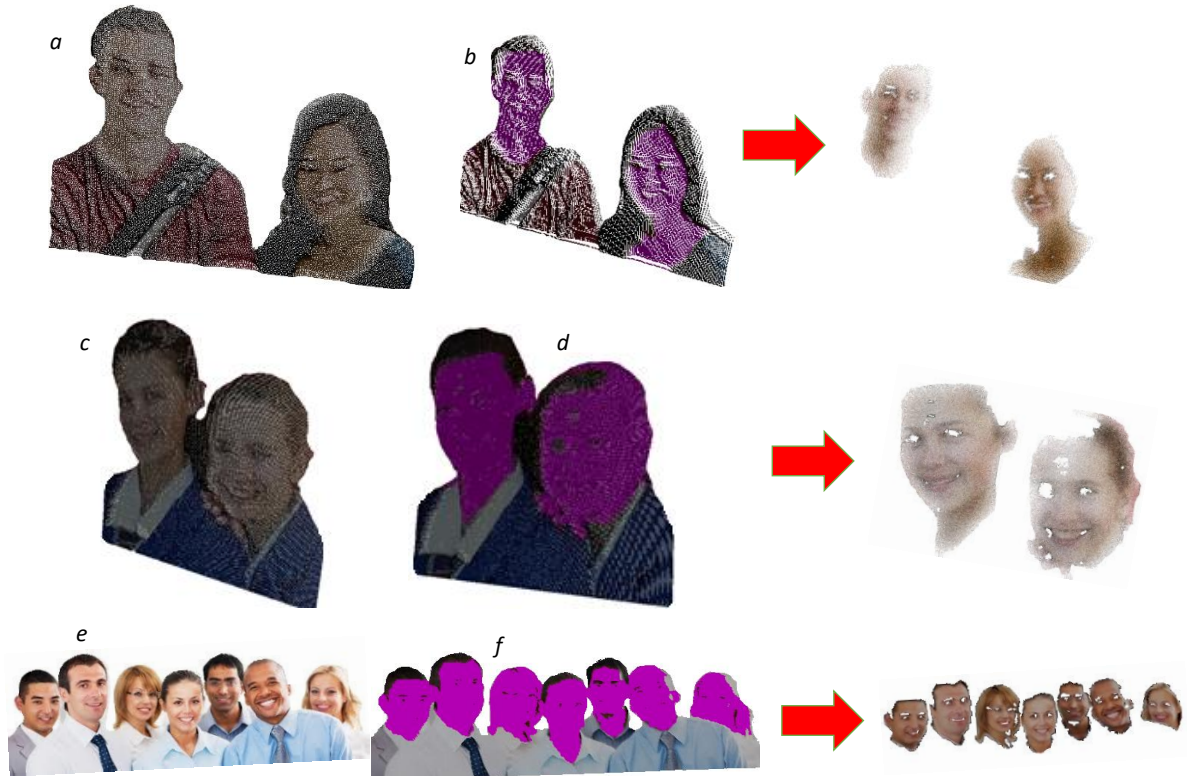


FIGURE 6.9 - Using our method with 3D point clouds and meshes containing multi faces

FIGURE 6.10 presents the result of applying our method on the 3D mesh without color. We can observe that the detected salient regions are located approximately in place of gaps that define the candidate face region. Thus, we can conclude that our method consists of two phases one detect a face and second affirm the detection.

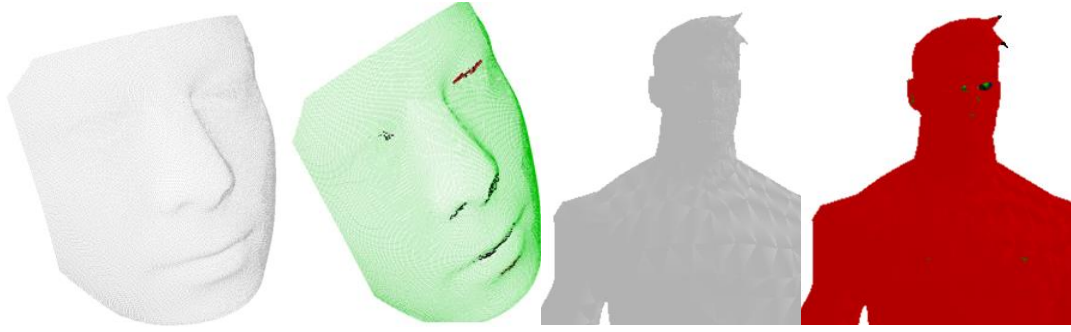


FIGURE 6.10 - Using our method with 3D mesh not colored

In addition to the visual results presented above, and in order to evaluate better the proposed method, the (TABLE 6.1) shows some numerical results:

TABLE 6.1 - Results obtained from 3D point cloud

	<i>Current method</i>	<i>Method of Chapter 3</i>
Objects count	145	145
Number of all faces	385	385
Number of truly detected faces	366	341
False positive rate (FPR)	3.88%	8.8%
False negative rate (FNR)	5%	3.63%
Accuracy	91.12%	87.57%

The accuracy of our proposed method is measured using (EQUATION 3.10):

As mentioned recently in this section, the variation in the parameters x and k can affect the number of salient regions detected. Consequently, the error rate (FPR) is affected as shown in (FIGURE 6.11).

