ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

თეონა სამხარაძე

სითბური ასახვა: სამგანზომილებიანი ობიექტის სითბური სეგმენტაცია.

(სითბური ცენტრების რაოდენობის პოვნა და იდენტიფიცირება)

კომპიუტერული მეცნიერებები

სამაგისტრო ნაშრომი შესრულებულია კომპიუტერული მეცნიერებების მაგისტრის ხარისხის მოსაპოვებლად

ალექსანდრე გამყრელიძე-

ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტის

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტის

კომპიუტერული მეცნიერებების მიმართულების

სრული პროფესორი

გოდერძი ფრუიძე-

ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტის

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტის

მოწვეული ლექტორი,

კიბერნეტიკის ინსტიტუტის მათემატიკური კიბერნეტიკის განყოფილების

მეცნიერ თანამშრომელი

თბილისი, 2013

სარჩევი

ანოტაცია

ნაშრომის მიზანია შეიქმნას პროგრამული უზრუნველყოფა, რომელიც მოახდენს სამგანზომილებიანი ობიექტის სეგმენტაციას. **სამგანზომილებიანი ობიექტის სეგმენტაცია**(შემდგომში **სოს**) გულისხმობს ერთი მთლიანი ობიექტის დაყოფას უფრო მცირე ზომის მნიშვნელოვან კომპონენტებად. **სოს** არის ფუნდამენტური ამოცანა, რომლის იმპლემენტაციაც მოითხოვს დაბალი დონის პროგრამირებისა და კომპიუტერული გეომეტრიის საფუძვლების ცოდნას. ამ ამოცანის შედეგები და აპლიკაციები გამოიყენება მრავალ სხვადასხვა დარგში, როგორებიცაა მაგალითად: კომპიუტერული ხედვა, კომპიტერულად მათვადი დიზაინი, ბიოინფორმატიკა, სამგანზომილებიანი სამედიცინო დასურათება. დღესდღეისობით არსებობს **სოს**-ის რამდენიმე მიდგომა, თუმცა მათი უმრავლესობა მხოლოდ თეორიული ნაშრომია და არ არის ფართოდ დანერგილი პრაქტიკაში. ამ ნაშრომში გავაანალიზებთ **ობიექტის სითბური სეგმენტაციის** მეთოდს და მიღებულ ცოდნას მივცემთ პრაქტიკულ გამოყენებას. უფრო კონკრეტულად კი მოვახდენთ **სითბური ცენტრების რაოდენობის დათვლის(Estimating number of the heat centers) და მათი იდენტიფიცირების(Hunting heat centers)** ალგორითმის იმპლემენტაციას.

Abstract

The main goal of this work is to create 3D object segmentation software. **3D object segmentation(3DOS)** is the process which divides one whole 3D object to smaller meaningful components. **3DOS** is fundamental task, the implementation of which requires knowledge of computational geometry and low-level programming. The results and applications of **3DOS** are used in areas as diverse as computer vision, computer-aided design, bioinformatics, 3D medical imaging. Nowadays there are several **3DOS** methods, but most of them are just theoretical works and they don't have wide distribution in practice. In this paper we analyze **3D object heat segmentation** and apply acquired knowledge to practice. More specifically we implement the **estimation of heat centers** and **hunting heat centers** algorithms.

1. შესავალი

1. 1 საფუძვლები

ბოლო წლების განმავლობაში სწრაფად გაიზარდა სამგანზომილებიანი მოდელების გამოყენება სხვადასხვა პროფესიულ დარგებში, რასაც მოჰყვა სამგანზომილებიანი მოდელების ათვისების ტექნოლოგიების განვითარება. შესაბამისად, წარმოიშვა მოდელების ავტომატური დამუშავებისა და გაანალიზების მუდმივად ზრდადი მოთხოვნა. პირველი მნიშვნელოვანი ნაბიჯი მოდელის დამუშავებისა, არის მისი სეგმენტაცია უფრო მცირე ზომის კომპონენტებად, რაც საკმაოდ რთული ამოცანაა. სეგმენტაციის მიდგომას წინ უდგას შემდეგი პრობლემები

- ზედაპირის იზომეტრიული გარდაქმნებისადმი მგრძნობელობა.
- ზედაპირის ტოპოლოგიური დარღვევებისადმი მგრძნობელობა.
- დასამუშავებელ მოდელში თანდაყოლილი ხარვეზებისადმი მგრძნობელობა.
- სეგმენტაციის შედეგის ადამიანის აღქმასთან შეუთავსებლობა.

ჩვენ განვსაზღვრავთ სეგმენტაციის სქემას, რომელიც გაითვალისწინებს და გადაჭრის ზემოთხსენებულ პრობლემებს. ამ სქემას ეწოდება **ადამიანის ვიზუალურ აღქმასთან შესაბამისობაში მყოფი სეგმენტაცია(perceptually consistent mesh segmentation (PCMS))(განსაზღვრება 1.1.1)**

განსაზღვრება 1.1.1 ადამიანის ვიზუალურ აღქმასთან შესაბამისობაში მყოფი სეგმენტაცია(perceptually consistent mesh segmentation (PCMS)): როდესაც ადამიანი ეცნობა ნებისმიერ უცხო ობიექტსა თუ გაურკვეველ ფორმას, ის ცდილობს მის დაკავშირებას უკვე ნაცნობ სამგანზომილებიან სხეულებთან, ქვეცნობიერად ქმნის ფორმების გარკვეულ შაბლონებს და ამის მიხედვით ცდილობს, რომ დაადგინოს ობიექტის წარმომავლობა. ადმიანის გონება იგივე პრინციპით ახდენს ობიექტების დაყოფას მნიშვნელოვან კომპონენტებად და სწორედ ეს იგულისხმება ჩვენს მიერ მოყვანილი სქემის **ადამიანის ვიზუალურ აღქმასთან შესაბამისობაში.**

PCMS უფრო მეტ ინფორმაციას გვაძლევს მოდელის თვისებების შესახებ, ამიტომ **PCMS-ში** ინტეგრირებული მახასიათებლები უფრო მნიშვნელოვანი გახდა მოდელების დამუშავებისა და ანალიზის დროს. ეს მეთოდი აადვილებს სამგანზომილებიანი

მოდელის ინტერპრეტაციას, ვინაიდან ის წარმოაჩენს ზედაპირის თავდაპირველად დაფარულ გეომეტრიულ სტრუქტურას. ეს შეიძლება გამოისახოს წმინდა გეომეტრიული ტერმინებში, სემანტიკური ინფორმაციის სახით ან ორივეთი ერთად. შედეგად, მიღებულ გეომეტრიულ სტრუქტურას გააჩნია საკმარისი ინფორმაცია იმისათვის, რომ დაყოს ობიექტები ფუნქციონალურ კომპონენტებად. ეს პროცესი მიმდინარეობს ადამინის აღქმასთან მიახლოებული გზით, ამიტომ სეგმენტაციის სხვა მიდგომებთან შედარებით **PCMS**-ი არის უფრო ადეკვატური მოდელის თვისებების გაანალიზებისას, რაც გამოყენებას პოულობს სხვადასხვა სამოდელო აპლიკაციებში.

ამის მაგალითებია:

- ორი ან მეტი სამგანზომილებიანი ობიექტის ან მათი ნაწილების შესაბამისობის დადგენა.
- **მოდელიდან ჩონჩხის ამოღება (განსაზღვრება 1.1.2).**
- ტექსტურის ასახვა მოდელზე.
- მოდელის გამარტივება.

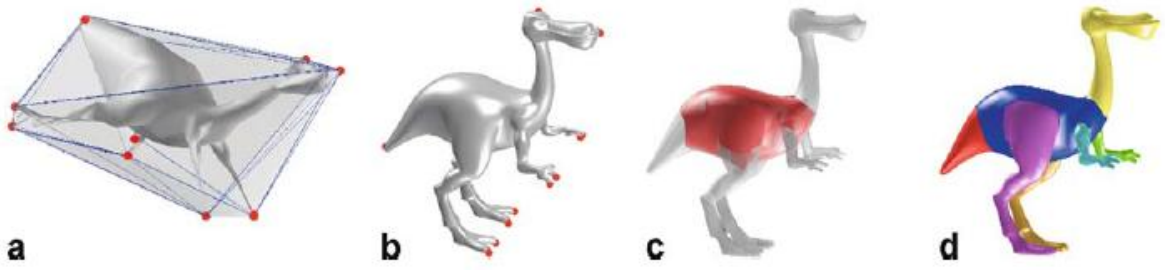
განსაზღვრება 1.1.2 მოდელიდან ჩონჩხის ამოღება. სამგანზომილებიანი ობიექტის ჩონჩხი ვუწოდოთ იგივე დანიშნულებისა და თვისებების მქონე სტრუქტურას როგორიცაა ადამიანის ჩონჩხი. მოდელის ჩონჩხი შეიცავს ძვლებს და სახსრებს და გამოყენება კომპიუტერულ ანიმაციაში მოდელის ანიმირებისათვის.

როგორც წესი, მოდელის ჩონჩხის აგება ხდება ანიმატორების მიერ, ხოლო მოდელიდან ჩონჩხის ამოღებაში ჩვენ ვიგულისხმებთ ჩონჩხის გენერირების ავტომატურ პროცესს, წინასწარ შექმნილი პროგრამული უზრუნველყოფის საშუალებით.

