Advanced Vision Practical 2

Bob Fisher
School of Informatics

February 2017

Abstract

This describes the second assignment for assessment on Advanced Vision. The main goal is to analyse the 3D reconstruction accuracy of a 1000 frame per second depth sensor. The assignment is due: **4pm Thursday 23 March**. You must do this practical in teams of 2, and submit 1 PDF report only. There will also be an assessed live demonstration of your practical on Friday morning, March 24.

Task Background

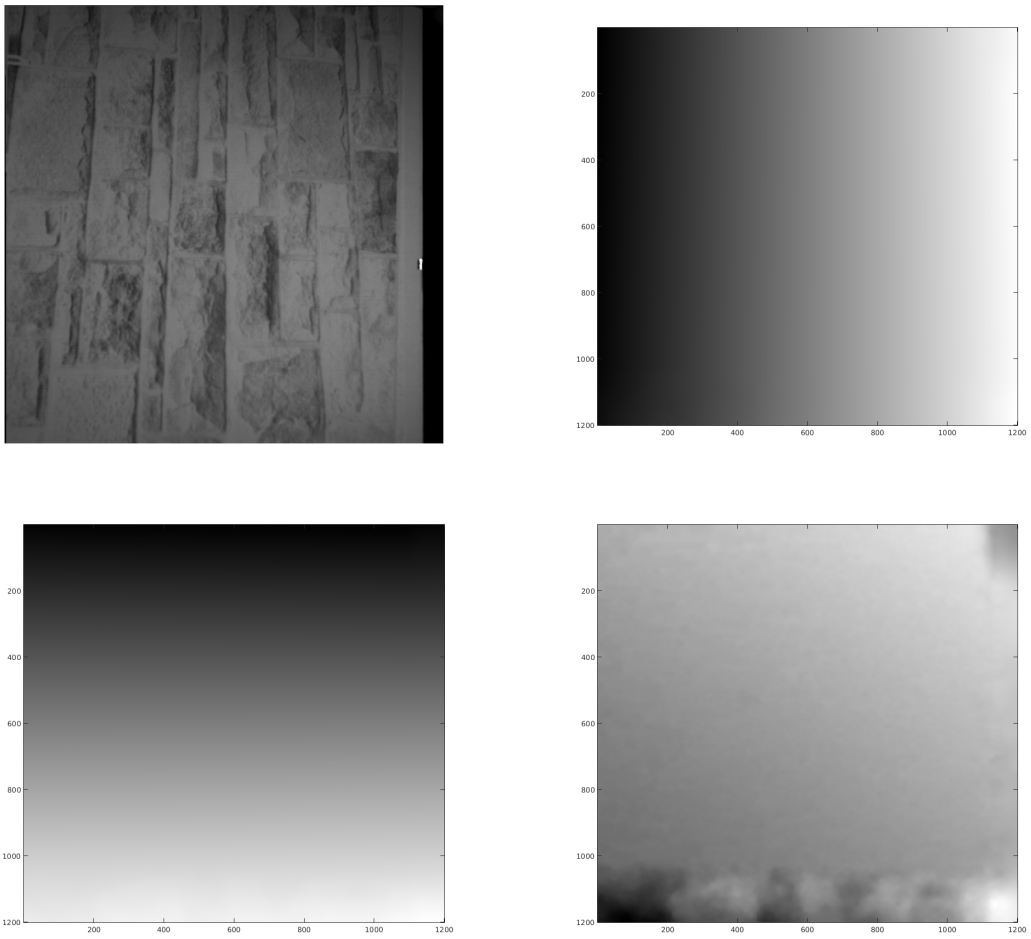
The data for this practical can be found at:

`http://homepages.inf.ed.ac.uk/rbf/AVDATA/AV217DATA/zstatic.tar`
`http://homepages.inf.ed.ac.uk/rbf/AVDATA/AV217DATA/xmove.tar`
`http://homepages.inf.ed.ac.uk/rbf/AVDATA/AV217DATA/ymove.tar`
`http://homepages.inf.ed.ac.uk/rbf/AVDATA/AV217DATA/zmove.tar`
`http://homepages.inf.ed.ac.uk/rbf/AVDATA/AV217DATA/fscatter32.m`
`http://homepages.inf.ed.ac.uk/rbf/AVDATA/AV217DATA/load_pcl_mat.m`
`http://homepages.inf.ed.ac.uk/rbf/AVDATA/AV217DATA/plotpcl_mesh.m`

The goal of the practical is to estimate the 3D reconstruction accuracy of our 1000 frame per second stereo depth sensor. The data that is observed is from a planar surface (a sheet of glass) that has a textured sheet of paper pasted onto it. The sensor is a binocular stereo system, in this case with the 2 cameras aligned vertically. The stereo calculation algorithm is related to the dense stereo algorithm presented in the lectures, except more sophisticated.

One of the intensity images can be seen at the top-left of the figure below. You can see there is a lot of texture, so that the stereo system can match the two images more reliably. If the pixels were all the same colour, then each pixel could match with any other pixel and the stereo calculation would produce bad results.

