Group Members: Manju Kumar BASAVARAJ / Olivier Sehou AGBOHOUI / Prem PRASAD

Midterm Report / Prof Dr Yohan Fougerolle

Software Engineering Project: " Harris 3D Operator "

Contents

1 Introduction

1.1 Brief Summary:

- Understand the research paper.
- Implement in C++ how to use Harris operator for finding interest point in 3D meshes.
- Create own functions/library where necessary.

1.2 Harris Operator:

- Operator used to help find interest points.
- Using SSD(sum of square differences).

1.3 Interest Points:

- Clearly defined points in an image.
- The region around an interest point has important information (ex: texture, changes,etc).

1.4 What we learn?

- How to successfully implement a research paper.
- Good level of C++ programming(Create a graphical interface,etc).
- 3D meshes and interest points.

2 Objectives

- Acquire a good use and understanding of Object Oriented Programming and QT.
- Understand the importance and usage of C++ programming language in Computer Vision Environment.
- Develop a good sense of teamwork and time management.

3 Members Roles

- Prem PRASAD:
  - Group Head
  - Github repository manager
  - Coding
- Manju Kumar BASAVARAJ:
  - Research Manager
  - Coding
- Olivier AGBOHOUI:
  - Project Manager
  - Reports (Latex) Manager
  - Coding

4 Project Structure

4.1 Flowchart

4.2 Steps

4.2.1 Finding Local Neighbourhoods

- The neighbourhood of a vertex can have different meanings. It could mean other vertices directly connected by an edge, or vertices that share the same face, or vertices that are depth 'd' from a given vertex.
- If given a vertex 'v' and a graph/mesh 'G', then one possible definition of neighbourhood of 'v' can be all vertices in G that are directly connected with vertex 'v' by an edge 'e'. This is an example of neighbourhood at depth 1. If we need to find the neighbourhood of a vertex at depth 'd', we apply the same procedure to the set of neighbouring vertices of initial vertex 'v', 'd' number of times.
- Following is the algorithm to find the set of direct neighbouring vertices of 'v':

  - Algorithm: to get the direct neighbours of vertex

```
Input: 'i' (index of vertex), 'Faces' (array of faces)
Output: S Neighbours
begin:
for all the faces at index j
  if Faces[j] contains point i
    add remaining indexes to S
  end for
remove duplicates in S
return S
end
```

  - Algorithm to find the set of vertices at depth 'd' from a given vertex

```
Input: i (index of vertex), 'Faces' (array of faces), 'd' depth, N(k) direct
    neighbours of P_k
Output: S_d neighbours at depth 'd'
begin:
$S_0$ <- P
$S_1$ <- N(i)
for x = 2 to d
  $S_x$ = N($S_{x-1}$) - $S_1$ - $S_0$
  $S_0$= $S_1$
  $S_1$= $S_x$
end for
return $S_1$
end
```

4.2.2 Fit Quadratic Surface

- Once we have the local neighbourhood of a given vertex, let v be the vertex under consideration, and Vk(v) the neighbourhood considering k rings around v, we need to fit a quadratic surface that best fits these points.
- To do so, we first do the following:
  - 1. Find the centroid of the neighbourhood
  - 2. Translate the centroid so that it sits on the origin of the 3D coordinate system
  - 3. Apply Principal Component Analysis (PCA) on the set of points
  - 4. The eigenvector with the lowest associated eigenvalue is taken as the normal to the fitted plane.
- The normal we obtain can be pointing in any direction, so we translate the points so that the normal of the fitting plane is pointing in the z-axis direction.