FIGURE 6.11 - Influence of x and k on face detection process

6.4.2. 2D images

As mentioned before, our method can be applied on any data that can be represented by a graph such as 2D colored images and 3D point clouds. To deal with 2D images, we construct a K -NN graph, and then we construct the patch at a vertex v_i as a rectangle of m cells with a cell length $L(v_i)/m$. We

assign to each patch at v_i the average of gradient of each vertex v_j according to v_i . The weight function ω_g is calculated as

$$\omega_g(v_i, v_j) = e^{-\frac{|A(v_j) - A(v_i)|}{\sigma(v_i) * \sigma(v_j)}}, \quad (6.9)$$

The figures below show some results of applying our method on some 2D colored images. Note that a rectangle is drawn around the detected face region as shown in (FIGURE 6.12).

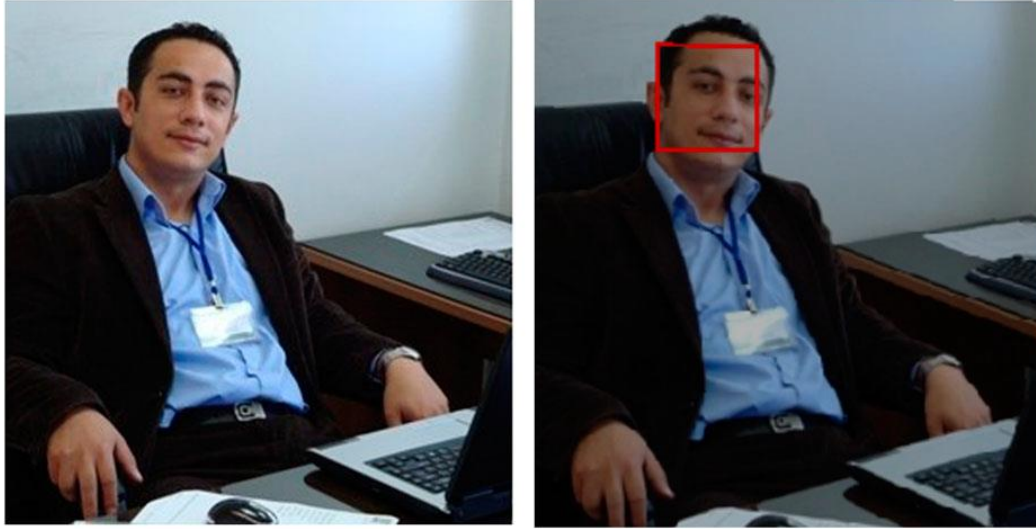


FIGURE 6.12 - Using our method with 2D image



FIGURE 6.13 - Using our method with 2D colored image containing multi faces

In the (FIGURE 6.13), all faces are detected using our method. The (FIGURE 6.14) presents a face detection from a picture with a complicated background.



FIGURE 6.14 - Using our method with 2D colored image containing face and has a complex background

The (FIGURE 6.15) demonstrates the efficiency of our method with images of different skin color.

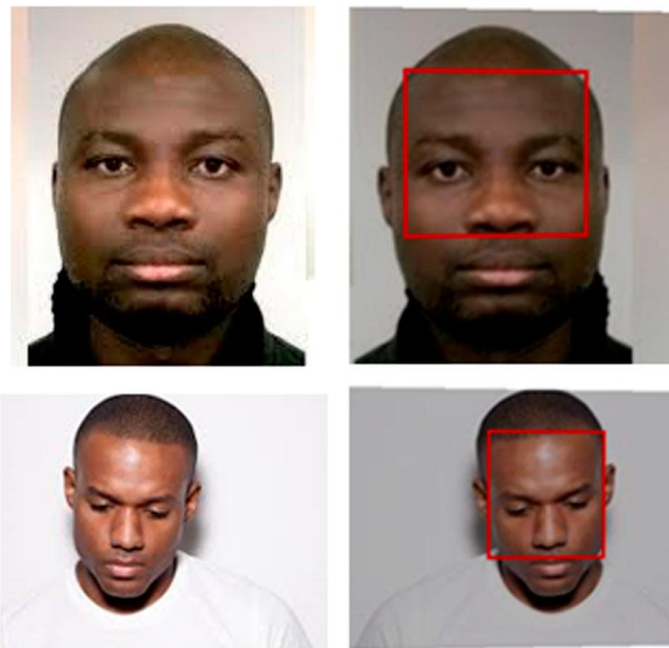


Figure 6.15 - Using our method with 2D colored image containing face of black skin



FIGURE 6.16 - Result using our method applied on a colored image containing horizontal face

The (FIGURE 6.16) demonstrate that our method detect faces in multi directions.

Finally, our experiments made on a sample of 200 images with different lighting conditions, races, and sexes chosen from “Helen” dataset, “Faces in the Wild” Dataset “Caltech” image dataset and others according to the following parameters values ($k=1$, $m=9$ and $x=12$) in addition to the following settings:

- [$NBFormsMin=3$, $NBFormsMax=20$]: Interval of number of non-skin regions in the face region.
- [$YShMin=0$, $YShMax=4$]: Differences between the ordinates of two eyes.
- [$MinDistance=5$]: Minimum distance between eyes.

