Advanced Vision 2011/2012
Practical 3

Stephen McGruer and Sean Wilson
s0840449@sms.ed.ac.uk, s0831408@sms.ed.ac.uk

Abstract
Modifying a video clip to have images projected on different surfaces can be a done using the depth data of a scene. Depth is an effective way to differentiate regions of an image. The Microsoft Kinect provides both depth and colour data which can be used to find planes for the areas of interest such as the background. These regions can be found using plane growing and edge detection algorithms. The accuracy of the Kinect depth data can be affected by environmental noise, and so must be cleaned to ensure an aesthetically pleasing final product.

Introduction
Background
This paper tackles the task of projecting images onto different planes in a video clip of a person walking across a screen. A background image is projected onto the wall behind the person, and a video clip is projected onto a quadrilateral he holds. The video clip was recorded using the Microsoft Kinect which provides both colour and range data.

The Kinect sensor uses an infrared laser projector to find depth data, projecting a laser beam for each pixel in the camera. The data from the Kinect is recorded as a 6-tuple of the *x* and *y* location of the pixel, the *z* range data and the RGB value. This paper describes the methods used to project images onto different extracted planes in a Kinect video clip.

Figure 1 shows the background image to be inserted and the region in which the image will be projected. The person in the video is also carrying a quadrilateral object. Using planar extraction and homographic transformation, another video clip will be projected onto this quadrilateral.

Figure 1: Left, range and colour image with projection planes. Right, background image

The Kinect is equipped with a single VGA camera which gives colour data for a 640x480 screen. For each of these pixels an (x, y, z) position is also obtained via triangulation of the two IR sensors.

However, due to reflection or refraction of the infra-red lasers, the data obtained for a particular pixel may be incorrect or completely missing. In both of these cases, the data must be cleaned before other algorithms are applied in order to present the best output image. An example of missing data can be seen in Figure 2.

Another problem occurs when the quadrilateral is not fully in the scene. When there are not four corners of the quadrilateral in view, it becomes difficult to identify the points necessary for the homographic transformation.

Figure 2: Left, the black border around the person occurs because of missing depth data as pictured on the right

Methods

Cleaning zero-depth data

Due to the issues discussed above, some pixels do not have depth data. A pixel with no depth data also has no corresponding RGB value, as can be seen in Figure 2. Such pixels have a *z*-value of zero and thus do not lie on either the person or the background. For most cases in the video, missing depth data is due to reflection or refraction around the edges of the person and the quadrilateral. Small areas of zero-depth data can be cleaned by comparison to the neighbouring pixels and the planar equation.

The algorithm used for cleaning zero-depth data iterates over each zero-depth pixel in the image, and splits the neighbouring pixels into three categories: those on the background plane, those not on the plane, and those with no depth data. It was found that if five or more neighbours had depth data, giving the pixel in question the average depth and colour value of the dominant neighbouring type produced satisfying results. As there are not always enough neighbours with depth data, the algorithm is applied iteratively until there are either no zero-depth pixels left or an infinite loop is detected.

Algorithm 1 (fix_pixels.m): Zero-depth data clean-up algorithm. Corrects a 'no-depth' pixel to have the average depth of its neighbours. Also sets the pixel's colour value as the average of the neighbours. on_plane_avg and off_plane_avg holds the average *z*, and RGB values.

```
num_on_plane, num_off_plane ← 0
on_plane_avg, off_plane_avg ← (0, 0, 0, 0)
for each non-zero depth neighbour
    if neighbour is on the plane
        num_on_plane += 1
        on_plane_average += neighbour's z and rgb
    else
        num_off_plane += 1
        off_plane_average += neighbour's z and rgb
on_plane_avg ← on_plane_avg / num_on_plane
off_plane_avg ← off_plane_avg / num_off_plane
if not enough neighbours were non-zero depth
    avg_z ← 0
    color ← current color
else if num_on_plane > num_off_plane
    avg_z ← on_plane_avg(1)
    color ← on_plane_avg(2:4)
else
    vg_z ← off_plane_avg(1)
    color ← off_plane_avg(2:4)
return avg_z, color
```

Homographic Transformation

A stationary camera and background allows for manual retrieval of the coordinates of the background plane corners. These coordinates are used to project the supplied image into the correct shape and angle to fit the plane. The homography method used to project the background image to the new position is described in [1].

Due to fluctuations in the brightness between frames of the input video, the homographised image at times appeared either too dark or too light for the scene, which was aesthetically displeasing. To correct this, in each frame both the scene and the homographised image were converted to the HSV colour space, and their mean brightnesses calculated. The homographised image's brightness was then adjusted according to the difference, and converted back into RGB space before being placed in the scene.

Extracting and Projecting the Background Plane

In order to project the homographised image onto the background plane without overlapping the person or the rest of the scene, each pixel must be categorised as on or off the background plane. The planar equation for the background was determined using the method described in [2], with a rectangular subsection of the background pixels used as input points.

