# Advanced Vision Practical

Bob Fisher
School of Informatics

February 2018

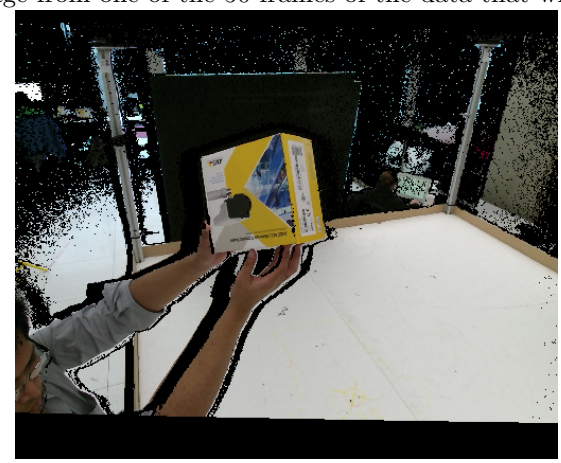
**Abstract**

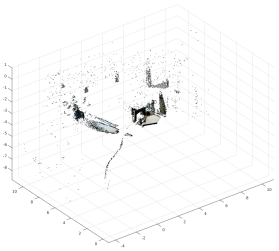
This describes the Advanced Vision assessed practical. The main goal is to reconstruct a box from a set of 3D point clouds acquired from a Kinect sensor that is rotated by a person. The assignment is due: **4pm Thursday 22 March**. You must do this practical in teams of 2, and submit 1 PDF report only. There will also be an assessed live demonstration of your practical on Friday morning, March 23.

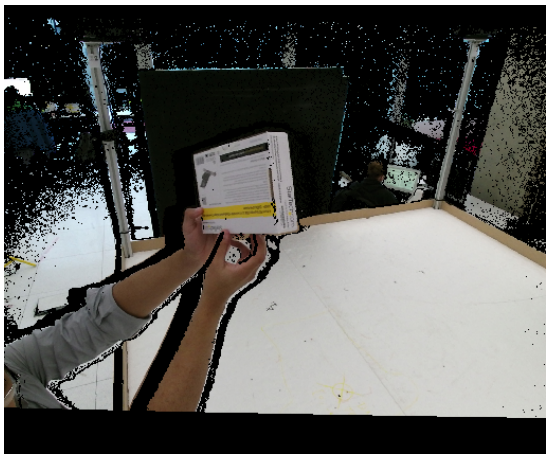
## 1 Task Background

The data for constructing the complete box comes from 50 3D point clouds acquired by a Kinect sensor. The core of the assignment is to estimate how the 3D points captured in each frame can be transformed into points in the initial frame. Once all points are transformed, in theory one can see a complete box.

In the top left image below, you can see an RGB image of the data from one frame. The black points are where there is no 3D data. The top right is a closeup of the 3D points (coloured with the surface colour). The bottom left is the full set of data in the point cloud. The box is in the centre of the plot. Most of the data points are irrelevant. The bottom right image is an RGB image from one of the 50 frames of the data that will be used in the live demonstration.







## 2 Resources

The data (188 MB) for this practical can be found at:

`http://homepages.inf.ed.ac.uk/rbf/AVDATA/AV118DATA/assignment_1_box.mat`

The data consists of 50 frames of 3D point cloud data, where each frame is from a different viewpoint. The data was acquired by a Kinect RGBD sensor, and is stored in a cell structure with 50 cells with 2 entries: X.Color which is the RGB value of each point and X.Location, which is the XYZ location of each point (and X is the cell index - see the sample code in `reading_points.m`. The original RGB or XYZ images are 512 rows by 424 columns by 3 channels but are encoded as a list of points(512*424,3). You can reshape one of the channel lists (*e.g.* the red channel) back to an image by `reshape(list(:,1), [512, 424])`. The colour image shown in the top left image above was formed by reshaping the X.Color array.

There are also 2 Matlab support functions:

`http://homepages.inf.ed.ac.uk/rbf/AVDATA/AV118DATA/imag2d.m`

`http://homepages.inf.ed.ac.uk/rbf/AVDATA/AV118DATA/reading_points.m`

The function `imag2d` displays a list of colour points as a colour image as in the top left image above. The function `reading_points` shows loading and displaying the set of point clouds.

Not all frames show the box - you can use these for background subtraction.

Because of specular reflections, not all points from the box surfaces are visible. You can ignore frames where there are not enough points to accurately estimate the planes (*e.g.* extracted planes should have normals that are approximately perpendicular).

## 3 Task Details

The overall task for this assignment is to fuse the individual XYZ point clouds to to create as much of a complete 3D model as possible. To do this you need to write a set of programs that can:

1. Extract the relevant data from each point cloud.
2. Extract the planes from each point cloud.
3. Estimate the 3D positions of the corners where planes meet.
4. Use the corners and planes to register the point clouds together and fuse the XYZ data from all images.
5. Fuse all of the points into a single 3D coordinate system.
6. Build as complete a 3D model as possible and characterise the surfaces.

Each of these is described in more detail below.

### 3.1 Extract the relevant data from each point cloud

One core problem is finding where the data from the box is in each video frame. Fortunately, each frame uses the same global coordinate system, so the box is always near position (-0.71,-0.30,0.81). You can eliminate points that are far from this point.

There are 2 sets of irrelevant data: 1) background datapoints and 2) hand datapoints. You can eliminate the background by thresholding the 3D point depth, and the background is further away.

Background subtraction might also remove some irrelevant points.

The hand datapoints can be eliminated by using the colour of the skin pixels in a thresholding operation. You might need to use normalised RGB or convert to HSV (and use the H : hue).