The (TABLE 6.2) shows the statistical data:

TABLE 6.2 - Results obtained from 2D images

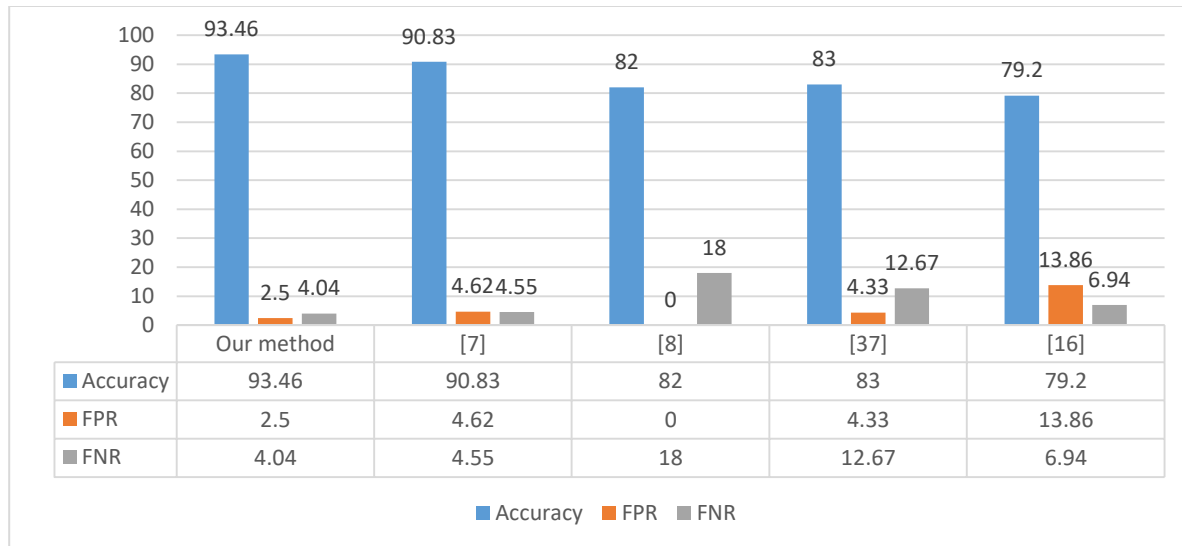
	<i>Current method</i>	<i>Method of Chapter 3</i>
Images count	200	250
Number of all faces	297	304
Number of really detected faces	279	278
False positive rate (FPR)	2.5%	7.01%
False negative rate (FNR)	4.04%	1.97%
Accuracy	93.46%	91.02%

6.5. Comparison with the state-of-the-art

In this section, we compare our method with some related works. First, note that our method deals with 3D colored meshes and 2D images. Most of the methods described in the literature deal with 2D images or 3D meshes that lie on geometry features.

As another evaluation of our work, we compare the accuracy of our method to some methods described in the literature. The (TABLE 6.3) shows the statistical data:

TABLE 6.3 - Comparison of our method and others in terms of accuracy, FPR and FNR



In addition to the statistical comparison, a visual example of failure case presented by [7] and successfully detected by our method shown in the (FIGURE 6.17):

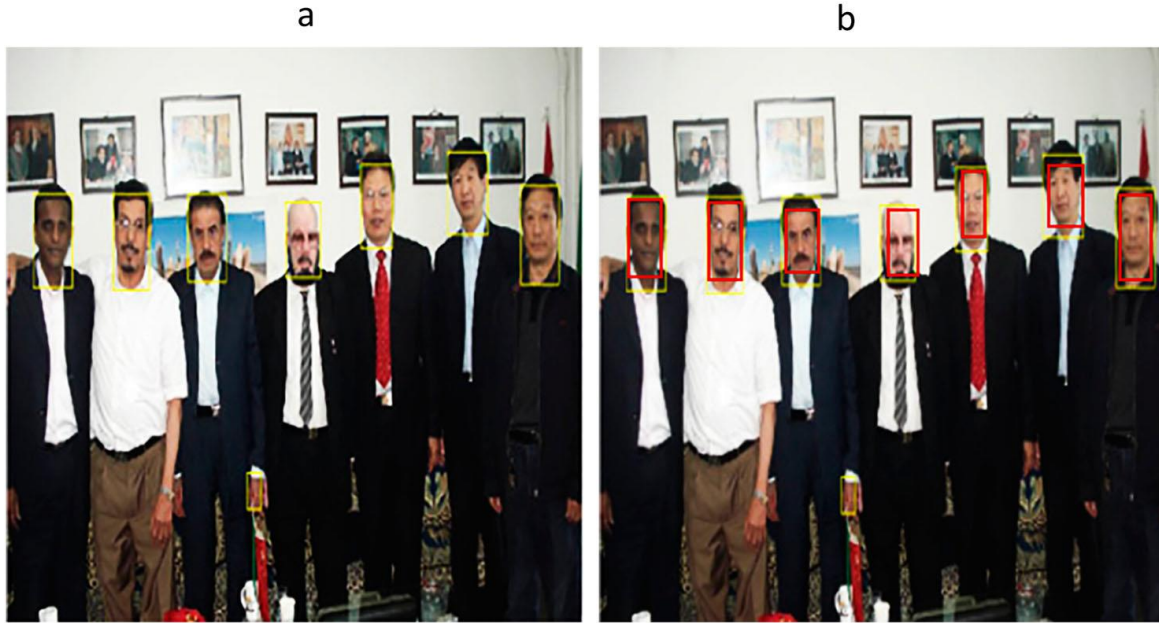


FIGURE 6.17 - Image (a) shows a screen shot from experimental result of [7], image (b) shows the result of applying our method on the same screen shot

The result obtained by [7] presents a failure case where a hand is interpreted as a face as shown in image (a). The author of [7] indicates in his experimental results that false alarms and misses still exist and this is an example. In contrast, by applying our method on the same image, all faces are detected correctly as shown in image (b). All other results presented in experimental result of [7] are also detected correctly by our method. Finally, [7] deals only with 2D colored images.

The method presented in [8] deals with 3D faces, the authors mentioned that this technique is highly sensitive to noises and to the presence of holes surrounding the regions of nose and eyes. In contrast, our technique deals with noisy 3D colored point clouds as shown in (FIGURE 6.18).



