Computer Science Project Proposal

Constructing 3D models from image sequences

I. G. Sorescu, Newnham College, igs23

Originator: I. G. Sorescu

23^{th} October 2014

Project Supervisor: K. Palyutina

Director of Studies: Dr J. K. Fawcett

Project Overseers: Dr A. C. Rice & Dr T. G. Griffin

Introduction and Description of the Work

The aim of this project is to extract a partial 3D model of a small object from a sequence of plain images and then to export the model into a 3D geometry format such as PLY. The resulting file can then be rendered and processed by MeshLab or similar mesh processing software.
The main focus of the project is to obtain a partial wireframe model from a sequence of images taken from pre-established locations with respect to the object. An open-sorce ray tracer will be used in order to set up a static scene consisting of the object to be reconstructed placed in a well-lit environment against a white background. The sequence of images to be inputted into the 3D model constructor will be acquired by taking digital snapshots of this scene from the pre-established locations, thus imitating the process of photographing the object with a pre-calibrated camera. The project is designed to be later extended to support real camera input.
What follows is a description of a technique which might be used in order to obtain the image sequences. The setup is described here in terms of photographing real objects in order to emphasize the fact that the project can naturally be extended to interact with a real camera. However, for the purposes of this project, this technique will be imitated by digitally constructing the equivalent snapshots of an existing 3D model in an attempt to reconstruct its shape:
*D*efine an origin on a perfectly horizontal table and place the camera there facing a white background. Place an object of a pre-determined maximum height at a fixed distance from the origin in front of the camera. Take the first picture. Rotate the object by 30 degrees counter clockwise and take the second picture. Repeat until 12 pictures are taken. Input the pictures in the order in which they were taken. This process is assumed to have taken place in well-lit environment with no major light differences between pictures in the same set.
The outcome of any 3D model constructor used on this input is considered to be a partial solution because the input provides no information about the shape of the object as seen from above and below.

Starting Point

I am planning to use an open-source ray tracer such as POV-Ray in order to create image sequences from an existing 3D model in an attempt to recreate its shape. I will also be using the feature detection and feature matching functionalities of OpenCV.

Resources Required

Some open source libraries will be used as described in the previous section. They can be downloaded freely from http://www.povray.org/ and http://opencv.org/ respectively.
MeshLab will be used in order to view the resulting 3D model. It can be downloaded from http://meshlab.sourceforge.net/.

No special hardware resources will be required unless the extension to provide support for camera interaction is undertaken. The project is aimed at home users so the digital camera to be used should have no specific requirements. To ensure this I will be using my own cameras.

I am also planning to use my own machine, doing regular backups to github and to PWF. I can resume the development on any of the machines in the Computer Lab should my machine suddenly fail.

Substance and Structure of the Project

To ensure the successful completion of the project, the workload has been divided into the following manageable chunks:

1. **Research:** get familiarized with epipolar geometry and with the algorithms mentioned in this section.

2. **Module design:** split the project into modules, establish how they interact, construct the corresponding UML diagrams.

3. **Test harness:** implement the test harness using POV-Ray. Given a 3D model, the test harness will generate the image sequences to be inputted into the partial 3D model constructor by taking snapshots of the rendered model from each of the pre-established locations in turn. This module will be used in order to evaluate the success of the project by adding a measured error to the snapshots and measuring the accuracy of the outputted 3D model. Error can be added by rendering the object at a distance ϵ from the specified location, or by adding blur etc.

4. **First prototype:** implement triangularization in order to get the first prototype of a partial 3D constructor.

5. **Refinement of the partial 3D constructor:** This step consists of iteratively improving the current prototype in order to get a more complete and more accurate 3D model. Some of the possible prototypes are listed below. They need not be all implemented if the current results are satisfactory.

If multiple satisfactory prototypes are implemented they will be evaluated individually and included in the final solution. The users can then specify which one they would like to use.