Two checks are performed to determine if a pixel is on the background plane. Firstly, the pixel's (x, y, z) data is used to determine if it lies on the plane. If this check passes, another test is performed in image space. As the homographised image has zero RGB values for pixels that are not part of the

background, the corresponding pixel on the homographised image is checked. If the pixel has an RGB value of (0, 0, 0) it is not on the background and so is not included. This test is possible as there are no fully black pixels (zero for all colour values) in the image itself. If both checks pass, then the current pixel is part of the background and so the homographised image colour is used at that point.

Extracting the Plane and Boundary of the Quadrilateral

The extraction of the planar quadrilateral carried by the person was implemented using plane growing as described in [3]. To increase performance, the search space is confined to the lower half of the image, as the quadrilateral never leaves this space. As this process was performed offline before the cleaning described above, zero depth data is removed - it does not affect the output plane. To reduce the size of the starting-point search space, only pixels with a RGB value lower than (50,50,50) are considered as potential start points. The plane growing algorithm is then applied to the cleaned lower half of the image (which includes all pixels, regardless of colour.)

The algorithm selects, at random, a group of pixels and grows the region by including nearby pixels that are on the same plane. As the region grows the plane equation is recalculated, to give a better fit. This process continues until the plane can no longer grow, the fit error becomes too large, or an early termination heuristic occurs. The early termination heuristics include checks for the region growing too large or too tall, with limits derived from manual examination of the quadrilateral over all frames.

Once the plane finishes growing, the resulting region is checked to see if its size is acceptable for the quadrilateral. If the region is an acceptable fit, the final planar equation is calculated (if not, the entire process repeats.) This equation is then used to select all matching pixels from the entire image, in order to collect pixels not included in the plane growing algorithm. The largest connected region of these pixels is then taken to be the quadrilateral, removing spurious data points that are also on the same plane.

Overlay of New Video on the Quadrilateral

It was decided that the video clip overlaid on the quadrilateral would be that of the video itself.For simplicity, this was done by using the previous frame as the source image. In order to perform the homographic projection, the corner coordinates of the quadrilateral had to be found.

Canny edge detection [4] is applied to the quadrilateral found in the previous step to isolate the edge pixels. The equations of the sides are determined by applying the RANSAC algorithm [5] four times, removing the detected pixels each time. The four intersection points are then calculated by splitting the lines into parallel pairs and finding the cross-pair intersections, which are the corners of the quadrilateral. These are the target points for the homographic projection.

In several frames the quadrilateral is only partially visible, with less than four corners visible. This affects the location of the quadrilateral plane (as the visible area is smaller) and also the RANSAC line detection. In the case of three corner points (one frame), the algorithm was still able to successfully find the line equations, and performed the projection correctly. In the frames with only two points visible (five frames), the location of the other two points were found by hand. Given two visible points (x_1, y_1) and (x_2, y_2), the gradient of the known edge is found using (1), and the perpendicular gradient calculated using (2).

$$m = (y_2 - y_1) \div (x_2 - x_1) \quad (1)$$

$$m' = \frac{-1}{m} \quad (2)$$

Using this gradient it is possible to project along each of the two edges to the non-visible points. By substituting x_1 and the edge distance *d* into (3), the x position of one hidden corner can be found. This x, along with (x_1, y_1) and the perpendicular gradient *m'*, is then substituted into (4) to obtain the y value of the corner. The application of (3) and (4) can then be repeated with (x_2, y_2) to find the final corner, and the projection performed as before.

$$x = x0 \pm (d \div \sqrt{1 + m'^2}) \quad (3)$$

$$y = y_1 + m'(x - x_1) \quad (4)$$

It was determined that there was not enough of the quadrilateral visible in the frame with only one corner for it to be possible to find the other corners, and so that frame was skipped.

Reconstructing the Video

Once the background and quadrilateral images are projected onto each frame and they have been cleaned, the frames can be combined into the .avi video format using the Matlab 'VideoWriter' class and the 'writeVideo' function.

Due to the jagged edges produced by some of the methods used, a Gaussian blur is applied over each frame before they are placed into the video. This smooths out some of the more sharp edges on the person, background, and quadrilateral.

Discussion

The background image was successfully projected onto the scene without overlapping the person's body or the quadrilateral in any of the frames. However, it was found that some of the background pixels surrounding the person and the quadrilateral had the same depth data as the foreground. This results in a layer of pixels around the person that are not modified to the new background RGB values, as seen in Figure 3. This layer of pixels are not easily corrected. Colour space isolation was not successful due to the similar colouring between parts of the t-shirt and the background. A possible solution that was not tried would be to apply depth-blurring. This may fix the boundary problem but may also make planes harder to distinguish.