This is a dense stereo algorithm, meaning that it computes a depth (or 3D point) for each pixel. There are a variety of ways to visualise the 3D data computed by the stereo algorithm. Here, each pixel has an associated 3D point. We plot the x component, y-component, and z-component of these points in the top right, bottom left and bottom right of the figure below.



Note that the x and y components show a smooth gradient. This is because this coordinate value increments smoothly as the image column or row increases. The z component is dependent on the shape and position of the surface itself.

Note that there is a large smooth patch of z values (the planar patch), but also some substantially different values at the bottom and possibly sides. The values at the bottom are invalid, and arise because the top and bottom cameras do not see exactly the same portion of the scene. The depth triangulation only works if the same points are matched. If there is nothing to correctly match, the algorithm might produce erroneous data. A similar effect can happen at the sides of the image. We will want to avoid these areas of bad data.

Each of the 4 tar files listed above has data from 11 depth video frames in a different data situation:

- `zstatic.tar`: The plane surface is placed at 11 different depths based around the calibration position. The plane is stationary.
- `zmove.tar`: This set has 11 depths, except that the plane is moving towards the sensor.
- `xmove.tar`: This set has the surface moving horizontally.
- `ymove.tar`: This set has the surface moving vertically, which is an interesting case because the motion is now aligned with the epipolar lines in the image. Image blur in this direction might make matching harder.

The main goal of the practical is to analyse the accuracy of the sensor over different ranges, and with different motions. The analysis will be largely based on 4 plots. Each plot shows the

amount of RMS error as a function of the average x, y, or z depth, depending on the direction of motion. From these plots, one can predict the usable working range of the sensor. More details are given below.

Data Files

Each of the tar files contains 11 matlab files with filenames like `planez_04.mat`. The data in this file can be loaded into matlab using the command: `[DATA] = load_pcl_mat('planez',4);`, where the first argument is the filename prefix and the second the file number.

The file numbers will be different for each set. In the case of the 3 moving data sets (xmove, ymove, zmove), the data is captured at 100 FPS. Thus, if the file numbers differ by N, then the time between image captures is $N/100$ seconds. You will need this figure to estimate motion speeds.

The loaded file gives an array `DATA(ROWS,COLS,VALS)` where the ROWS and COLS correspond to the row and column positions of the original top camera image. VALS is a 6 dimensional value $(red, green, blue, x, y, z)$, where $(red, green, blue)$ is the colour of the scene point that corresponds to the current row and column position in the image. (x, y, z) is the 3D position of the corresponding point.

In this case $red = green = blue$ because the original image was grey.

The x component of all of the points as seen in the image above was created in Matlab by:

```
figure(1)
imshow(DATA(:,:,4))
```

Other Support Functions

The function `plotpcl_mesh(DATA)` will show the 3D points as a mesh in a 3D viewer, with the points coloured with their original intensity. (The function `fscatter32` is an auxiliary function.)

Error Computation

In the lectures, we saw how to fit a 3D plane to a set of 3D points. Here, the points $\{(x_i, y_i, z_i)'\}$ will need to be augmented with a final 1 value to become $\{\vec{d_i}\} = \{(x_i, y_i, z_i, 1)'\}$. Assume that the least square plane fit solution was the 4 vector $\vec{p}$. In other words, the each of the data points $\vec{d_i}$ approximately satisfies $\vec{p}\,'\vec{d_i} \doteq 0$.

We want the RMS (Root Mean Square) error of the plane fitting. We can calculate this from the error between each point and the fitted plane. If $\vec{p} = (p_x, p_y, p_z, p_d)'$ then compute an equivalent plane description $\vec{q} = \vec{p}/\sqrt{p_x^2 + p_y^2 + p_z^2}$. This makes the surface normal of the plane to be a unit vector, and then $e_i = \vec{q}\,'\vec{d_i}$ is the signed distance of the point from the plane. If it is on one side of the plane, the sign is positive; if on the other side, then it is negative (which side has which sign depends on the direction of the surface normal).

To calculate the RMS of the plane (*i.e.* its roughness) from a set of N points $\{\vec{d_i}\}$, compute:

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(e_i)^2}$$

It turns out that the process of computing the least squares plane given in lecture also leads to the RMS. However, here we will also want to know the error e_i at each point.

IMPORTANT NOTE: If you look at the intensity and Z images, you will see that not all of the points have a valid depth value. There are several possible causes: 1) the pixel is from a point not in the field of view of both cameras, 2) the pixel is not part of the textured test surface, or 3) the stereo algorithm did not work well at this pixel. When calculating the performance statistics, use a 500×500 region of the data (or as much as is possible), that avoids conditions 1 and 2 above.

Do not deliberately avoid condition 3. Also: use data from at least 50 pixels away from the edge of the image to avoid any image boundary problems.

Outputs

There are 8 images to compute from the data, each of these to be computed from data taken from the 500×500 region discussed above.