1.2 სეგმენტაციის სხვადასხვა მიდგომები

არსებობს მოდელის სეგმენტაციის ორი განსხვავებული მიდგომა: **არა-გაბნევადი(non-diffusion)** და **გაბნევაზე დაფუძნებული(diffusion-based)** სეგმენტაციები.

არა-გაბნევადი სეგმენტაციის ერთ-ერთი მაგალითია **სეგმენტაცია მახასიათებელი წერტილებისა და ბირთვის განსაზღვრის გზით(mesh segmentation using feature points and core extraction).** ეს მეთოდი მდგრადია მოდელის პოზების ცვალებადობის მიმართ, მაგრამ დამაკმაყოფილებელი სეგმენტაციის მიღება რთულია, რადგან მახასიათებელი წერტილები მგძნობიარენი არიან ზედაპირის ტოპოლოგიური დარღვევებისა და მაღალი ხარვეზიანობის მიმართ.



სურ. 1.2.1 ალგორითმის მონახაზი. a) მრავალ განზომილებიანი მაშტაბირება(multi-dimensional scaling(MDS)) და მისი ამოზნექილი გარსი b) მახასიათებელი წერტილები c) ბირთვი d) სეგმენტაცია

განსაზღვრება 1.2 Multi-dimensional scaling(MDS) - MDS არის ალგორითმი, რომელიც მოცემულ ბადეს გარდაქმნის პოზების მიმართ ინვარიანტულ ახალ ბადეში. საზოგადოდ, ახალი ბადის ჩადგმა შეიძლება მოხდეს ნებისმიერ m განზომილებიან ევკლიდურ სივრცეში, მაგრამ ჩვენ მაინც ვიყენებთ სამგანზომილებიან სივრცეში ასახვას. რაც უფრო განსხვავებულია ობიექტის ნაწილები ერთმანეთისაგან, მით უფრო მეტადაა ერთმანეთისაგან დაშორებული ეს ნაწილები MDS ალგორითმით მიღებულ ბადეში. ამ ალგორითმით მიღებული ბადე უფრო მოსახერხებელია ობიექტის სეგმენტაციის დროს.

გაბნევაზე დაფუძნებული მეთოდები უფრო თანამედროვეა და ორიენტირებულია PCMS-ზე. ეს მოთოდები მოდელის სტრუქტურის გამოსავლენად იყენებენ გაბნევის საზომებს. ერთ-ერთი ასეთი მაგალითია **ლაპლას-ბელტრამის დიფერენციაური ოპერატორის საკუთრივი მნიშვნელობებისა (Eigenvalues) და საკუთრივი ფუნქციების (eigenfunctions) გამოყენება ობიექტის კომპონენტებად დასაყოფად ისე, რომ შედეგი იყოს მდგრადი ობიექტის დეფორმაციების მიმართ (Laplace-Beltrami Eigenfunctions for Deformation Invariant Shape Representation)**. აქვე წარმოდგენილია GPS ხელწერა, რომელიც

შემდეგ შემოწმდა დეფორმირებულ ზედაპირებზე და აჩვენა კარგი შედეგები.

ლაპლასის ოპერატორი

მათემატიკაში **ლაპლასის ოპერატორი,** იგივე **ლაპლასიანი,** არის დიფერენციალური ოპერატორი, რომელიც მოქმედებს ევკლიდურ სივრცეში ან მის ღია არეებზე განმარტებულ ფუნქციებზე და განისაზღვრება როგორც გრადიენტის დივერგენცია. **ლაპლასიანი** აღინიშნება $\nabla \cdot \nabla$, ∇^2, Δ სიმბოლოებით. ჩვენ ვიყენებთ **დისკრეტულ ლაპლასის ოპერატორს,** რომელიც **ლაპლასის ოპერატორის** ანალოგიურია და განიმარტება დისკრეტულ სტრუქტურებზე, როგორიცაა გრაფები და დისკრეტული ბადეები. ზოგადად, **ლაპლასის ოპერატორი** თავს იჩენს დიფერენციალურ გამოსახულებებში, რომლებიც აღწერენ სხვადასხვა ფიზიკურ ფენომენებს, მაგალითად: ელექტრონულ და გრავიტაციულ პოტენციალებს, სითბურ და სითხის ნაკადის გაბნევას, ტალღების გავრცელებასა და კვანტურ მექანიკას.

განსაზღვრება 1.2.1 ლაპლას-ბელტრამის ოპერატორი. დიფერენციალურ გეომეტრიაში **ლაპლასის ოპერატორის** განზოგადება გამოიყენება ევკლიდეს სივრცეში არსებულ ზედაპირებზე განსაზღვრულ ფუნქციებზე ოპერირებისას, უფრო ზოგადად კი **რიმანის (Riemannian)** და **ფსევდო-რიმანის (pseudo-Riemannian)** მრავალსახეობებზე. ამ განზოგადებას ეწოდება **ლაპლას-ბელტრამის ოპერატორი.** ლაპლასის ოპერატორის მსგავსად, **ლაპლას-ბელტრამის ოპერატორიც** განისაზღვრება როგორც გრადიენტის დივერგენცია და არის წრფივი ოპერატორი, რომელიც ფუნქციას ასახავს ფუნქციაში.

განსაზღვრება 1.2.2 საკუთრივი მნიშვნელობა, საკუთრივი ვექტორი, საკუთრივი ფუნქცია. ვთვათ, მოცემული გვაქვს წრფივი ოპერატორი წრფივ სივრცეში, ან შესაბამისი კვადრატული მატრიცა A, მაშინ მისი **საკუთრივი ვექტორი** განიმარტება, როგორც ისეთი არანულოვანი ვექტორი v, რომლისთვისაც $Av = \lambda v$ რაიმე λ სკალარისათვის. ამ შემთხვევაში λ-ს ეწოდება A მატრიცის v ვექტორის შესაბამისი **საკუთრივი რიცხვი** ან **საკუთრივი მნიშვნელობა.** ხოლო თუ ოპერატორი მოქმედებს ფუნქციათა სივრცეზე, მაშინ

საკუთრივი ვექტორების ნაცვლად გვექნება საკუთრივი ფუნქციები.

ვთქვათ, $\mathcal{D}$ არის რაიმე დიფერენციალური ოპერატორი, რომელიც განმარტებულია უსასრულოდ დიფერენცირებად, ნამდვილ მნიშვნელობიან ფუნქციათა C^{∞} სივრცეზე.

საკუთრივი ფუნქცია ეწოდება ისეთ არანულოვან f ფუნქციას, რომელიც აკმაყოფილებს შემდეგ პირობას:

$$\mathcal{D}f = \lambda f.$$

1.3 მახასიათებელი წერტილები და მათი იდენტიფიცირება

ობიექტის სეგმენტაციის პროცესი მახასიათებელ წერტილებისა და ბირთვის ამოღების გამოყენებით(mesh segmentation using feature points and core extraction) , ისევე როგორც **სითბური სეგმენტაცია(განსაზღვრება 1.3.1)**, მიიღება ორ ეტაპად.

> პირველი ეტაპი: სამგანზომილებიან ზედაპირზე სეგმენტაციისთვის შესაფერისი წერტილების რაოდენობის პოვნა და მათი იდენტიფიცირება.
> მეორე ეტაპი: მიღებული ინფორმაციისა და სეგმენტაციის ალგორითმების გამოყენებით მოდელის კომპონენტებად დაყოფა.

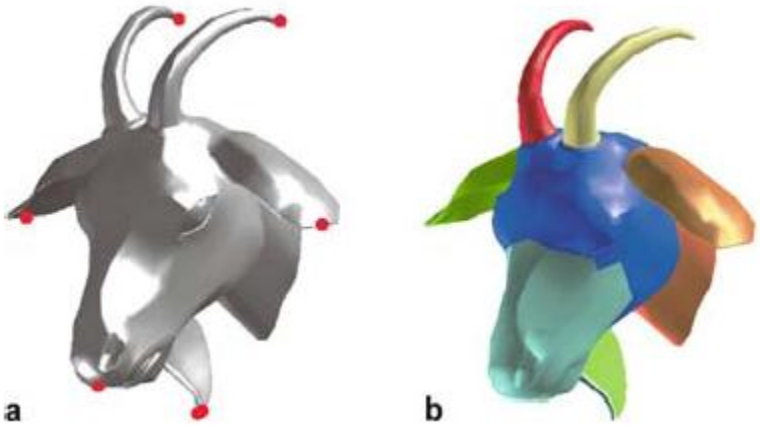
ჩვენი სამაგისტრო ნაშრომის ფარგლებში ჩვენ განვიხილავთ ამ პროცესის პირველ ფაზას და მოვახდენთ სითბური წერტილების რაოდენობის განსაზღვრასა და მათ იდენტიფიკაციას.

განსაზღვრება 1.3.1 სამგანზომილებიანი ობიექტის სითბური სეგმენტაცია იყენებს სამგანზომილებიანი ობიექტის ზედაპირის თბოგამტარობის თვისებებს და შესაბამისი ანალიზის გამოყენებით ახდენს ობიექტის მნიშვნელოვანი და ფუნქციონალური კომპონენტების გამოყოფას.

