3D Reconstruction

R&D Project Manual

Sriram Kashyap M S
Roll No: 08305028

Under the guidance of

Prof. Sharat Chandran

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai

2009

Table of Contents

1 Introduction

This project involves capturing images of an object from multiple view points and building a 3D model (the visual hull of the input images) out of it. This model will be further used to perform image based rendering of the object from novel viewpoints.

The project augments the initial phase of the existing Image Based Animation framework. The currently available system has been tested on synthetic inputs from rendered 3D scenes. The main goal of the project will be to build an image acquisition system which will simplify the process of capturing images of real-world objects and extracting 3D object information from these images.

The phases involved are as follows:

1. Create a image capture setup: This setup will provide object images and corresponding calibration images
2. Image Segmentation: The visual hull construction algorithm requires each image pixel to be binary-classified as background/foreground
3. Camera Calibration: Given a set of images with a calibration pattern in them, compute the camera matrices corresponding to those views
4. System Integration: Modify the existing system so that it can seamlessly accept data coming in from the above steps

This document describes the tools written as part of the reconstruction project, and the process of using these tools.

2 List of tools

1. Camera Control(cam.o): Used to capture a sequence of images from any connected webcam (streaming video device)
2. Pattern Finder(points.o): Used to locate calibration pattern features in a given image
3. Calibration Tool(cvCalib.o): Camera calibration tool, to generate camera matrices from point correspondences
4. Matrix Visualizer(render.o): Visual representation of generated camera matrices (a manual verification tool)
5. Segmentation (segment.o): Background subtraction utility
6. PathEditor (patheditor.o): Modified 3d reconstruction and relighting tool

2.1 Image Capture

```
Usage: ./cam.o <number of images> <delay(sec)> [width] [height]
```

1. The captured images will be output to the current directory, with sequential 3 digit names (000.png, 001.png...)
2. The delay can be fractional. For example, 1.5 is a valid number
3. The delay is only the minimum delay between capture of frames. If the capture device is slow, the actual time between frames can be longer. It is advisable to set this depending on how fast the actual camera is, so that the time between each frame is constant.

4. The width and height are only indications of desired capture dimensions. The camera can choose to ignore these values if it does not support them. In such cases, a fallback resolution is automatically chosen by the camera driver (usually 640x480)

2.2 Pattern Finder

`Usage: ./pattern <inputfile> [mode 0/1...] [debugmode 0-2] [threshold 0-255]`

1. inputfile: The first line of this file contains the name of the image file to be read. Remaining lines can contain other data depending on the mode
2. mode: Mode 0 is default calibration mode. It is no longer used. Any number greater than zero specifies the number of calibration squares to search for. This project uses 64 as the mode parameter. In this case, the program outputs the (x,y) positions of the centers of black squares in the image
3. debugmode: 0=Silent mode, 1=Basic debug mode, 2=Full debug mode
4. threshold: Starting threshold for finding boxes. Not required (default 0)
5. Other Inputs: color.conf: This conf file should be present in the same directory as pattern finder, and contain 3 RGB values (9 integers). The first 3 numbers denote RGB values of the black squares, the next 3 numbers denote RGB colors for red boxes, and the next 3 numbers denote RGB colors for green boxes. Note that the names black, red and green are used for convenience and the colors need not actually be these values.

2.3 Camera Calibration

`Usage: ./cvCalib.o <2d-datafile> <3d-datafile> <numPoints> <numImages> <width> <height>`

The output is a list of numImages number of camera matrices (3x4) followed by a 3 component inverted view vector each.

1. 2d-datafile: Contains 2d point values (x and y separated by spaces/newlines) for N images
2. 3d-datafile: Contains corresponding 3d point values (x and y separated by spaces/newlines). These values are same for all images.
3. numPoints: Number of points specified in the 3d-datafile
4. numImages: Number of images for which 2d points are specified in 2d-datafile
5. width and height: Width and height of input images from which 2d-Datafile was generated

2.4 Matrix visualizer

`Usage: ./render.o <numMatrices>`

This tool renders a square centered at 0,0 and of 2 units length, using a list of camera matrices specified in a datafile named "camlist".

The numMatrices parameter specifies how many matrices should be rendered. To render the next matrix, set focus on the render window and press any key.

Note that this tool takes as input the matrices generated by the calibration tool. Therefore, it assumes there will be 3 extra numbers after every 12 numbers (view vector is attached to each 3x4 matrix). These 3 numbers should be there, and will be ignored.