- Prototype #2: Implement dense stereo matching [1][2] to estimate the depth of each point from a view point. Convert the results into a depth map [2]. The depth map can be directly used as a 3D model by displaying darker shades for more distant points. However, this cannot be exported into a common 3D geometry format so it should be used in conjunction with the algorithms described in prototype #3 to provide a complete solution. I consider this partial solution to be a prototype by itself because the ability to visualize the output enables me to evaluate the results of this intermediate step. This might prove useful in order to discuss whether using meshes was the right choice since many other approaches could be taken instead.
- Prototype #3: Construct the corresponding 3D mesh using depth map constructed in prototype #2. A possible approach consists of overlying a 2D triangular mesh on top of one of the images and then wrapping the vertices of the triangles to correspond to the values in the depth map [1].
- Prototype #4: Use a volumetric depth map integration approach [1].
- Prototype #5: Implement a Kalman filter [1].

6. **Evaluation:** Evaluate the results using the test harness by adding a measured error to the input and comparing the output to that obtained from the equivalent error-free images.

7. **Write the dissertation**

Success Criteria

The following must necessarily be achieved for the project to be considered a success:

- A tool for constructing 3D models from a sequence of digital snapshots produced as described in the introduction has been implemented. The tool will be evaluated and shown to be working by applying it on a input consisting of consistent snapshots of a cube.

Possible Extensions

- Add a camera module in order to provide support for real camera input (instead of using digital snapshots generated using a ray tracer).

- Add shadowing to the outputted 3D mesh.
- Add texture to the outputted 3D mesh.
- Implement structure-from-motion and self-calibration: this would allow the users to input virtually any set of pictures representing an object as long as any consecutive pictures have a large overlap (i.e.: no need to pre-establish the camera pose for each photo in the sequence).

Timetable

Planned starting date is 09/10/2014.

1. **Slot 0: 9^{th} Oct - 24^{th} Oct**

   - Read relevant literature and plan the project.

   Milestone: Submit proposal

2. **Slot 1: 25^{th} Oct - 14^{th} Nov**

   - Further research: gain a deeper understanding of the techniques involved in edge detection and 3D modelling

   Milestone: Have a clear understanding of the techniques needed to complete the project

3. **Slot 2: 15^{th} Nov - 28^{th} Nov**

   - Start implementation: implement the digital renderer
   - Use the renderer to design a few basic unit tests for the edge detector, dense set generator and wireframe generator.

   Milestone: finish implementing the digital renderer and the test harness

4. **Slot 3: 28^{th} Nov - 5^{th} Dec**

   - Implement the wireframe generator. This should be implemented first because it is the riskiest module of the project.

   Milestone: Pass the unit tests for the wireframe generator

5. **Slot 4: 6^{th} Dec - 26^{th} Dec**

   - Implement dense set generator

Milestone: Pass the unit tests for the dense set generation module

6. **Slot 5: 27^{th} Dec - 16^{th} Jan**

   - Implement the edge-detector

   Milestone: Finish implementation, Pass the unit tests for the edge-detection module

7. **Slot 6: 17^{th} Jan - 30^{th} Jan**

   - Buffer time: catch up or start doing the extensions
   - Write progress report

   Milestone: Submit progress report

8. **Slot 7: 31^{st} Jan - 13 th Feb**

   - Further tests

   Milestone: Finish writing integration tests and more in-depth unit tests

9. **Slot 8: 13^{st} Feb - 27 th Feb**

   - Debug

   Milestone: Pass all of the tests

10. **Slot 9: 27^{st} Feb - 13^{th} Mar**

    - Catch-up time or extensions

    Milestone: Pass all of the tests

11. **Slot 10: 14^{th} Mar - 3^{rd} Apr**

    - Analysis

    Milestone: Finish doing the evaluation graphs

12. **Slot 11: 4^{th} Apr - 17^{th} Apr**

    - Plan and start writing the dissertation

    Milestone: Write the main parts of the dissertation

13. **Slot 12: 18^{th} Apr - 8^{th} May**

    - Write the dissertation

Milestone: Finish writing dissertation

14. **Slot 13: 9^{th}May - 15^{th}May**

    - Safety slot

    Milestone: Submit dissertation

References

[1]M. Pollefeys, *3D from Image Sequences: Calibration, Motion and Shape Recovery*, Mathematical Models of Computer Vision: The Handbook, N. Paragios, Y. Chen, O. Faugeras, Springer, 2005.

[2] R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer Verlag 2011.