FIGURE 6.18 - Face detection from a noisy 3D colored point cloud

The approaches presented in ([9], [10]) localize faces, rather than detecting them. In [9], the presented approach is strongly limited to the dataset used where every input 3D object must contain only one face. Otherwise, our method deals with 3D colored point clouds, meshes and 2D images based on skin color containing single or multiple faces of different sizes and locations.

6.6. Conclusion

In this chapter, we have presented a novel method to detect 3D facial regions from colored point clouds and 2D images. This method was divided into two phases. Firstly, the candidates faces regions based on skin features are detected, then the salient features are computed in each detected regions to affirm the presence of human face. We also presented a simple and efficient algorithm derived from the Hill Climbing algorithm using a morphological dilation operator to label skin and salient regions. We have showed the robustness of our proposed method through some experimental results. Then, we compared our method with various state-of-arts methods.

General conclusion

In this work, we present a new approach for the detection of skin color on 3D colored mesh based on data mining techniques and linear programming using the weighted graph representation. Then, this method has been extended to detect human faces. To illustrate the robustness of our approach, we have shown that our method could solve two signal-processing problems. The first is related to the classification of vertices based on the information of the skin color and the second is related to the problem of face detection. We have also shown that our method can handle all the data that can be represented by weighted graphs such as 2D images, 3D surfaces and 3D point clouds.

On the other hand, we have also presented a new method for detecting the salient regions on 3D meshes. A local patch descriptor has been created for each vertex where the patch cell is filled with the average of the deviation factor of each vertex based on neighboring sums (average of the vertex deviation factors projected on the same location associated with the cell). The patch is then used as a local descriptor for the surface vertices of the 3D meshes. In addition, a similarity measure between the patch descriptors and a height map is calculated and to calculate the degree of similarity of the saliency degree of vertex. Finally, we have presented the robustness and efficiency of our approach by showing that our approach detects most of the visual salient regions in a 3D mesh surface. In addition, we have shown that our approach can detect salient regions on any data that can be represented by a weighted graph such as the 3D point cloud. We also presented the effectiveness of our method by presenting several applications. Among these applications, we cite the following applications:

- In order to improve the face detection process presented by the first approach, we proposed a new method to detect 3D facial regions from colored point clouds and 2D images. This method has been divided into two phases. Firstly, the face candidate regions were defined based on the characteristics of the skin, and then salient features are calculated in each detected region to affirm the presence of human face. We also presented a simple and efficient algorithm derived from the Hill Climbing algorithm using a morphological dilation operator to label the salient regions of the skin. We have proved the robustness of our proposed method through some experimental results on the failures cases of the first approach. Then we compared our method with different related methods.
- In addition, we presented a new approach for 3D point clouds simplification. The idea behind our approach is to detect the characteristic regions that describe the surface of the shape using the measure of saliency degree. The vertex saliency was calculated using a measure of similarity between the local patches. In order to detect feature regions, we presented an efficient algorithm based on the connected components labeling algorithm according to the degree of saliency of the vertices. Then we detected feature points in these regions. Note that the user can control the degree of simplification by a thresholding parameter where the simplification rate increases with the value of this parameter. Finally, we have shown the efficiency of our method through some experimental results.
- Finally, we presented a new approach to detect the matching between two 3D shapes. A new 3D invariant surface descriptor has been created for each vertex based on Zernike coefficients. In addition, a degree of saliency is calculated to detect the feature regions. Then, a multi-scale saliency map is calculated to improve the quality of the measured saliency and to cope with noise. In addition, we presented a linear algorithm to segment the 3D surface into feature regions based on the degree of salience, which are used in the matching process rather than mapping points to points. Finally, the matching process is modeled as a subgraph isomorphism problem and we

presented a heuristic algorithm to solve it. In addition, we showed the robustness of our proposed method through different experimental results. Finally, we present the stability and robustness of our method with respect to noise, scale, translation, rotation, etc.

Different parts of our work have been published in specialized international journals.

International journals

List of published articles:

1. “Efficient Image Classification using Data Mining”, International Journal of Combinatorial Optimization Problems and Informatics, Vol. 2, No. 1, Jan-April, 2011, pp. 27-44. ISSN: 2007-1558.
2. “3D face detection based on salient features extraction and skin color detection using data mining”. The Imaging Science Journal 65 (7), 2017, pp. 393-408.

<https://doi.org/10.1080/13682199.2017.1358528>

3. “Efficient 3D point clouds classification for face detection using linear programming and data mining”. The Imaging Science Journal 66 (1), 2018, pp. 23-37.

<https://doi.org/10.1080/13682199.2017.1376772>

4. “Efficient 3D mesh salient region detection using local homogeneity measure”. IET Image Processing,

<http://dx.doi.org/10.1049/iet-ipr.2017.0598>

5. “An Efficient Simplification Method for Point Cloud Based on Salient Regions Detection”. RAIRO - Operations Research.

<https://doi.org/10.1051/ro/2018082>

Conferences:

6. “An Invariant Multi-scale saliency detection for 3D mesh”, IEEE international conference applied and theoretical computing and communication technology (iCATccT-2018), (06-08 September 2018).

List of submitted articles:

7. “3D Object Matching Using Invariant Descriptors and Linear Programming”. Applied Mathematics and Computation.

In future works, we propose as a perspective:

- Some methods based on our methods presented in this work. Among these applications, we can mention:
 - **Classification of 3D objects:** The idea behind this is to classify 3D objects as human or not, for example. To do this, a database of 3D human parts is constructed such as arms and legs of different sizes. Thus, each matching between a 3D object and parts of the data set makes it possible to classify the object as a human.
 - **Colorization of digitized 3D point clouds:** The historians, the archaeologists and the curators are interested to protect the oeuvres of art and to want to put them at the disposal of a public so wide as possible in the form of the colored 3D objects. In this context, we propose as perspective an application of coloring digitized point clouds from 3D scanner based on our method of saliency detection.