2.5 Segmentation

```
Usage: segment.o <RGB threshold> <CbCr threshold> <foreground Ratio> [list of images]
```

This tool subtracts the image 'back.png' (which is assumed to be present in the current directory) from the list of images and generates images with transparent backgrounds, each suffixed with ".seg.png".

1. RGB threshold: Specifies starting threshold for pixel difference based on RGB values (typically 30, can be zero if foreground Ratio is known)
2. CbCr threshold: Specifies starting threshold for pixel difference based on CbCr values (typically 5 to 10, can be zero if foreground Ratio is known)
3. Foreground Ratio: Maximum ratio of number of foreground pixels to total pixels in image. Use this to adaptively segment the images. If the ratio is unknown, set it to 1, and experiment with various values of RGB and CbCr thresholds to find the ratio.
4. List of Images: paths to images that should be segmented. The segmented output images have suffixes as described above, and are created in current directory. The background pixels in these images have alpha =0, remaining pixels have alpha=1.

2.6 Patheditor

This section describes internal changes to the patheditor tool. For full documentation, refer to the original patheditor manual.

The input format for dataset file has changed. Earlier, the dataset contained a list of xyz coordinates of the cameras. Now it contains 15 floating point numbers delimited by spaces/tabs, per camera. The first 12 numbers are the 3x4 matrices for the cameras and the next 3 numbers denote the inverted view vector of the camera (approximating the position of the camera).

3 Usage

3.1 Preparing the dataset

1. Capturing N images of calibration pattern and N corresponding images of the object (use cam.o as described above)
2. Place the calibration images under

   ```
   data/<dataset name>/cam/
   ```

3. Place the object images under

   ```
   data/<dataset name>/obj/
   ```

4. Create a file

   ```
   data/<dataset name>/dimensions
   ```

   which contains the visual hull resolution (points/unit distance), and the starting and ending values of x, y, z for the bounding box surrounding the object to be reconstructed. Refer to the patheditor manual for more information.

5. Create a file

   ```
   data/<dataset name>/color.conf
   ```

   which contains the color values as described in the Pattern Finder section

6. Create a file

   ```
   data/<dataset name>/back.png
   ```

   which contains the background image to be used for background subtraction. Note that all the images in cam directory, obj directory and the file back.png must have same dimensions, and must be png files.

3.2 Basic operation

1. Execute

   ```
   ./run.sh <dataset name> all <threshold ratio>
   ```

   where dataset name is the name of the directory inside "data" directory, under which the dataset is stored, "all" denotes that all processing should be performed, and Threshold ratio is as described in the segmentation section.

2. After processing, the script will output a line: "dataset file: ...". This is the path to the dataset file which will be required for the patheditor tool.

3. The patheditor will automatically be launched. Select the perspective view, press "h" key, and follow the instructions to generate a visual hull.

3.3 Troubleshooting

Sometimes, the visual hull obtained by the above process can be grossly incorrect. To fix this follow the steps detailed below:

1. Run

   ```
   ./camtest <dataset name>
   ```

   This will start up the matrix visualiser. Check manually whether each matrix is behaving as required. Note down the sequence numbers of each incorrect matrix (these numbers are displayed in the console).

2. Open

   ```
   data/<dataset name>/passed.txt
   ```

   This file contains a list of valid calibration images. Remove those sequence numbers that were found to be incorrect in the previous step. Remember that the sequence numbers obtained in previous step are zero based.

3. Open

   ```
   data/<dataset name>/matrices.txt
   ```

   Delete those lines in this file that correspond to the sequence numbers of the incorrect matrices. Note here that the line numbers usually start from 1, while the sequence numbers are Zero based.

4. Now run

   ```
   ./sequence.sh
   ```

   ```
   ./dataset.sh
   ```

   to re-select the images and re-generate the dataset file.

5. Run

   ```
   ./run.sh <dataset name>
   ```

   Note that the "all" parameter is missing. This means the entire process need not be repeated. Only the patheditor is restarted and visual hull construction can be performed as specified in the last step of "Basic Operation" section.

3.4 Notes

1. To generate color.conf file, open any calibration pattern image in a picture editor, pick the red and green colors and check for their RGB values. Note that if you have black squares, it is best to ignore the actual RGB value and specify it as 0 0 0 in the color.conf file.

2. To generate back.png, the normal method is to take an image of the setup without any object on the turntable. But this may produce washed out results depending on the camera. An alternative method is to open an image which has the object + background in a image editor and use the clone tool to manually remove the object. This is found to be simple and effective.