განვიხილოთ ალგორითმი რომელიც პოულობს **მახასიათებელ წერტილებს (განსაზღვრება 1.3.2)**. ის არ საჭიროებს რაიმე წინასწარ ცოდნას მახასიათბელი წერტილების რაოდენობის შესახებ, ან მომხმარებლის მიერ წინასწარ მიწოდებულ რაიმე პარამეტრს.

ინტუიცია გვკარნახობს, რომ მახასიათბელი წერტილები უნდა იყვნენ განლაგებული

მოდელის თვალსაჩინო კომპონენტების წვეროებში,როგორც ეს შემდეგ სურათზეა მოცემული.



სურ. 1.3. მახასიათებელი წერტილები და შესაბამისი სეგმენტაცია

როგორც ვხედავთ, მახასიათებელი წერტილები განლაგებულია რქების, ყურების, პირის და წვერის წვეროებზე. უფრო მეტიც, მახასიათებელი წერტილები უნდა იყვნენ მოდელის პოზებისგან დამოუკიდებელნი.

მახასიათებელი წერტილები გამოიყენება სხვადასხვა აპლიკაციებში, როგორებიცაა: მეტამორფოზა, დეფორმაციის გადატანა, ბადის ამოღება, ჯვარედინი პარამერიზაცია, ტექსტურის ასახვა, სეგმენტაცია.

ვთქვათ მოცემულია S ბადე. ამ ბადის ყოველი v წვეროსათვის განვსაზღვროთ N_v, როგორც v წვეროს ყველა მეზობელი წვეროს სიმრავლე. ვთქვათ $GeodDist(v_i, v_j)$ არის S ბადის v_i და v_j წვეროებს შორის **გეოდეზური მანძილი(Geodesic distance)**. ერთ-ერთი პირობა, რომელიც აუცილებლად უნდა დააკმაყოფილოს მახასიათებელმა v წვერომ მდგომარეობს შემდეგში: ამ v წვეროს ყოველი მეზობელი v_n წვეროსათვის უნდა სრულდებოდეს შემდეგი უტოლობა:

$$\sum_{v_i \in S} GeodDist(v, v_i) > \sum_{v_i \in S} GeodDist(v_n, v_i)$$

ფორმულა 1.3

სხვა სიტყვებით რომ ვთქვათ, წერტილი მდებარეობს კომპონენტის წვეროზე თუ ის არის გეოდეზური მანძილების ჯამის ლოკალური მაქსიმუმი.

უნდა აღვნიშნოთ, რომ ზემოთ მოცემული პირობა ლოკალური ხასიათისაა, ვინაიდან წვეროს ვადარებთ მხოლოდ მის მეზობელ წვეროებს, ამიტომ ამ პირობით მიღებული წერტილების სიმრავლე მოიცავს არამარტო ყველა მახასიათებელ წერტილს, დამატებით წერტილებსაც, რომლებიც არ არიან მახასიათებელი წერტილები. ეს გამოწვეულია მცირე ხარვეზებით. დამატებითი წერტილების გაფილტვრა ხდება იმ დამატებითი შეზღუდვით, რომ მახასიათებელი წერტილები უნდა მდებარეობდეს MDS-ალგორითმის შედეგად მიღებული ბადის ამოზნექილი გარსის წვეროებში. ეს ბუნებრივად ჩანს, ვინაიდან MDS ალგორითმი უფრო გაშლის საწყისი სამგანზომილებიანი მოდელის ცალკეულ კომპონენტებს.

განსაზღვრება 1.3.2 მახასიათებელი წერტილი. ბადის წვერო არის მახასიათებელი წერტილი, თუ ის აკმაყოფილებს ფორმულა 1.3-ს და მდებარეობს MDS ალგორითმით მიღებული ბადის ამოზნექილ გარსზე.

2. სითბური ცენტრების რაოდენობის დათვლა და მათი იდენტიფიცირება

2.1 საშუალო სითბური ანაბეჭდი

განვსაზღვროთ საშუალო სითბური ანაბეჭდი (heat mean signature(HMS)), რომელიც რაოდენობრივად ითვლის სითბოს დინების პროცესის შედეგად მიღებულ ტემპერატურის განაწილებას. HMS-ს აქვს შემდეგი ფორმურილება:

$$HMS(i) = \frac{1}{N} \sum_{j, j \neq i} H_t(i, j)$$

ფორმულა 2.1

სადაც i და j აღნიშნავს ზედაპირის i-ურ და j-ურ წვეროებს, N არის წვეროების მთლიანი რაოდენობა, ხოლო $H_t(i, j)$ i წვეროდან j წვერომდე t დროში გავრცელებული სითბო.

$HMS(i)$ შეგვიძლია ფიზიკურად აღვწეროთ, როგორც ზედაპირის საშუალო ტემპერატურა, რომელიც მიიღება i-ურ წვეროზე ერთეულოვანი სითბოს გადაცემის შემდეგ რაღაც დროის ინტერვალში სითბური გაბნევის შედეგად. HMS-ს აქვს სასურველი თვისებები, რაც საჭიროა PCMS-თვის: ის დამოუკიდებელია მოდელის ოზომეტრიული გარდაქმნების და ტოპოლოგიური ხარვეზების მიმართ, სწორად ასავახს ტემპერატურის განაწილებას სითბოს გავრცელების დროს.

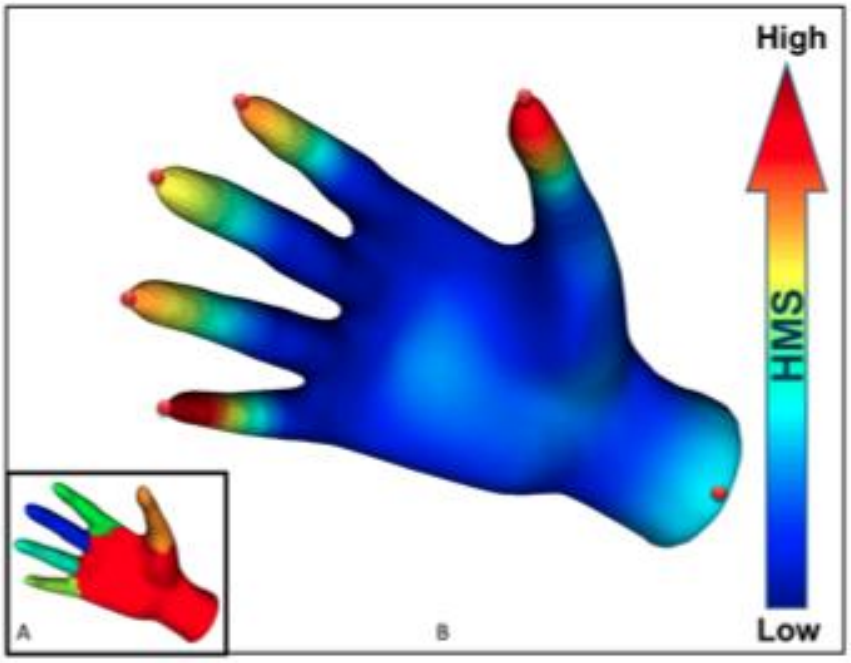
2.2 სითბური ცენტრი და სითბოს გამტარი

სითბური ცენტრი თითოეული სეგმენტისათვის განისაზღვრება, როგორც წვერო HMS-ის ყველაზე დიდი მნიშვნელობით ამ სეგმენტის წვეროებს შორის. სეგმენტის დანარჩენი წეერტილები განისაზღვრება როგორც სითბოს გამტარები მოცემული სითბური ცენტრისათვის. სითბური ცენტრი განისაზღვრება შემდეგნაირად:

$$H_c = \underset{v \in v_c}{argmax}\{HMS(V)\}$$

ფორმულა 2.2

სადაც H_c განსაზღვრავს C სეგმენტის სითბურ ცენტრს და V_c განსაზღვრავს წვეროების სიმრავლეს C-ზე. $HMS(V)$ განსაზღვრავს საშუალო სითბურ ანაბეჭდს V წვეროსათვის სურათი 2.2(B) ასახავს საშუალო სითბურ ანაბეჭდს, სითბურ ცენტრებს და სითბოს გამტარებს ადამიანის ხელის მოდელისათვის. HMS დაიტანება ზედაპირზე როგორც ფერების რუქა: რაც უფრო მაღალია HMS-ის მნიშვნელობა, მით უფრო ახლოა ფერები წითელთან. ხელის მოდელი დაყოფილია ექვს ფუნქციონალურ კომპონენტად (სურათი 2.2(A)). თითოული თითი წარმოადგენს ერთ სეგმენტს და ხელის მტევანი გამოყოფილია როგორც ცალკე სეგმენტი. აღმოჩენილი სითბური ცენტრები მონიშნულია წითელი სფეროებით. ადვილი შესამჩნევია, რომ სითბური ცენტრები განლაგებულია თავიანთი სეგმენტების HMS მნიშვნელობის ყველაზე მაღალი მაჩვენებლის მქონე უბნებში.