Also, all the box data is connected together, so this constraint might also help.

You could also remove data points that are not near to another point - these are probably noise.

See drill lab exercise 4 which demonstrates how to read, write, and interpret the point-cloud data and the relation between a point-cloud and its 2D image.

### 3.2 Extract the planes from each point cloud

See the course notes.

### 3.3 Estimate the 3D positions of the corners where planes meet

There might be 1, 2, or 3 planes visible depending on the viewpoint. If just one plane, then ignore this frame. If more than 1 frame is visible, then you can estimate the 3D positions of the corners where the planes intersect. These positions can be used in the next step.

### 3.4 Use the corners and planes to estimate the rotation and translation between two consecutive point clouds

You can use the 2 or 3 surface normals and the vector direction of the intersection line between 2 planes. Then use the rotation estimation algorithm from the course notes.

You will need to make sure that you have the correct corner points and vectors matched between the consecutive frames. You could do this manually, or you could search for different possible matches.

If you do this automatically, then you will need to verify the matches. You can use the coloured points from the 2 frames. If the registration is correct, then most of the nearby points from the 2 registered point clouds will have the same colour (maybe you need to use normalised RGB).

### 3.5 Fuse all of the points into a single 3D coordinate system

Given the reference frame transformation (rotation + translation) between each consecutive pair of frames, you can then map all of the points into the initial frame. Ie, if $T_{t,t+1}$ maps points in frame $t+1$ to frame $t$, then you can also map points from frame $t+2$ to frame $t$, because $T_{t,t+2} = T_{t,t+1} * T_{t+1,t+2}$. And so on, until you have all of the points from any frame $t$ to frame 1.

This means that you can get a model of all sides of the box (assuming that all sides were seen).

### 3.6 Evaluate the quality of the model as you add more frames

As each estimated transformation between consecutive frames has a little error due to image noise, there is potential for the errors to accumulate as more frames are added.

Show an image of the fused point cloud after 1, 2, 3, 4, 5, 10, 15, 20 ... 50 frames are added.

Pick one of the large box surface planes visible in the first frame (or first frame that has more than one plane visible). Record the unit surface normal of that plane and the corresponding surface normal of every time a new frame is fused with the same large plane. In theory all of these normals will point in the same direction, but noise and drift will cause some variation. After each new frame is fused, compute the average angle between all of the normals for that same box surface. That is, given a set of $N$ unit normals $\{\vec{n}_i\}_{i=1}^{N}$ compute the average of $cos^{-1}(\vec{n}_i \cdot \vec{n}_j)$ for all $j > i$ and $i = 1 : N-1$. Plot this average as the number of frames goes from 1 to 50.

Pick one of the corners visible in the first frame (or first frame that has more than one plane visible). Record the position of this corner every time it appears in a frame after it has been transformed into frame 1. Compute the average distance between the different instances of the same corner. That is, given a set of $N$ corner points $\{\vec{p}_i\}_{i=1}^{N}$ compute the average of $\| \vec{p}_i - \vec{p}_j \|$ for all $j > i$ and $i = 1 : N-1$. Plot this average as the number of frames goes from 1 to 50.

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, 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.

In the same manner as with the vectors and the corners, we want to know the RMS of all of the fused data points that belong to that same large frame. That is, suppose the large plane is seen in frames 1, 2, 4, 9, 15, ... . Then the data points for that plane in the fused frame 1 consist of only data points from the original frame 1. In fused frame 2 and 3, there are data points from original frames 1 and 2. In fused frame 4, data from original frames 1, 2 and 4 are fused. And so on. After each new fusion, compute again the RMS of the fused large plane fit, and plot this as a function of the input frame number.

Show some images of this fused plane, after fusing 1 view of the plane, 2 views, 3 views, and all views (there should be about 16).

## 3.7 Outputs

The are several images that should be included in your report:

1. Examples of box's complete fused point cloud after fusing several frames
2. Examples of box's large plane point cloud after fusing several frames
3. A plot of angular error *versus* number of frames fused.
4. A plot of corner position error *versus* number of frames fused.
5. A plot of plane fitting RMS error *versus* number of frames fused.

In addition to the images, include in your report a table, with 1 row for each of the 50 test images. Include in the table: { image name, number of 3D box pixels found, number of planes found, average plane fit RMS of the planes found in that frame }.

# 4 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 some original images and the detected box.
- Show some examples of any unsuccessful frame fusions.
- 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, but include a statement about what code and where it came from.

# 5 Other Comments

The assignment is estimated to take 20 hours coding/test and 5 hours report writing per person, resulting in a 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.

A single, joint, report is to be submitted. Split the work so that each partner can do 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% |

You can use the lecture example code from:
`http://www.inf.ed.ac.uk/teaching/courses/av/MATLAB/`

If you find a limit on the use of Matlab Image Processing Toolbox licenses, some alternative MATLAB functions are at:
`http://www.inf.ed.ac.uk/teaching/courses/av/MATLAB/UTILITIES/`

# 6 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 1 FILE.pdf
```

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

# 7 Live Demonstration

There will also be a demonstration session assigned between 9:00-13:00 on Friday March 23, where you will have to demonstrate your code on a new dataset. We'll email you about the location and schedule.

A new but similar dataset will be available for demonstrating your results. You will need your matlab program to show:

1. The original point cloud.
2. The extracted box points.
3. The surface normals for each of the main extracted planes. Draw on top of the same image as the extracted box points.
4. An image fusing this frame and the previous frame.
5. An image fusing this frame and all previous frames.
6. The angular, corner and RMS error plots as described above.

Put a pause after each frame.

# 8 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. Details about this can be found 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).

# 9 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.*

# 10 Late coursework policy

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