Conclusion général

Dans ce travail, nous présentons une nouvelle approche pour la détection de la couleur de la peau sur des nuages de points colorés en 3D en se basant sur les techniques de fouilles de données et la programmation linéaire et en utilisant la représentation graphique pondérée. Ensuite, cette méthode a été étendue pour détecter les visages humains. Pour illustrer la robustesse de notre approche, nous avons montré que la méthode que nous proposons pourrait être utilisée pour résoudre deux problèmes de traitement du signal. Le premier est lié à la classification des sommets basée sur les informations de la couleur de la peau et le deuxième est lié au problème de détection de visage. Nous avons également montré que notre méthode peut traiter toutes les données qui peuvent être représentées par des graphiques pondérés tels que des images 2D, des surfaces 3D et des nuages de points 3D.

D'autre part, nous avons présenté aussi une nouvelle méthode pour détecter les régions saillantes sur des maillages 3D. Un descripteur de patch local a été créé pour chaque sommet où la cellule de patch est remplie avec la moyenne du facteur de déviation de chaque sommet en fonction de sommes voisines (moyenne des facteurs de déviation de sommets projetés sur le même emplacement associé à la cellule). Le patch est ensuite utilisé comme un descripteur local pour les sommets de surface des maillages 3D. De plus, une mesure de similarité entre les descripteurs de patches et une carte de hauteur a été calculée et afin de calculer le degré de similarité du degré de saillance des sommets. Enfin, nous avons affirmé la robustesse et l'efficacité de notre approche en montrant que notre approche détecte la plupart des régions saillantes visuelles dans une surface de maillage 3D. En outre, nous avons montré que notre approche peut détecter des régions saillantes sur n'importe quelle donnée pouvant être représentée par un graphe pondéré tel que le nuage de points 3D. Nous avons également présenté l'efficacité de notre méthode en présentant plusieurs applications. Parmi ces applications, nous citons les applications suivantes :

- Afin d'améliorer le processus de détection de visages présenté par la première approche, nous avons proposé une nouvelle méthode pour détecter les régions faciales 3D à partir de nuages de points colorés et d'images 2D. Cette méthode a été divisée en deux phases. Tout d'abord, les régions candidats visage ont été définies en fonction des caractéristiques de la peau, puis les caractéristiques saillantes sont calculées dans chaque région détectée pour affirmer la présence de visage humain. Nous avons également présenté un algorithme simple et efficace dérivé de l'algorithme Hill Climbing utilisant un opérateur de dilatation morphologique pour labéliser les régions saillantes de la peau. Nous avons prouvé la robustesse de notre méthode proposée à travers quelques résultats expérimentaux sur les cas d'échecs de la première approche. Ensuite, nous avons comparé notre méthode avec différentes méthodes d'état de l'art.
- En plus, nous avons présenté une nouvelle approche pour la simplification des nuages de points 3D. L'idée derrière notre approche est de détecter les régions caractéristiques qui décrivent la surface de la forme en utilisant la mesure du degré de saillance. La saillance entre sommets a été calculée en utilisant une mesure de similarité entre les patches locaux. Afin de détecter les régions caractéristiques, nous avons présenté un algorithme efficace basé sur l'étiquetage des composants connectés selon le degré de saillance des sommets. Ensuite, nous avons détecté des points caractéristiques dans ces régions. Notez que l'utilisateur peut contrôler le degré de simplification par un paramètre de résolution où le taux de simplification augmente avec la valeur de ce paramètre. Enfin, nous avons montré l'efficacité de notre méthode à travers quelques résultats expérimentaux.

- Finalement, nous avons présenté une nouvelle approche pour calculer la correspondance entre deux modèles 3D. Un nouveau descripteur de surface invariant 3D a été créé pour chaque sommet en fonction des coefficients de Zernike. De plus, un degré de saillance est calculé afin de détecter les régions caractéristiques. Ensuite, une carte de saillance multi-échelle est calculée afin d'améliorer la qualité de la saillance mesurée et de faire face au bruit. En outre, nous avons présenté un algorithme linéaire afin de segmenter la surface 3D en régions d'entités en fonction du degré de saillance, qui sont utilisées dans le processus d'appariement plutôt que de mapper des points aux points. Enfin, le processus d'appariement est modélisé comme un problème d'isomorphisme de sous-graphe et nous avons présenté un algorithme heuristique pour le résoudre. En outre, nous montrons la robustesse de notre méthode proposée à travers différents résultats expérimentaux. Enfin, nous présentons la stabilité et la robustesse de notre méthode par rapport au bruit, à l'échelle, à la translation, à la rotation, etc.

Différentes parties de notre travail ont été publiées dans des journaux internationaux spécialisés.

Journaux internationaux

Liste des articles publiés :

1. "Efficient Image Classification using Data Mining", International Journal of Combinatorial Optimization Problems and Informatics, Vol. 2, No. 1, Jan-April, 2011, pp. 27-44. ISSN: 2007-1558.
2. "3D face detection based on salient features extraction and skin color detection using data mining". The Imaging Science Journal 65 (7), 2017, pp. 393-408.
<https://doi.org/10.1080/13682199.2017.1358528>
3. "Efficient 3D point clouds classification for face detection using linear programming and data mining". The Imaging Science Journal 66 (1), 2018, pp. 23-37.
<https://doi.org/10.1080/13682199.2017.1376772>
4. "Efficient 3D mesh salient region detection using local homogeneity measure". IET Image Processing,
<http://dx.doi.org/10.1049/iet-ipr.2017.0598>
5. "An Efficient Simplification Method for Point Cloud Based on Salient Regions Detection". RAIRO - Operations Research.
<https://doi.org/10.1051/ro/2018082>

Conferences:

6. “An Invariant Multi-scale saliency detection for 3D mesh”, IEEE international conference applied and theoretical computing and communication technology (iCATccT-2018), (06-08 September 2018).