სურათი 2.2

2.3 სითბური ცენტრების რაოდენობის დათვლა

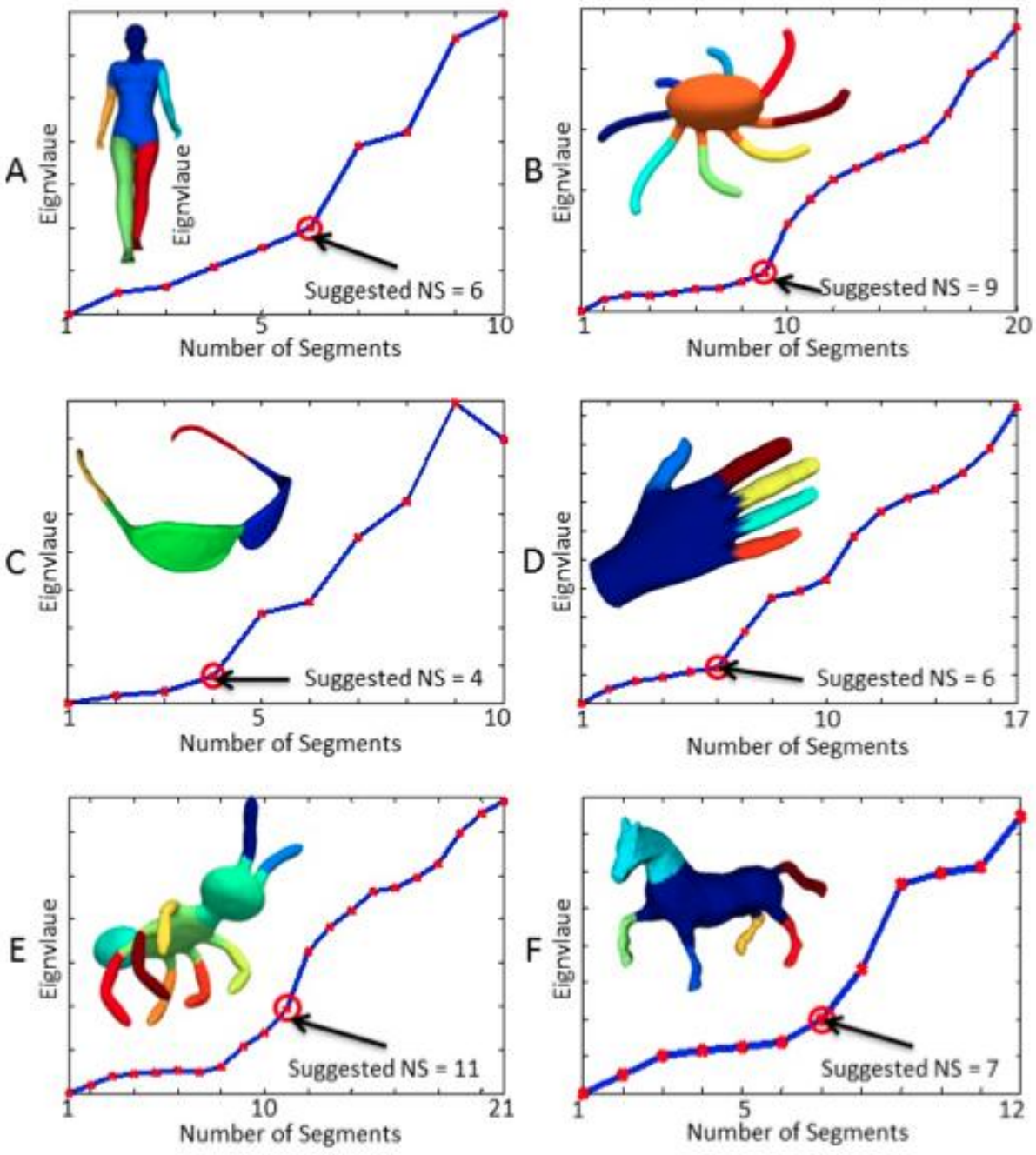
სითბური ცენტრების რაოდენობა ექვივალენტურია სეგმენტების რაოდენობისა. გამოთვლებისათვის ჩვენ აღვწერთ მარტივ მიდგომას, რომელიც ეფუძნება **ლაპლასის სპექტრის** ქცევის ანალიზს.

ვთქვათ $0 = \lambda_1 < \lambda_2 < \cdots < \lambda_n$ არის ლაპლასის სპექტრი მოცემული მოდელისთვის, სადაც λ_i არის i-ური საკუთრივი მნიშვნელობა და $Dif(i,j) = \lambda_j - \lambda_i$

მაშინ ჩვენ გვაქვს შემდეგი დაკვირვება: ციფრი i დაემთხვევა ბუნებრივი სეგმენტაციის კომპონენტების რაოდენობას, მაშინ როდესაც აღინიშნება პირველი მკვეთრი ცვლილება $Dif(i, i+1)$ i და $i+1$ ელემენტებს შორის. ამის დასამტკიცებლად მოგვყავს შემდეგი 6 სეგმენტაციის ექსპერიმენტი. განვსაზღვრავთ **საკუთრივ წირს(eigen-curve)**, რომელიც შეუსაბამებს ერთმანეთს საკუთრივ მნიშვნელობებს და სეგმენტების რაოდენობას. სურათი 2.3 ასახავს 6 საკუთრივ წირს სხვადასხვა მოდელების სეგმენტაციებისათვის. გამოყენებული მოდელებია:

(A)ქალი, (B)რვაფეხა, (C)სათვალეები, (D)ხელის მტევანი, (E)ჭიანჭველა, (F)ცხენი.

სეგმენტების სავარაუდო რაოდენობა მინიშნებულია ისრით. სეგმენტების სავარაუდო რაოდენობებია: 6, 9, 4, 11 და 7 შესაბამისად.



სურ 2.3 NS გვიჩვენებს სეგმენტების სავარაუდო რაოდენობას, რაც ემთხვევა ადამიანის აღმით მიღებულ სეგმენტების ბუნებრივ რაოდენობას.

2.4 სითბური ბირთვი

სითბური ბირთვი გამოითვლის სითბოს ნაკადს M მრავალსახეობაზე. ეს მნიშვნელობა არის უნიკალური თითოეული i და j წვეროებისათვის M-დან. ვთქვათ i კვანძს მივანიჭეთ ერთეული სითბოს რაოდენობა და სითბოს მივეცით გავრცელების საშუალება მრავალსახეობის ყველა წიბოს გასწვრივ. სიჩქარე, რომლითაც სითბო გაიბნევა მრავალსახეობის გასწვრივ არის წინასწარ განსაღვრული წონები წიბოებს შორის. სითბური ბირთვის მნიშვნელობა $H_t(i,j)$ არის j წერტილში დაგროვილი სითბო t დროის გასვლის შემდეგ. ის კრებს სითბოს დინებას ყველა შესაძლო გზის გათვალისწინებით ზედაპირის ორ წერტილს შორის, ამიტომ სითბური ბირთვი ხედავს და ინახავს ზედაპირის მიღმა მდგარ სტრუქტურას. თუ წვეროს აქვს რამდენიმე გზა სხვა წვეროებისკენ, მაშინ სითბო უფრო სწრაფად მიედინება და გროვდება მრავალსახეობის გასწვრივ. შესაბამისად საშუალო ტემპერატურა იზრდება როდესაც სითბოს ვანიჭებთ ისეთ წვეროებს, რომლებსაც გააჩნიათ სხვა წვეროებამდე შემაერთებლი მეტი გზა. სითბური ბირთვის მიახლოებას ვადგენთ საკუთრივი ფუნქციების გავრცელების საშუალებით. სამგანზომილებიანი მოდელი წარმოდგენილია როგორც გრაფი $G = (V, E, W)$, სადაც V არის წვეროების სიმრავლე, E არის წიბოების სიმრავლე და W არის თითოეული წიბოს შეწონილი მნიშვნელობა.

გრაფით განმარტებული ლაპლასის ოპერატორი(გრაფის ლაპლასიანი) განსაზღვრავს სითბოს გავრცელების ინტენსივობას შეწონილი გრაფის გასწვრივ, რომელიც დამოკიდებულია დროზე. თბოგამტარობის განტოლების ამონახსნის საპოვნელად ვიყენებთ ლაპლასის ოპერატორის მახასიათებელ ფუნქციებს და მაჩვენებლიან ფუნქციებს, სადაც არგუმენტად მონაწილეობს ლაპლასის ოპერატორის მახასიათებელი რიცხვები და დრო.

გრაფის ლაპლასიანი განისაზღვრება შემდეგნაირად:

$$L = D - W$$

ფორმულა 2.4.1

სადაც D არის ხარისხის დიაგონალური მატრიცა და მის დიაგონალზე მყოფი მნიშვნელობები წარმოადგენს W-ს შესაბამისი სტრიქონების ჯამებს. ნორმალიზებული ლაპლასიანი განისაზღვრება შემდეგნაირად:

$$L = D^{-1/2} L D^{-1/2}$$

ფორმულა 2.4.2

სითბური ბირთვი განისაზღვრება შემდეგნაირად:

$$H_t(i,j) = \sum_{k=1}^{|V|} \exp(-\lambda_k t)\, \phi_k(i) \phi_k(j)$$

ფორმულა 2.4.3

სადაც λ_k არის k-ური საკუთრივი მნიშვნელობა და ϕ_k არის k-ური საკუთვირივი ფუნქცია. $H_t(i,j)$ განისაზღვრება როგორც სითბური მსგავსება $Haf(i,j)$ ორ წერტილს შორის, რომელიც თვის მხრივ არის გავრცელებული სითბოს ოდენობა i-დან j-მდე t დროის შემდეგ.

