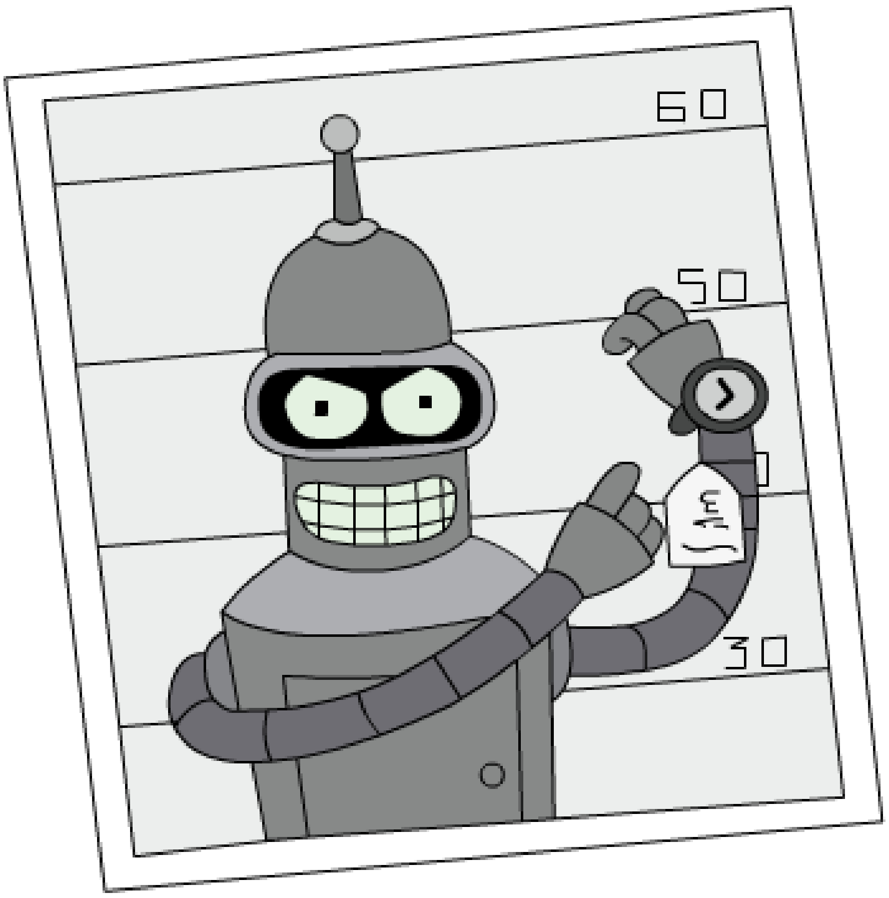
Image stitching by homography

ECE4076/5176 Computer Vision
Lab 2 (Weeks 5,6)



Academic integrity

Every lab submission will be screened for any collusion and/or plagiarism. Breaches of academic integrity will be investigated thoroughly and may result in a zero for the assessment along with interviews with the plagiarism officers at Monash University.

Late submissions

Late submission of the lab will incur a penalty of 10% for each day late. That is, with one day delay, the maximum mark you can get from the assignment is 7.2 out of 8, so if you score 7.5, we will (sadly) give you 7.2. Labs submitted with more than a week delay will not be assessed. Please apply for special consideration for late submission as soon as possible (*e.g.*, documented serious illness).

Lab Instructions and the Use of Generative AI

- You may **not** use any built-in opencv functions for this lab, other than those used for loading/ saving an image, extracting and matching keypoints, and computing homographies.
- You may use NumPy for array handling, and vectorizing your code (reducing the number of for-loops) is encouraged.
- You should use Matplotlib to display images and any intermediate results.
- You may use generative AI unless explicitly prohibited.

Lab Grading

Each lab is worth 8%, and there are a number of sections and tasks with their own weighting. A task is only considered complete if you can demonstrate a working program and show an understanding of the underlying concepts. Note that later tasks should reuse code from earlier tasks.

This lab is about stitching two images with known homography into a single wide-angle image using bilinear interpolation. There will be an obvious stitch between the two images, so blending techniques will be used to improve on the final result. The following are the five tasks that you should deliver. Collectively, tasks 1-5 will provide a final stitched image. In task 6, you will apply the full process from tasks 1-5 on your own two images to create a panorama image.

- Task 1: Draw test points on the left image
- Task 2: Use Homography to find right image points
- Task 3: Bilinear interpolation of the right image
- Task 4: Image stitching
- Task 5: Better blending
- Task 6: Now try your own!