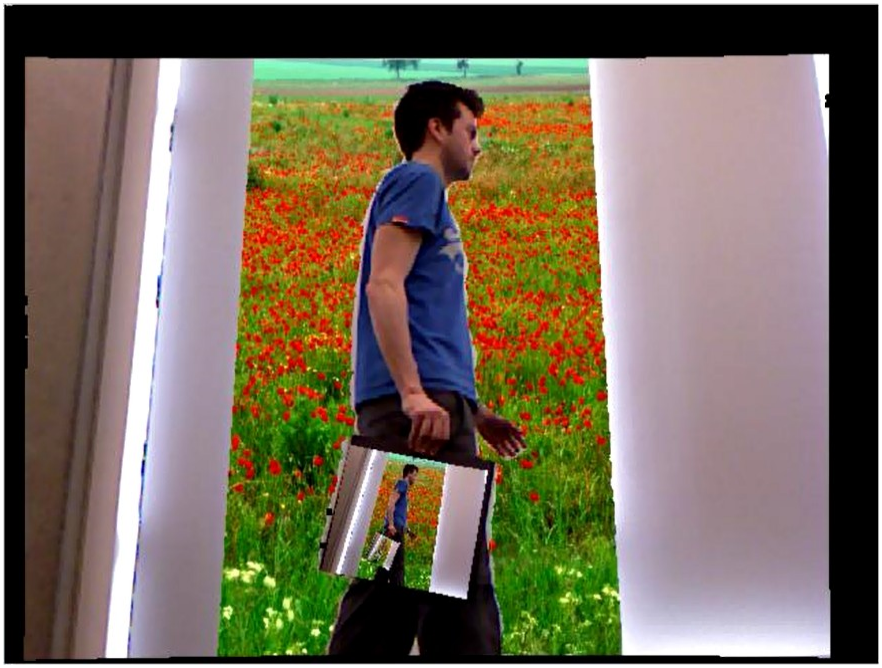

Figure 3: A layer of original background colour can be seen around the person

The cleaning algorithms worked well, both in correcting the zero-depth data and the background image brightness. As can be seen in Figures 3 and 4, the zero-depth data has almost all been corrected, with only a few fragments of zero-depth data around the edge of the quadrilateral. The algorithm could be extended by reducing the number of required non-zero neighbours when the algorithm enters an infinite loop. This will ensure that all zero-depth pixels are restored, but will not guarantee accurate values. The effect of the brightness correction can be seen in Figure 4.

Figure 4: Shows the brightness changes in the background image to match the scene

An image was successfully placed onto the quadrilateral automatically in the frames where the quadrilateral has four clear corners (frames 14 - 28). In frames 8 - 13 and frame 29, the quadrilateral is partially visible. Using the manual mathematic equations described above, the image was successfully

projected onto the visible part of the quadrilateral as seen in Figure 4. This would need to be automated for this program to work on a different video clip.

Similarly, the background image is projected using manually calculated projection points. This was done to speed up the production process of the final video clip. However, this process could be automated using the same plane growing methods used for the quadrilateral. This would ensure the background image is correctly applied even if the camera angle is changed during the video clip.

The plane growing algorithm at times took several attempts before finding a patch that was on the quadrilateral. Although the search space is constrained to the lower half of the image and those pixels under the threshold colour, there are still a lot of potential starting points for the algorithm, and due to the random nature it could potentially take infinite time. A useful heuristic would be to avoid starting a patch in the same area multiple times in a row.

Conclusion

The output video produced in this paper is an example of how videos can be modified when supplied with the range data that accompanies the colour data of the frames within the video. We have shown that stationary and moving planes can both be located automatically. Multiple methods of cleaning were implemented in order to improve the aesthetics of the output video, including zero-depth pixel correction, wrong-depth pixel correction, Gaussian smoothing and illumination correction. These methods cleaned the image and produced a more realistic final product, but unless all pixels have the correct range data there will always be a few incorrectly coloured pixels in the final product.

References

1. B. Fisher. (2012). *IVR Matlab Code* [Online]. Available FTP: inf.ed.ac.uk Directory: /teaching/courses/ivr/MATLAB/FLATPARTRECOG/ File: esthomog.m
2. B. Fisher. (2012). *3D Object Recognition from Range Data* [Online]. Available: http://www.inf.ed.ac.uk/teaching/courses/av/LECTURE_NOTES/SMALL/avsys6s.pdf, pp. 24
3. B. Fisher. (2012). *3D Object Recognition from Range Data* [Online]. Available: http://www.inf.ed.ac.uk/teaching/courses/av/LECTURE_NOTES/SMALL/avsys6s.pdf, pp. 23
4. B. Fisher. (2012). *3D Object Recognition from Stereo Image Data* [Online]. Available: http://www.inf.ed.ac.uk/teaching/courses/av/LECTURE_NOTES/SMALL/avsys5s.pdf, pp. 9 - 20
5. B. Fisher. (2012). *3D Object Recognition from Stereo Image Data* [Online]. Available: http://www.inf.ed.ac.uk/teaching/courses/av/LECTURE_NOTES/SMALL/avsys5s.pdf, pp. 22 - 27