დამატებით ჩვენ განვსაზღვრავთ სითბურ კოორდინატს, იმისათვის, რომ მოდელის წვეროები მოვაქციოთ ევკლიდურ სივრცეში.

$$Cor(i) = (e^{-\lambda_1 t/2}\phi_1(i), e^{-\lambda_2 t/2}\phi_2(i), \ldots, e^{-\lambda_k t/2}\phi_k(i))$$

ფორმულა 2.4.4

სადაც λ_k არის k-ური საკუთრივი მნიშვნელობა, t განსაზღვრავს დროს, რომლის

განმავლობაშიც სითბო გავრცელდება მრავალსახეობაზე, ხოლო ϕ_k არის k-ური საკუთვირივი ფუნქცია. სითბური კოორდინატი განისაზღვრება n განზომილებიან სივრცეზე.

2.5 სითბური ცენტრების ძებნა

სითბური ბირთვი ზომავს სითბურ მსგავსებას ზედაპირის ნებისმიერ ორ წვეროს შორის. ნებისმიერ წვეროზე სითბოს მინიჭებით სითბური მსგავსება ასახავს სითბოს დინებას ზედაპირის გასწვრივ. რადგან სითბოს გაბნევის ეფექტურობა დამოკიდებულია წვეროებზე, სითბოს ეფექტური დინებისათვის უნდა ვიპოვოთ გლობალურად ოპტიმალური წვეროები(სითბური ცენტრები) .

სითბური ცენტრების საძიებო გლობალური ალგორითმი შედგება 4 ძირითადი ნაწილისაგან.

1. V არის N ცალი წვეროს სიმრავლე M ბადეზე. უნდა ვიპოვოთ H_cსითბური ცენტრების C რაოდენობა M ზედაპირზე. S_c არის წვეროების სიმრავლე, რომელიც შეიცავს სითბურ ცენტრს H_c-ს და მის შესაბამის სითბურ გამტარებსაც.
2. S_c -თვის განვსაზღვრავთ დიამეტრს D_c , რომელიც არის სითბური მსგავსების უდიდესი მნიშვნელობა იმ სითბური გამტარებიდან S_c -ში, რომელთა სითბური მსავსების შედარება H_c -თან აღემატება D_c /2-ს.
3. დიამეტრი D_m, M-ის ყველა წვეროსათვის განისაზღვრება ფორმულით:

$$D_M = \min_{c=1\ldots C} D_c = \min_{c=1..C} \min_{V_i \in S_c} (H_{af}(V_i, H_c))$$

ფორმულა 2.5.1

სადაც C არის სითბური ცენტრების რაოდენობა M-ზე, H_c არის სითბური ცენტრი, v_i არის i-ური წვერო, რომელიც არის სითბური გამტარი მისი

საკუთარი სითბური ცენტრისთვის, Haf(V_i, H_e) განსაზღვრავს სითბურ მსგავსებას V_i-სა და H_e-ს შორის.

4. გადავცემთ რა სითბური ცენტრების რაოდენობა C-ს, ჩვენ ვახდენთ Dm-ის მაქსიმიზაციას, რაც იძლევა წვეროებს შორის სითბური მსგავსების მაქსიმიზაციის გარანტიას. სითბური ცენტრის ძებნის პრობლემის ფორმულირება შეიძლება შემდეგნაირად:

$$H_c = {}^{argmax}_{H_c} D_M$$

ფორმულა 2.5.2

სადაც H_C განსაზღვრავს სითბურ ცენტრს, D_M არის H_C-ს ფუნქცია(ნახეთ ფორმულა 2.5.1) და თითორული კლასტერი H_C ექვემდებარება მის განსაზღვრებას ფორმულაში 2.2.

სითბური ცენტრების ძებნის ალგორითმის ფსევდოკოდი:

1. ალგორმითმი მიიღებს შემდეგ პარამეტრებს: HMS-ს თითოეული წვეროებისათვის M-ზე, სითბური ცენტრების რაოდენობა C და წვეროების საერთო რაოდენობა N. V_j განსაზღვრავს j-ურ წვეროს.
2. ავირჩიოთ წვერო
   V1 = argmaxHMS(V), V ეკუთვნის M-ს.
3. G არის სითბური ცენტრების სიმრავლე.
4. $g_1 = V_1, G = G \cup g_1$

5. for j = 1 დან N-მდე do
6. -->$HeatAff$ არის მასივი, რომელშიც ინახება ყველა წვეროს სითბური მსგავსება თავიანთ სითბურ ცენტრებთან.

$$HeatAff(V_j) = H_{af}(V_j, g_1)$$

7. -->S ინახავს წვეროების ნიშნულებს.
8. $S(V_j) = 1$
9. end for

10. $V_i = {}^{argminHeatAff}_{V \in M}(V)$
11. for i = 2-დან C-მდე do
12. -->ხდება ახალი სითბური ცენტრის ინიხიალიზაცია იმ წვეროთი, რომელსაც აქვს ყველაზე ნაკლები სითბური მსგავსება მიმდინარე სითბურ ცენტრებთან.

$$g_i = V_i, G = G \cup g_i$$

13. for j = 1-დან N-მდე do
14. -->თუ წვერო ახლოსაა ახლად შექმნილ სითბურ ცენტრთან

$if\ H_{af}(V_j, g_i) \geq HeatAff(V_j)$

15. -->განაახლე $HeatAff$ მასივი
16. $HeatAff(V_j) = H_{af}(V_j, g_i)$

17. -->განაახლე წვეროების ნიშნულები.

$$S(V_j) = i$$

18. -->იპოვე წვერო, რომელსაც აქვს ყველაზე დიდი HMS იმ წვეროების სიაში, რომელთა დასახელებაც მონიშნულია i-თ.

$$g_i \coloneqq {}^{argmax}_{V \in \in \{S(V):S(V)=i\}} HMS(V)$$

19. -->განაახლე სითბური ცენტრები.

$$g_i = V_i, G = G \cup g_i$$

20. end for
21. end for
22. return G

3. სითბური ცენტრების რაოდენობის პოვნა და მათი იდენტიფიცირება

3.1 პროგრამული იმპლემენტაცია

```
void OglWidget::importFbx(QString path)
{
    if(!m_sdkManager)
        return;
    // Create fbx importer
    m_fbxImporter = FbxImporter::Create(m_sdkManager, "");
    if(m_fbxImporter->Initialize(path.toStdString().c_str()))
    {
        if(m_fbxImporter->Import(m_fbxScene))
        {
            // Convert Axis System to what is used in this example, if needed
            FbxAxisSystem SceneAxisSystem = m_fbxScene->GetGlobalSettings().GetAxisSystem();
            FbxAxisSystem OurAxisSystem(FbxAxisSystem::eYAxis, FbxAxisSystem::eParityOdd,
FbxAxisSystem::eRightHanded);
            if(SceneAxisSystem != OurAxisSystem)
                OurAxisSystem.ConvertScene(m_fbxScene);
            // Triangulate mesh
            FBXUtils::TriangulateRecursively(m_fbxScene->GetRootNode());
            // Bake the scene for one frame
            FBXUtils::LoadCacheRecursive(m_fbxScene, NULL, path.toStdString().c_str(),
true);
            // Set scene to renderer for draw
            m_mainRenderer->SetScene(m_fbxScene);
        }
        else
        {
            printf("Error during import scene!\n");
        }
    }
    else
    {
        printf("Cannot initialize FBX importer!\n");
    }
    m_fbxImporter->Destroy();
    m_fbxImporter = NULL;
}
```

სურათი 3.1.1 მეთოდი რომლიც ახდენს .FBX ფაილის იმპორტირებას აპლიკაციაში FBX SDK-ს გამოყენებით

```
void VBOMesh::Draw(int materialIndex, Renderer::ShadingMode shadingMode) const
{
    // Where to start.
    GLsizei lOffset = m_subMeshes[materialIndex]->m_indexOffset * sizeof(unsigned int);
    if ( shadingMode == Renderer::SHADING_MODE_SHADED)
    {
        const GLsizei lElementCount = m_subMeshes[materialIndex]->m_triangleCount * 3;
        glDrawElements(GL_TRIANGLES, lElementCount, GL_UNSIGNED_INT, reinterpret_cast<const
GLvoid *>(lOffset));
    }
    else
    {
        for (int lIndex = 0; lIndex < m_subMeshes[materialIndex]->m_triangleCount; ++lIndex)
        {
            // Draw line loop for every triangle.
            glDrawElements(GL_LINE_LOOP, TRIANGLE_VERTEX_COUNT, GL_UNSIGNED_INT,
reinterpret_cast<const GLvoid *>(lOffset));
            lOffset += sizeof(unsigned int) * TRIANGLE_VERTEX_COUNT;
        }
    }
}
```