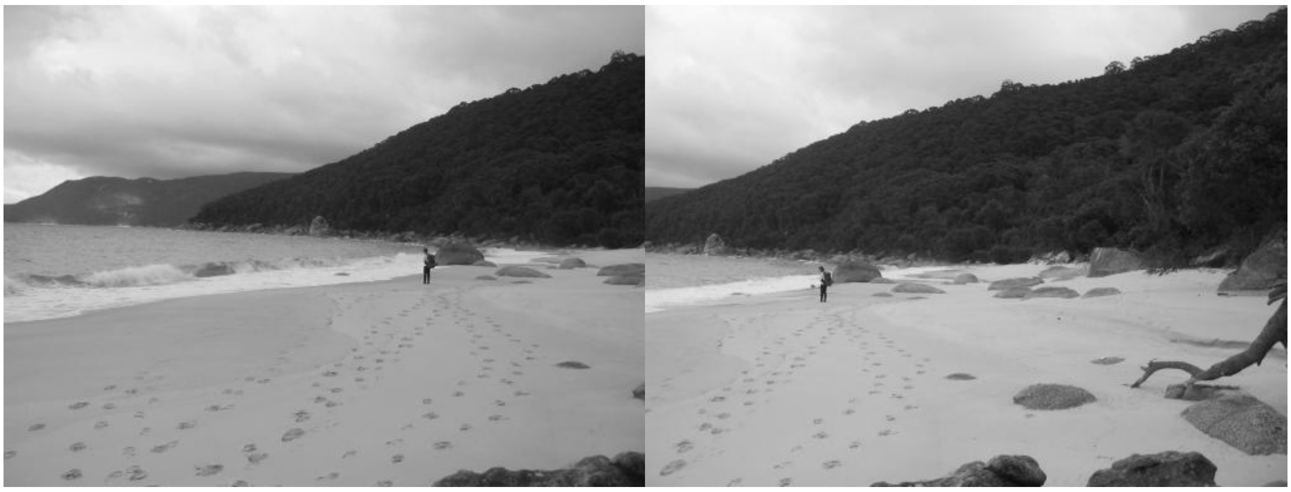


Figure 1: Images to be stitched together

- Discussion questions: These are at the end of each task within the notebook, and require you to answer them when submitting your python notebook. There is no discussion question for task 1 in this lab.

References

- Image stitching
- Bilinear interpolation

Resources

- Lab2_student_template.ipynb - please complete the lab tasks in this template notebook and use the markdown spaces provided to report your findings/ describe your approach.
- left.jpg and right.jpg are located in the lab2 folder
- Keypoint detection and homography calculation in week 4's interactive lectures

Task 1. Draw test points on the left image

Draw the following points onto the left image as red crosses. Display the resulting image.

{337,196,1}, {467,289,1}, {252,169,1}, {262,255,1}, {241,135,1}

Recall from lectures that these 3-element homogeneous coordinates can be transformed to 2D image pixel coordinates by dividing the first and second elements by the third (needed for later tasks).

Task 2. Use Homography to find right image points

The following homography transforms pixel coordinates between the left and right images as

$$\mathbf{x_R} = \mathbf{H} * \mathbf{x_L}$$

$$H = \begin{bmatrix} 1.6010 & -0.0300 & -317.9341 \\ 0.1279 & 1.5325 & -22.5847 \\ 0.0007 & 0 & 1.2865 \end{bmatrix}$$

Apply the homography to transform the left image points in Task 1 to the corresponding locations in the right image. Draw the transformed points as red crosses. Check your result before moving on.

Task 3. Bilinear interpolation of the right image

The transformed coordinates via homography can be in between pixel locations. Write a bilinear interpolation function to compute the intensity of the transformed pixel coordinate in right.jpg using intensity values from neighbouring pixel locations. Print the interpolated intensity value for each transformed point in Task 2. The first point should be around 68 whereas the last point should be around 59.

HINT: The bilinear interpolation function should take the transformed pixel coordinate and the intensity values of its four neighbours as input arguments, and should output the interpolated intensity value.

Task 4. Image stitching

Create a 1024x384 (width x height) image and fill the LHS with the left image. This stitched image will use the left image coordinate system (xl) throughout the stitching process.

Next, fill in the remaining 512x384 pixels on the RHS by transforming their pixel coordinates (left image coordinates) to the right image coordinates via the homography from Task 2. Sample the right image to fill in the missing parts of the stitched image pixel-by-pixel as follows: 1) If the right pixel coordinate is valid, generate the pixel value using bilinear interpolation 2) If the right pixel coordinate is invalid, use a pixel value of zero

Display the stitching results. It should look like a wide-angle image with a visible seam where the two images join.

Task 5. Better blending

Improve the visual quality of the stitched image by trying the following image processing techniques:

1. Adjust the width of the output image automatically so that less black pixels are visible
2. Adjust the brightness (by a scaling factor) of each image so that the seam is less visible
3. Apply a small amount of Gaussian blur or alpha blending near the seam to make it less visible
4. Adjust the horizontal location of the seam (it can be moved further to the left as the right image overlaps into the left by quite a few pixels)

Note that you do not have to try all of the above. However, you will only receive a mark here depending on

- the quality of the stitched image
- whether a serious programming attempt is made to improve the stitched image

Task 6. Now try your own!

In this final task, you will:

1. Take two images from different perspective of the same scenery and display it
2. Find and match key points across the two images
3. Calculate the homography matrix[1]. Print out the homography matrix that you end up using.
4. Apply image stitching for a final smooth image (from tasks 1 to 5)

Discussion Questions

These are discussion questions that need to be answered when submitting your python notebook.

[1]Hint: Steps 2-3 in this final task can be done with OpenCV functions. You can see an example in the week 4's workshops.