- This allows us to work on the XY plane and calculate the derivatives. But before calculating the derivatives, we need to translate the points so that the vertex under analysis is at the origin.
- We now fit a quadratic surface to the set of transformed points using the least square method to find a paraboloid. For a function with two variables 'x' and 'y', a paraboloid with six terms is best. We only need coefficients for 'x2', 'y2', 'xy', 'x', 'y', and a constant, i.e. total six terms.
  $f(x,y) = p_1/2x^2 + p_2xy + p_3/2y^2 + p_4x + p_5y + p_6$

4.2.3 Calculate Harris Response for each Vertex

- To calculate the Harris Response for a vertex, we need a matrix 'E' (explained below) to compute the response value using the following formula:
  $h(x,y) = det(E) - K(E^T)^2$ where 'k' is a constant.
- Matrix E is represented by:

  $$E = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$
  Where:
  $A = P_4{}^2 + 2P_1{}^2 + 2P_2{}^2$
  $B = P_5{}^2 + 2P_2{}^2 + 2P_3{}^2$
  $C = P_4P_5 + 2P_1P_2 + 2P_2P_3$

4.2.4 Selection of Interest Points

- At this stage, we have a Harris operator value associated with each vertex.
- We first need to find the vertices which are the local maximums. To find if a vertex is a local maximum, it has to satisfy the condition that its Harris operator value ?h(v)? should be greater than all its? first ring neighbours? Harris operator values. In mathematical terms:
  $h(v) > h(w), \forall w \in ring1(v)$
- The above gives us a set of points which are the local maximums, but the final set of interest points need to be found.
- To find the final set of interest points, we have several methods:
  - 1. Selecting the points with the highest Harris response:
    - Here, we just select a fraction of the interest points from the above step that have the highest Harris values.
    - This method only shows points with high saliency i.e. quality by which they stand out from their neighbours, and therefore some parts of the object do not have interest points.
  - 2. Representatives of Interest Points Clusters:
    - This approach is used when we want an even distribution of interest points on our object surface.
    - Two steps are needed to compute the representatives.
      - First we need to arrange the pre-selected interest points in descending order of Harris operator values.

· Second, we use the following algorithm to cluster the sorted points and select the final set of interest points:

```
Input: Set P of pre-selected interest points in decreasing order of
    Harris operator value
Output: Final set of interest points

1: Let Q be a set of points
2: Q <- {empty set}
3: for i <- 1 to |P| do
4:  if minj E [1,|Q|] ||Pi-Qj||2 > p then
5:  Q <- QU{Pi}
6:  end if
7: end for
8: Return Q
```

5 Useful Libraries

- The Point Cloud Library (PCL):
  is a standalone, large scale, open project for 2D/3D image and point cloud processing.PCL is released under the terms of the BSD license, and thus free for commercial and research use. We are financially supported by a consortium of commercial companies, with our own non-profit organization, Open Perception. We would also like to thank individual donors and contributors that have been helping the project.

- Eigen:
  is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms.

  Characteristics:

  – Eigen is versatile:
    It supports all matrix sizes and all standard numeric types.
  – Eigen is Fast:
    Fixed-size matrices are fully optimized.
  – Has a Good compiler support.
  – Eigen is Reliable.

6 What we have done so far ?

- Gone through the paper and understood it
- Researched on the algorithms
- Divided the work based on skill level
- Work in close collaboration with other groups

7 Conclusion

- 1. Next Steps: Now that we have a good understanding of the concepts, we are going to implement the individual steps mentioned above with code in Qt Environment. The work load will be divided accordingly based on difficulty level. We will update our GitHub repositories frequently as our code is being built.
- 2. Expected date of completion: Before Christmas vacations

8 References

- http://aishack.in/tutorials/harris-corner-detector/
- https://compvisionlab.wordpress.com/2013/03/01/harris-interest-point-detection-theory/
- https://users.dcc.uchile.cl/ isipiran/harris3D.htmlhttps://users.dcc.uchile.cl/ isipiran/harris3D.html
- http://aishack.in/tutorials/harris-corner-detector/
- http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm
- http://pointclouds.org/
- http://bitbucket.org/eigen/eigen/
- http://eigen.tuxfamily.org/index.php?title=Main$_{page}$