სურათი 3.1.2 მეთოდი რომელიც ხატავს სამგანზომილებიან ობიექტს Vertex Buffer-ის გამოყენებით. ამ ტექნოლოგიით ხატვა გაცილებით ოპტიმალურია.

```
void Model::fillNormalizedLaplacian()
{
	float tempLap = 0;

	for ( int i = 0; i < m_size; i++)
	{
		for (int j = 0; j < m_size; j++)
		{
			if (i == j && m_degreeMatrix[j][j] != 0)
			{
				m_normalizedLaplacian(i,j) = 1;
			}
			if( i != j && m_weights[i][j] != 0 )
			{
				tempLap = sqrt( (float)(m_degreeMatrix[i][i] *
m_degreeMatrix[j][j]) );

					if (tempLap != 0 )
					{
						m_normalizedLaplacian(i,j) = -1/tempLap;
					}
					else
						m_normalizedLaplacian(i,j) = 0;
			}
			else
				m_normalizedLaplacian(i,j) = 0;
		}
	}

	for(int i = 0; i < m_size; i++)
		free(m_weights[i]);
	free(m_weights);

	for(int i = 0; i < m_size; i++)
		free(m_degreeMatrix[i]);
	free(m_degreeMatrix);
}
```

სურათი 3.1.3 მეთოდი რომელიც ავსებს ლაპლასის მატრიცას ნორმალიზებული მნიშვნელობებით

```
void Model::calcEigens()
{
	EigenSolver<MatrixXf> eshelper(m_normalizedLaplacian);

	m_eigenVect.resize(m_size,m_size);
	m_eigenVal.resize(m_size);

	m_eigenVal = eshelper.eigenvalues();
	m_eigenVect = eshelper.eigenvectors();
}
```

სურათი 3.1.4 მეთოდი რომელიც ითვლის საკუთრივ მნიშვნელობებს(EigenValues). ამისათვისგამოიყენება სპეციალური ბიბლიოთეკას Eigen.h

```
int Model::calcSegmentCount()
{
	QVector<float> tempDif;
	tempDif.resize(m_size);
	float tempMax = 0;

	tempDif[0] = m_eigenVal(1).real() - m_eigenVal(0).real();
	tempMax = tempDif[0];

	for ( int i = 1; i < m_size; i++)
	{
		tempDif[i] = ( m_eigenVal( (i+1) % m_size).real() - m_eigenVal(i).real() );

		if (tempMax < tempDif[i])
		{
			tempMax = tempDif[i];
			heatSegments = i;
			break;
		}
	}

	return heatSegments;
}
```

სურათი 3.1.5 მეთოდი რომელიც ითვლის სეგმენტების რაოდენობას და ახდენს მათ იდენტიფიცირებას

```
float Model::HeatAffinity(int i, int j)
{
	float heatAff = 0;

	for (int k = 0; k < m_size; k ++)
	{
		heatAff += exp(-m_eigenVal(k).real()) * m_eigenVect(i,k).real() *
m_eigenVect(k,j).real();
	}

	return heatAff;
}
```

სურათი 3.1.6 მეთოდი რომელიც ითვლის სითბურ მსგავსებას მიღებული ინდექსების მქონე წერტილებს შორის

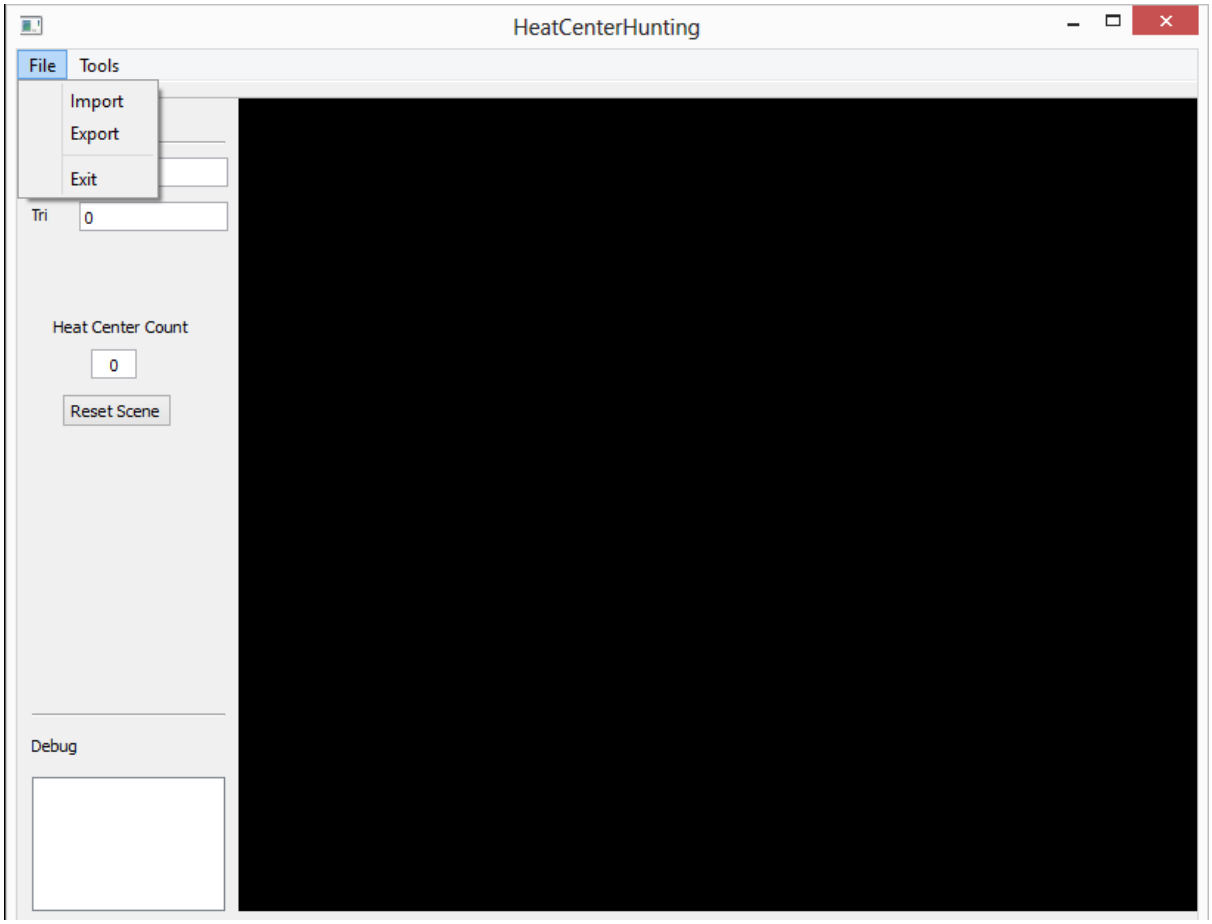
```
void Model::runHMS()
{
	int tempVert = 0;
	int tempMax = 0;
	float temphf = 0;
	QVector<float> tempHMS;
	QVector<float> cenVector;
	QVector<float> corVector;

	if(heatSegments <= 0)
		return;

	for (int i = 0; i < m_size; i++)
	{
		for (int j = 0; j < m_size; j++)
		{
			m_heatAff[i] = HeatAffinity(i,j);

			if( i != j)
			{
				m_HMSArray[i] +=  HeatAffinity(i,j)/m_size;
			}
			else
				m_HMSArray[i] = 0;
		}
	}

	m_heatCenters.resize(heatSegments);
	tempHMS.resize(heatSegments);

	tempVert = argMaxHC();

	m_heatCenters[0] = tempVert;

	for ( int  i = 0; i < m_size; i++)
	{
		m_vertLabels[i] = 0;
	}

	for ( int  i = 1; i < heatSegments; i++)
	{
		tempVert = argMinHA();
		m_heatCenters[i] = tempVert;

		for ( int j = 0; j < m_size; j++)
		{
			temphf = HeatAffinity(j,m_heatCenters[i]);
			if( temphf >= m_heatAff[j])
			{
				m_heatAff[j] = temphf;
				m_vertLabels[j] = i;
			}
		}
	}

	for ( int  i = 1; i < heatSegments; i++)
	{
		for ( int j = 0; j < m_size; j++)
```

```
{
                    if( m_vertLabels[j] == i)
                    {
                           tempHMS[i] = m_HMSArray[j];
                           if(tempMax < tempHMS[i])
                           {
                                  tempMax = tempHMS[i];
                                  m_heatCenters[i] = tempMax;
                           }
                    }
             }
      }
      m_enableSegment = true;
}
```

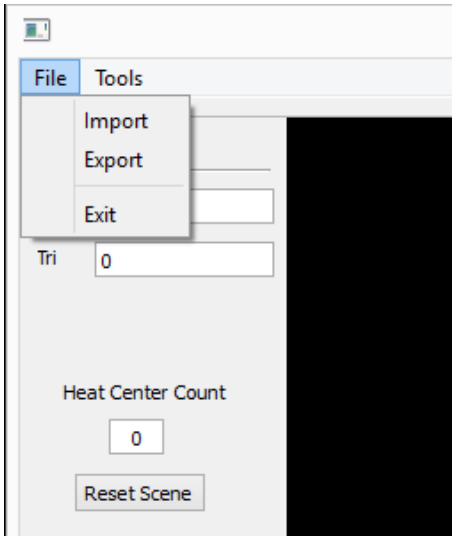
სურათი 3.1.7 ალგორითმის იმპლემენტაცია, რომელიც ახდენს სითბური ცენტრები რაოდენობის ძებნის ინიცირებას და ასახავს მიღებულ შედეგს მოდელზე.