Liste des articles soumis :

7. “3D Object Matching Using Invariant Descriptors and Linear Programming”. Applied Mathematics and Computation.

Dans les prochains travaux, nous proposons comme perspective :

- Quelques méthodes basées sur nos méthodes présentées dans ce travail. Parmi ces applications, on peut citer :
 - **Classification des objets 3D** : L'idée derrière cela est de classer les objets 3D comme humains ou non par exemple. Pour faire cela, une base de données de pièces 3D humaines est construite tels que des bras et des jambes de différentes tailles. Ainsi, chaque correspondance entre un objet 3D et des parties de l'ensemble de données permet de classer l'objet comme être humain.
 - **Colorisation des nuages de points 3D numérisés** : Les historiens, les archéologues et les conservateurs de musée s'intéressent à préserver les œuvres d'art et vouloir les mettre à la disposition d'un public aussi large que possible sous forme des objets 3D colorés. Dans ce cadre, nous propose comme perspective une application de coloriage de nuages de points numérisés à partir de Scanner 3D basée sur nos méthodes de détection de régions de saillances.

Bibliography

- [1] M. S. Chen, J. Han, and P. S. Yu. "Data mining: An overview from a database perspective". IEEE Trans. Knowledge and Data Engineering, 8(6), (1996), PP. 866-883.
- [2] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. "Advances in Knowledge Discovery and Data Mining". AAAI/MIT Press, (1996).
- [3] R. Kohavi, J.R. Quinlan. "Data mining tasks and methods: Classification: decision-tree discovery". In Handbook of data mining and knowledge discovery. Oxford University Press, Inc., (2002). pp. 267-276.
- [4] J. R. Quinlan. "C4.5: Programs for Machine Learning". Morgan Kaufmann, San Mateo, CA, (1993).
- [5] W. Moudani, A.R. Sayed. "Efficient Image Classification using Data Mining". International Journal of Combinatorial Optimization Problems and Informatics, 2 (1), (2011), pp. 38-55.
- [6] M. Joenes, J. Rehg. "Statistical Color Models with Application to Skin Detection". IEEE Conference Computer Vision and Pattern Recognition, CVPR'99, (1999), pp. 274-280.
- [7] K. H. Bin Ghazali, J. Ma and R. Xiao. "An Innovative Face Detection based on Skin Color Segmentation". International Journal of Computer Applications, 34 (2), (2011), pp. 6-10.
- [8] A. Colombo, C. Cusano, and R. Schettini. "3D face detection using curvature analysis". Pattern Recognition, 39 (3), (2006), pp. 444-455.
- [9] A. Mian, M. Bennamoun, and R. Owens. "An efficient multimodal 2D-3D hybrid approach to automatic face recognition". IEEE Trans. Pattern Anal. Machine Intell, 29 (11), (2007), pp. 1927-1943.
- [10] R. Niese, A. Al-Hamadi, and B. Michaelis. "A novel method for 3D face detection and normalization". Journal of Multimedia, 2 (5), (2007), pp. 1-12.
- [11] N. Payet and S. Todorovic. "From contours to 3D object detection and pose estimation". In ICCV, (2011), pages 983-990,
- [12] H. Su, M. Sun, L. Fei-Fei, and S. Savarese. "Learning a dense multi-view representation for detection", viewpoint classification and synthesis of object categories. In ICCV, (2009), pp.213-220.
- [13] E. Kheirkhah and Z. Tabatabaie. "A Hybrid Face Detection Approach in Color Images with Complex Background". Indian Journal of Science and Technology, Vol 8(1), (2015), pp. 49-60
- [14] H.J. Lin, S.Y. Wang, and S.H. Yen. "Face Detection Based on Skin Color Segmentation and Neural Network". In IEEE, 2, (2015), pp. 1144-1149.
- [15] J. Wang, H. Yang. "Face detection based on template matching and 2DPCA algorithm". Congress on Image and Signal Processing, 4, (2008), pp. 575-579.
- [16] H. Huang, Z. Mu, H. Zeng, and B. Zhang. "3D facial point cloud preprocessing based on skin color detection using SVM". 10 th International Conference on Natural Computation (ICNC), (2014), pp. 524-528.
- [17] Lee, C. Ha, A. Varshney, and D.W. Jacobs. "Mesh saliency." ACM transactions on graphics (TOG), 24(3), (2005), pp. 659-666.