1. For the zstatic dataset, a graph of RMS *versus* average z value.
2. For the zmove dataset, a graph of RMS *versus* average z value.
3. For the xmove dataset, a graph of RMS *versus* x scene position. The x scene position calculation requires some interaction. You need to find a specific point on the textured surface that is visible in all 11 images. Use the x value from the (x, y, z) position corresponding to that point. The x values should change approximately uniformly.
4. For the ymove dataset, a graph of RMS *versus* y scene position. The y value needs to be calculated in a similar manner to the x value for xmove graph.
5. The final 4 images are colour images of the residuals over the 500×500 region from the middle image of each of the 4 datasets. At each pixel i calculate the residual e_i as described above. Let σ be the RMS value for this image's 500×500 region. Create a 500×500 colour image where the pixel has the colour:
   - If $e_i < -\sigma$ use red.
   - If $-\sigma \leq e_i < 0$ use yellow.
   - If $0 \leq e_i < \sigma$ use green.
   - If $\sigma \leq e_i$ use blue.

In addition to the 5 images, include in your report a table, with 1 row for each of the 44 test images. Include in the table: { image name, number of pixels actually used (maximum 500×500), RMS, number of pixels with $\mid e_i \mid > 5 \times RMS$ }. The latter gives an indication of the number of outliers.

Your Report

Each team writes a single report that describes:

- The algorithms that you used for each stage of the process.
- How well the algorithms performed on the supplied test data. Show the statistical results and images requested above.
- Show some example images of each processing stage, including original images with 500×500 regions outlined.
- Show some examples of any unsuccessful stereo reconstructions.
- Discussion on performance: successes and failures, causes of failures and potential remedies.
- As an appendix, add the code that your team wrote. Do not include code that was downloaded from the AV or IVR or other web sites.

Other Comments

1. You can use the lecture example code from:
   `http://www.inf.ed.ac.uk/teaching/courses/av/MATLAB/`
2. Because there are a limited number of MATLAB Image Processing library licenses available, use alternative MATLAB functions from
   `http://www.inf.ed.ac.uk/teaching/courses/av/MATLAB/UTILITIES/`

Assignment Submission

Submit your report in PDF online by 4pm Thursday March 23. The online submission line on the School's DICE system is:

```
submit av 2 FILE
```

where `FILE` is the name of your PDF file.

Live Demonstration

There will also be a demonstration session assigned between 9:00-13:00 on Friday March 24, where you will have to demonstrate your code. We'll email you about the location and schedule.

You will need your matlab program to show, for an image selected at random by the marker:

1. The original intensity image, with the 500×500 region outlined.
2. The associated x, y, and z data for the 500×500 region.
3. The 4 colour image of the residual magnitude.
4. The RMS value calculated for the 500×500 region.

The assignment is estimated to take 10 hours coding/test and 5 hours report writing per person, resulting in a 5-10 page report plus the code appendix. You must do this assignment in teams of 2. You must find your partner and email Bob Fisher (rbf@inf.ed.ac.uk) the name of your partner. **You must have a different partner from the first practical.**

A single, joint, report is to be submitted. Split the work so that each partner does most work independently (i.e. share the work rather than duplicate it).

The assignment will be marked as follows:

Issue	Percentage
1. Clear description of sensible algorithms used	30%
2. Performance on the data set	30%
3. Clear Matlab code	10%
4. Discussion of result quality and causes of any failures	10%
5. Live demonstration performance	20%

Publication of Solutions

We will not publish a solution set of code. You may make public your solution **but only 2 weeks after the submission date**. Making the solutions public before then will create suspicions about why you made them public.

Good Scholarly Practice: Please remember the University requirement as regards all assessed work for credit. Details about this can be found at:

http://www.ed.ac.uk/schools-departments/academic-services/students/undergraduate/discipline/academic-misconduct

and at:

http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct

Furthermore, you are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on a public repository then you must set access permissions appropriately (generally permitting access only to yourself, or your group in the case of group practicals).

Plagiarism Avoidance Advice

You are expected to write the document in your own words. Short quotations (with proper, explicit attribution) are allowed, but the bulk of the submission should be your own work. Use proper citation style for all citations, whether traditional paper resources or web-based materials.

If you use small amounts of code from another student or the web, you must acknowledge the original source and make clear what portions of the code were yours and what were obtained elsewhere. You can ignore this condition for the AV lecture examples, which can be used freely.

The school has a robust policy on plagiarism that can be viewed here: http://web.inf.ed.ac.uk/infweb/admin/policies/guidelines-plagiarism.

The school uses various techniques to detect plagiarism, included automated tools and comparison against on-line repositories. *Remember: a weak assignment is not a ruined career (and may not reduce your final average more than 1%), but getting caught at plagiarism could ruin it.*

Late coursework policy

See: http://web.inf.ed.ac.uk/infweb/student-services/ito/admin/coursework-projects/late-coursework-extension-requests