3.2 პროგრამის მუშაობის აღწერა

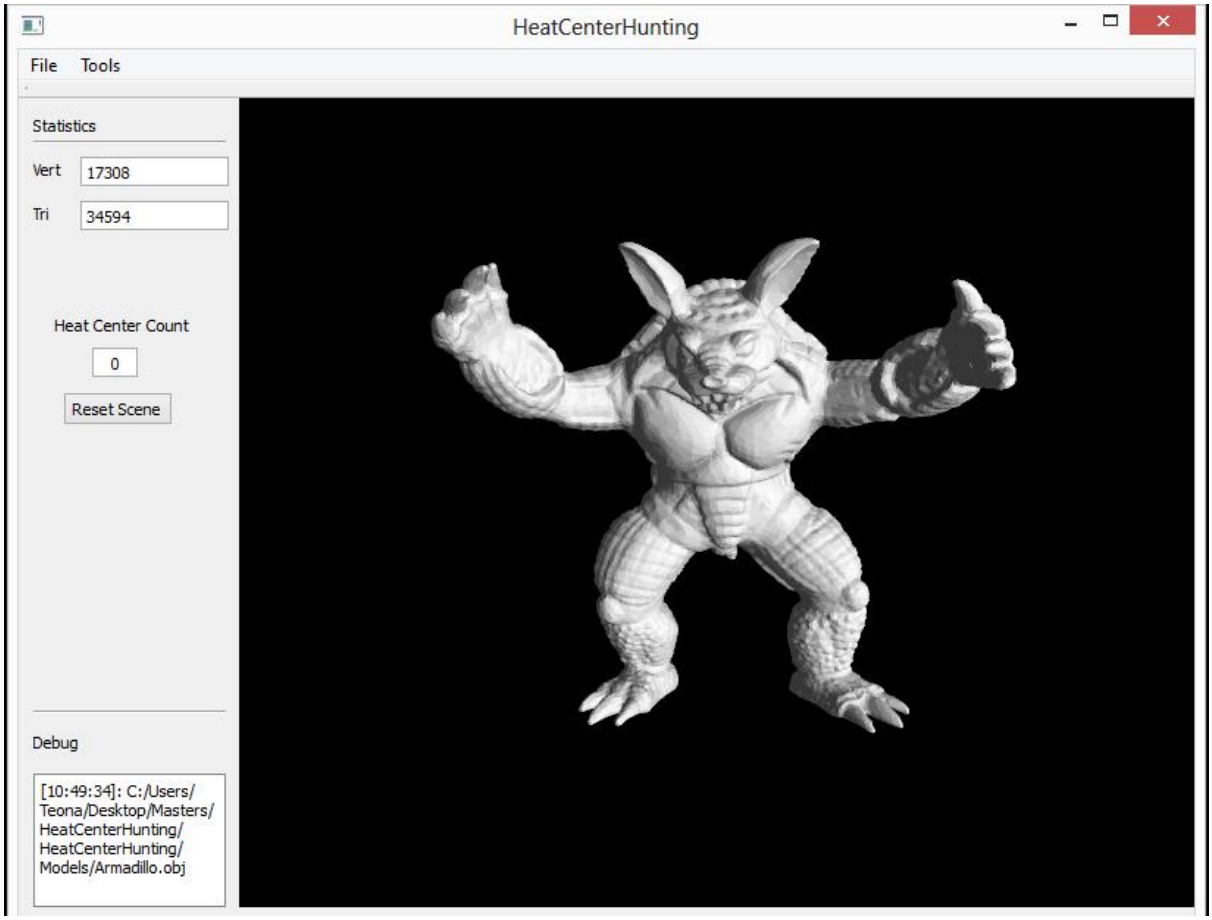
პროგრამის გაშვებისას გამოდის მთავარი ფანჯარა, რომელზეც განთავსებულია საწყისი ცარიელი სცენა და მომხმარებლის ინტერფეისი(სურათი 3.2.1). მომხმარებელს შეუძლია ჩატვირთოს სამგანზომილებიანი მოდელი, მოახდინოს სითბური ბირთვების ძებნის ალგორითმის ინიცირება, შეცვალოს მოდელის გადახატვის მეთოდები.