- [18] R. Song, Y. Liu, R.R. Martin, P.L. Rosin. “Mesh saliency via spectral processing”. In *ACM Transactions on Graphics (TOG)*, 33(1), (2014), PP. 6.
- [19] J. Wu, X. Shen, W. Zhu, L. Liu. “Mesh saliency with global Rarity”, *Graphical Models*, 75(5), (2013), pp. 255–264.
- [20] A. Nouri, C. Charrier, O. L  zoray. “Multi-scale mesh saliency with local adaptive patches for view point selection”, *Signal Processing: Image Communication*, 38, (2015), pp.151–166
- [21] A. Tal, E. Shtrom, G. Leifman. “Surface regions of interest for view-point selection”, In: *IEEE Conference on Computer Vision and Pattern Recognition*, (2012), pp. 414–421.
- [22] P. Tao, J. Cao, S. Li, X. Liu, and L. Liu. “Mesh saliency via ranking unsalient patches in a descriptor space,” *Computers & Graphics*, 46(1), (2015), pp. 264 – 274.
- [23] Y. Zhao, Y. Liu, R. Song, M. Zhang. “A saliency detection based method for 3d surface simplification”, In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, Kyoto, Japan*, (2012), pp.889–892.
- [24] Y. Zhao, Y. Liu. “Patch based saliency detection method for 3D surface simplification”. In: *Proceedings of the 21st International Conference on Pattern Recognition*, (2012), pp. 845–848.
- [25] Y. Zhao, Y. Liu, Z. Zeng. “Using region-based saliency for 3d interest points detection”. In: *Computer Analysis of Images and Patterns*, 2, (2013), pp.108–116.
- [26] M. Elad. “Retinex by two bilateral filters”, In: *Proceedings of the 5th International Conference on Scale Space and PDE Methods in Computer Vision*, 3459, (2005), pp. 217–229.
- [27] X. Liu, P. Tao, J. Cao, H. Chen, and C. Zou. “Mesh saliency detection via double absorbing Markov chain in feature space”, *The Visual Compute*, 32 (9), (2016), pp.1121-1132
- [28] K. H. Lee, H. Woo, T. Suk. “Point Data Reduction Using 3D Grids”, *The International Journal of Advanced Manufacturing Technology*, 18(3), pp. 201-210, 2001.
- [29] M. Pauly, M. Gross, L. P. Kobbelt. “Efficient Simplification of Point-Sampled Surfaces”, *Proceedings of the conference on Visualization’02*, IEEE Computer Society, pp. 163-170, 2002.
- [30] C. Moenning, N. A. Dodgson. “A new point cloud Simplification algorithm”, *Proceedings of third IASTED Conference on Visualization, Imaging and Image Processing*, (2003), pp. 1027-1033.
- [31] C. Moenning, N. A. Dodgson. “Intrinsic point cloud Simplification”, *Proceedings of the 14th International Conference on Computer Graphic and Vision (GraphiCon)*, Moscow, Russia, (2004).
- [32] P. F. Lee, B. S. Jong, Point-based Simplification Algorithm, *Journal WSEAS Transactions on Computer Research*, Vol. 3, No. 1, pp. 61-66, 2008.
- [33] X. Peng, W. Huang, P. Wen, X. Wu. “Simplification of Scattered Point Cloud Based on Feature Extraction”, *WGECC’09 Proceedings of the 2009 third International Conference Genetic and Evolutionary Computing*, (2009), pp. 335-338.
- [34] H. Song, H. Y. Feng. “A global clustering approach to point cloud Simplification with a specified data reduction ratio”, *Computer-Aided Design*, 40(3), (2007), pp. 281-292.

- [35] Y. Miao, R. Pajarolac, J. Feng. "Curvature-aware adaptive re-sampling for point-sampled geometry", *Computer-Aided Design*, 41(6), (2009), pp. 395-403.
- [36] B. Q. Shi, J. Liang, Q. Liu. "Adaptive simplification of point cloud using k-means clustering", *Computer-Aided Design*, 43(8), (2011), pp. 910-922.
- [37] F. R. Schmidt, D. Farin, and D. Cremers. "Fast matching of planar shapes in sub-cubic runtime". In *IEEE Int. Conf. on Computer Vision*, Rio de Janeiro, (2007), pp. 1, 3, 7.
- [38] T. Windheuser, U. Schlickewei, F. R. Schmidt, & D. Cremers. "Geometrically consistent elastic matching of 3d shapes: A linear programming solution". In *Computer Vision (ICCV)*, IEEE International Conference. (2011), pp. 2134-2141.
- [39] F. Mémoli and G. Sapiro. "A theoretical and computational framework for isometry invariant recognition of point cloud data". *Foundations of Computational Mathematics*, 5(3), (2005), PP. 313–347.
- [40] A. Bronstein, M. Bronstein, and R. Kimmel. "Efficient computation of isometry-invariant distances between surfaces". *SIAM J. Sci. Comput.*, 28(5), (2006), pp. 1812–1836.
- [41] Y. Wang, X. Gu, K. Hayashi, T. Chan, P. Thompson, and S. Yau. "Brain surface parameterization using riemann surface structure". In *MICCAI* (2), (2005), pp. 657–665.
- [42] S. Kurtsek, E. Klassen, Z. Ding, and A. Srivastava. "A novel riemannian framework for shape analysis of 3d objects". In *CVPR*, (2010), pp. 1625–1632.
- [43] Y. Zeng, C. Wang, Y. Wang, X. Gu, D. Samaras, and N. Paragios. "Dense non-rigid surface registration using high-order graph matching". In *CVPR*, (2010), pp. 382–389.
- [44] L. Torresani, V. Kolmogorov, and C. Rother. "Feature correspondence via graph matching: Models and global optimization". In *ECCV*, (2), (2008), pp. 596–609.
- [45] R. Diestel: *Graph Theory (Graduate Texts in Mathematics)*. Springer, 2005.
- [46] M. Hein, U. Von Luxburg, and M. Maier. "Influence of graph construction on graph-based clustering measures". In *Advances in Neural Information Processing Systems*, (2008), pp. 1025–1032.
- [47] J. W. Jaromsky and G. T. Toussaint. "Relative neighborhood graphs and their relatives". In *Proceedings of the IEEE*, 80, (1992).
- [48] T. Smith, E. Frank. "Statistical genomics: methods and protocols, chapter introducing machine learning concepts with WEKA". New York (NY): Springer; (2016), 353–378.
- [49] Helen dataset. <http://www.ifp.illinois.edu/~vuongle2/helen/>
- [50] A. Rajput, R.P. Aharwal, M. Dubey, et al. "J48 and JRIP rules for E-governance data". *International Journal of Computer Science and Security (IJCSS)*, 5(2), (2011), pp. 201.
- [51] E. Hjelm, B. Low. "Face detection: a survey. *Computer vision and image understanding*", 83(3), (2001), pp. 236–274.
- [52] R. Fergus, The Caltech face database. Caltech. Feb 2004. <http://www.vision.caltech.edu/html-files/archive.html>