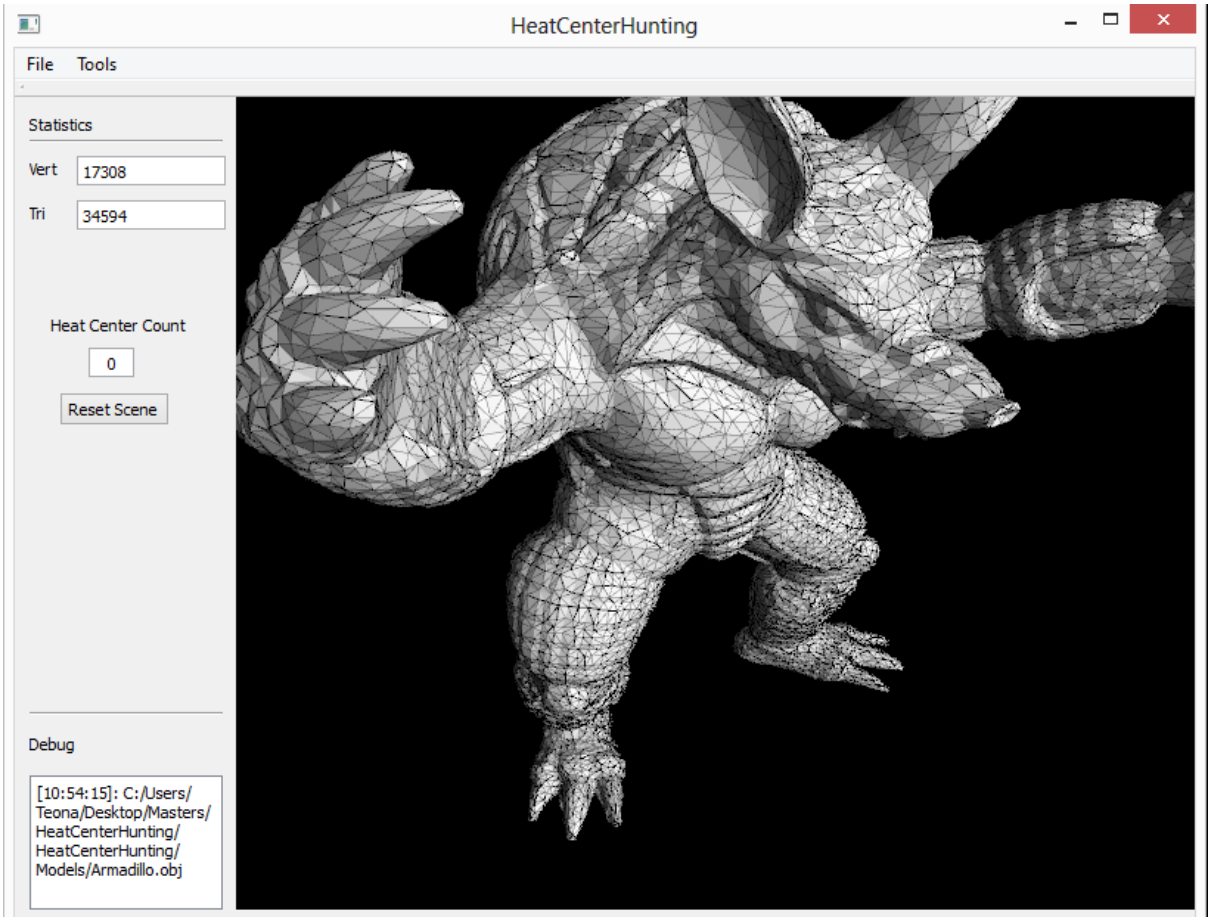
სურათი 3.2.1 პროგრამის მთავარი ფანჯარა



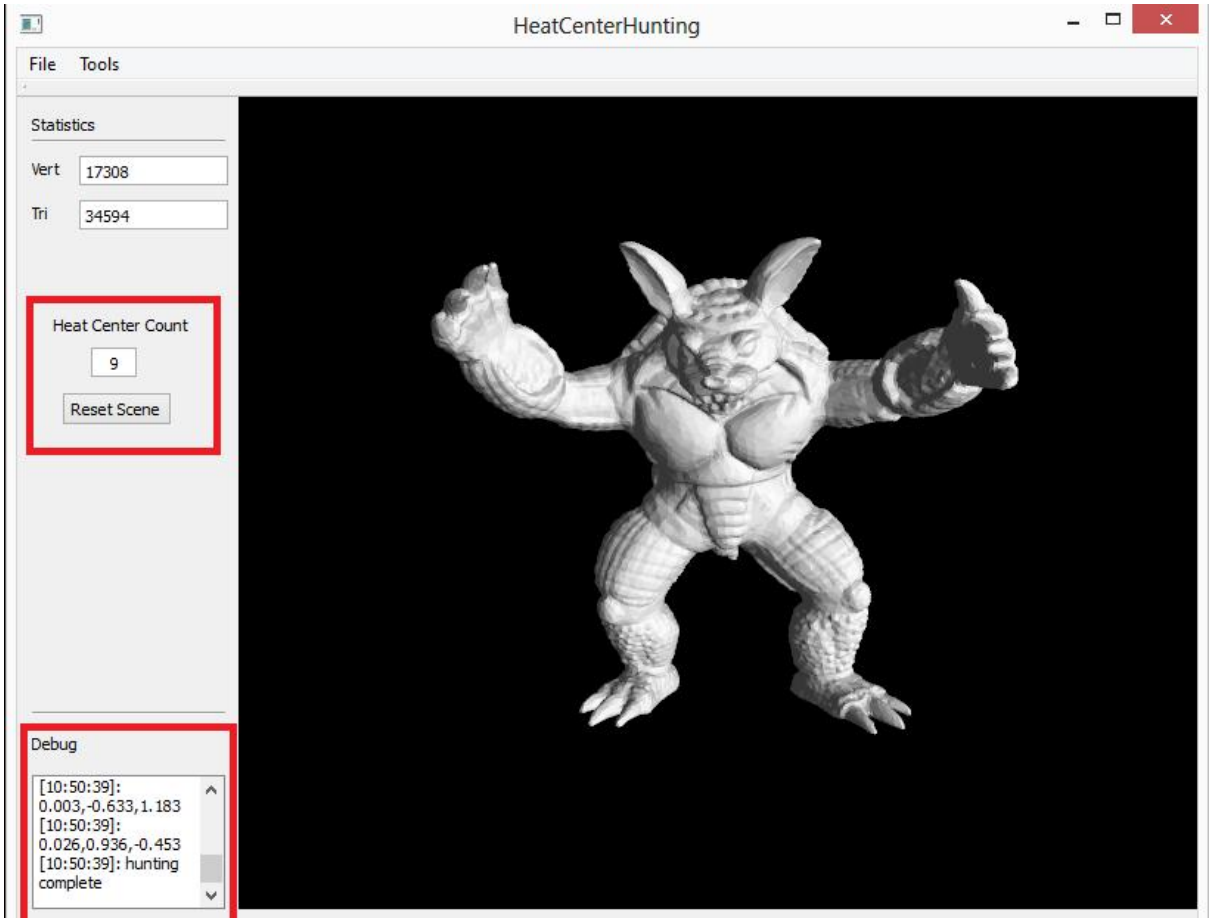
სურათი 3.2.2 პროგრამის მთავარი მენიუ



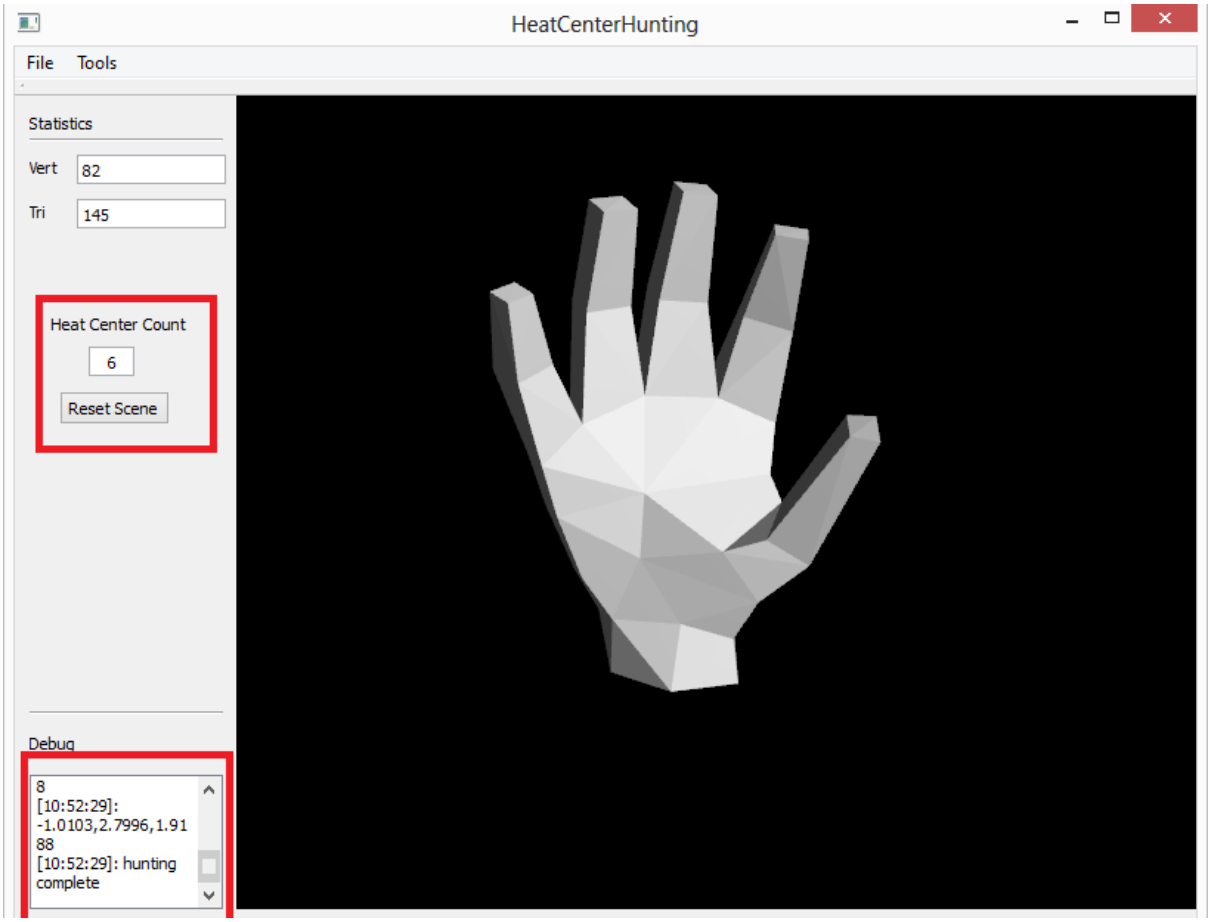
სურათი 3.2.3 სცენაში ჩატვირთული მოდელი



სურათი 3.2.4 ჩატვირთული მოდელის wireframe ვიზუალიზაცია



სურათი 3.2.5 Armadillo-ს მოდელზე ჩატარებული გამოთვლების შედეგები



სურათი 3.2.6 ადამიანის ხელის მოდელზე ჩატარებული გამოთვლების შედეგები

დასკვნა

ნაშრომში განვიხილეთ სამგანზომილებიანი ობიექტის სეგმენტაციის პროცესები. მოვახდინეთ სეგმენტებისათვის აუცილებელი ბირთვების პოვნის სხვადასხვა მეთოდების განხილვა და შედარება. ნაშრომის მთავარი მიზანი იყო პროგრამული უზრუნველყოფის შექმნა იმისათვის, რომ მოგვეხდინა სითბური ბირთვების რაოდენობის პოვნის და მათი იდენტიფიცირების ალგორითმების იმპლემენტაცია.

პროგრამული უზრუნველყოფა შეიქმნა Microsoft Windows 8-ს გარემოში, კერძოდ Microsoft Visual Studio 2010-ის გამოყენებით. პროგრამის ბირთვი და მთავარი გამოთვლები ხდება C++ ის გამოყენებით, მომხმარებლის ინტერეფეისის ფუნქციონალისთვის გამოყენებულია Qt-ს ბიბლიოთეკები, ხოლო სამგანზომილებიან გრაფიკასთან სამუშაოდ ვიყენებთ OpenGL-ის ბიბლიოთეკებს. უნდა აღინიშნოს, რომ საკუთრივი მნიშვნელობების და ვექტორების გამოსავლელად გამოყენებული იქნა C++ ის ბაზაზე შექმნილი წრფივი ალგებრის ბიბლიოთეკა Eigen-ი.

მიღებული შედეგები მდგრადია ზედაპირის მცირე ხარვეზების და ტოპოლოგიური დეფექტების მიმართ. ნაშრომის მომავალი განვითარებისათვის სასურველი იქნება საკუთვირივი მნიშვნელობებისა და საკუთრივი ვექტორების გამოსათვლელი ფუნქციების მოდიფიცირება(წარმადობის გაზრდის მიზნით).

გამოყენებული ლიტერატურა

1. Yi Fang, Mengtian Sun, Minhyong Kim, Karthik Ramani –“ Heat-Mapping: A Robust Approach Toward Perceptually Consistent Mesh Segmentation” – Purdue University, West Lafayette, IN 47907, University of College London, Gower Street, London WCIE, UK. Computer Vision and Pattern Recognition (CVPR), 2011 IEE Conference, Pages 2145-2152.
2. Sagi Kratz, George Liefman, Ayellet Tal – “ Mesh segmentation using feature point and core extraction” - Springer-Verlag – Septemper 2005-The Visual Computer , Volume 21, Issue 8-10, pp 649-658,.
3. OpenGL libraries - http://www.opengl.org/
4. Qt libraries - http://qt-project.org/
5. Eigen libraries - http://eigen.tuxfamily.org/index.php?title=Main_Page