- [53] T. Berg, A. Berg, J. Edwards, et al. «Forsyth Neural Information Processing Systems (NIPS)», 2004.
<http://tamaraberg.com/faceDataset/>
- [54] M.H. Yang, N. Ahuja. "Gaussian mixture model for human skin color and its applications in image and video databases". SPIE/EI&T Storage and Retrieval for Image and Video Databases. 3656, (1999), pp. 458–466.
- [55] R. Gal, D. Cohen-Or. "Salient geometric features for partial shape matching and similarity", ACM Trans. Graph, 2006, 25, (1), pp. 130-150
- [56] L. Jinho, M. Baback, P. Hanspeter et al. "Finding optimal Views for 3D face shape modeling", Proceeding soft he International Conference on Automatic Face and Gesture Recognition, Seoul, Korea, (2004), pp. 31–36
- [57] P. Shilane, T. Funkhouser. "Distinctive regions of 3D surfaces", ACM Trans. Graph., 2007, 26, (2), pp. 7
- [58] J. Wu, X. Shen, W. Zhu, et al. "Mesh saliency with global Rarity", Graphical Models, 2013, 75, (5), pp. 255–264
- [59] G. Lavoué. "A local roughness measure for 3D meshes and its application to visual masking", ACM Transactions on Applied perception (TAP), 2009, 5, (4), pp. 2, 4, 5, 8
- [60] J. D. Boissonnat, B. Geiger. "Three dimensional reconstruction of complex shapes based on the Delaunay triangulation", in Biomedical Image Processing and Biomedical Visualization, (PhD diss., INRIA, 1993), 1905, pp. 964–975
- [61] F. Lozes, A. Elmoataz, O. Lézoray. "Nonlocal processing of 3d colored point clouds". Pattern Recognition (ICPR), Tsukuba, Japan, (2012), pp.1968–1971
- [62] A. El Chakik, A. Elmoataz, X. Desquesnes. "Mean curvature flow on graphs for image and manifold restoration and enhancement", Signal Processing, (2014), 105, pp. 449–463
- [63] G. Gilboa, et al. "Nonlocal operators with applications to image processing", Multiscale Model. Simul, (2008), 7(3), pp. 1005–1028
- [64] X. Chen, A. Sapiro, B. Panget et al. "Schelling points on 3d surface meshes", ACM Trans. Graph, (2012), 31(4), pp.29
- [65] A. Tal, E. Shtrom, G. Leifman. "Fast-match: Fast affine template matching Surface regions of interest for view-point selection", IEEE Conference on Computer Vision and Pattern Recognition, (2012), pp. 414–421
- [66] R. Song, Y. Liu, Y. Zhao, et al. "Conditional random field-based mesh saliency", 19th IEEE International Conference on Image Processing, (2012), pp. 637–640
- [67] T. Ohashi, Z. Aghbari, & A. Makinouchi. "Hill-climbing algorithm for efficient color-based image segmentation". In IASTED International Conference on Signal Processing, Pattern Recognition, and Applications. (2003), pp.17-22.
- [68] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, et al. "The digital Michelangelo project: 3D scanning of large statues", Proceedings of ACM SIGGRAPH, (2000), pp. 131-144.
- [69] K. H. Lee, H. Woo, T. Suk. "Data reduction methods for reverse engineering". The International Journal of Advanced Manufacturing Technology. (2001), 17(10), pp. 735-743.

- [70] D. Brodsky, B. Watson. “Model Simplification through Refinement, Proceedings of International Conference on Graphics Interface”, (2000), pp. 221–228.
- [71] E. Shaffer, M. Garland. “Efficient adaptive simplification of massive meshes”, VIS’01: IEEE Transactions on Visualization’01, (2001), pp. 127-134.
- [72] H. Song and H.Y. Feng. “A progressive point cloud simplification algorithm with preserved sharp edge data”, International Journal Advanced Manufacturing Technology, (2009), 45(5), pp. 583–592.
- [73] Y. Yoshida, K. Konno, Y. Tokuyama, A Distributed Simplification Method with PC Cluster, The Journal of the Society for Art and Science,. 7(3), (2008), pp. 113-123.
- [74] C. Liao, X. Niu, M. Wang. “Simplification of 3D Point Cloud Data Based on Ray Theory”, COMPUTER MODELLING & NEW TECHNOLOGIES, (2014), 18, pp. 273-278.
- [75] S. Korman, D. Reichman, G. Tsur, and S. Avidan. “Fast-match: Fast affine template matching”, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2013), pp. 2331–2338.
- [76] <https://github.com/forsythrosin/cs348aHW4/tree/master/models/MeshsegBenchmark-1.0>
- [77] <https://en.wikipedia.org/wiki/Polytope>.
- [78] https://en.wikipedia.org/wiki/Convex_hull#Convex_hull_of_a_finite_point_set
- [79] J. Yang, D. Zhang, A. Frangi, J. Yang. “Two-Dimensional PCA: A new approach to appearance-based face representation and recognition”, In IEEE transactions on pattern analysis and machine intelligence, 26, (Januray 2004).
- [80] V. Lakshminarayanan et al. “Zernike polynomials: a guide”, Journal of Modern Optics, vol 58(7), PP. 545–561.
- [81] www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/