TSBB09 Laboratory Assignment: Panorama Stitching

Developed by Per-Erik Forssén, June 2008

Computer Vision Laboratory, Linköping University, Sweden

Last update: Maria Magnusson, November 2021

Contents

1 Introduction

Image stitching is the process of combining several images with overlapping content into a single, big image. Image stitching is used in aerial and satellite imaging to generate *ortho images* as used in e.g. web-based map services such as `maps.google.com`, `eniro.se` and `hitta.se`. Another popular application is construction of panoramas from pictures taken with consumer digital cameras [1]. This is what the present assignment is all about.

1.1 Preliminaries

Before coming to the computer session it is necessary to read through course material on panorama stitching (the document [2]), as well as this document.

The slides from the lectures on projective geometry and camera calibration are also useful. In this document you will also find a number of home exercises to be answered before the session. They are all clearly marked with a pointing finger.

Extra

Implementation exercises marked with an "Extra" box may be temporarily skipped and completed when the other exercises are finished.

Examples of MATLAB commands frequently used are `imshow`, `meshgrid`, `svd`, `inv`, `eig`, and `angle`. If you are unfamiliar with any of these commands, use the MATLAB `help` function to find out how they work.

This lab assignment also features the following extra functions:

- `correspondences_select`
- `homography_stls`
- `image_lensdist_inv`
- `image_resample`
- `image_resample_sphere`

Use `help <command>`, and optionally `type <command>` to find out how they work.

2 Image Rectification

Start MATLAB and add the path to the `panorama` functions:

```
>> l5path = '/courses/TSBB09/Panorama/'
>> addpath([l5path 'util/'])
```

Select one of the image sets `[l5path 'images1/']` or `[l5path 'images2/']`.

QUESTION 1: These panorama images were obtained through a total of 360° horizontal rotation of the camera. As you can see, there are 24 images in each set. What is probably the angular difference between two consequtive images?

Load an image, and try to rectify it using `image_lensdist_inv`. It is recommended that you use the `'atan'` distortion model. If the image is described in polar coordinates (r, θ), then r is modified according to

$$r_{\text{out}} = \frac{\arctan(r_{\text{in}} \cdot \gamma)}{\gamma}$$

by the `'atan'` rectification.

QUESTION 2: How can a good value for the γ parameter be found? Hint: Start with $\gamma = 0.001$ and make subplots for different values of γ.

---

3 Uncalibrated Stitching

Now we will load two consecutive images from the panorama set, and try to blend them. In order to minimise the amount of typing you need to do it is recommended that you write a MATLAB script with cells (divide the program by lines starting with %%), and evaluate parts of the script as needed.

First load two images and rectify them as above. Then use the function `correspondences_select` to manually indicate a set of corresponding points. Tip: Try `help correspondences_select`. To avoid having to select correspondences again, make sure you save them to file.

```
>> [x1,x2]=correspondences_select(img1,img2);
>> save corresp_1to2.mat x1 x2
```

QUESTION 3: What is the minimal number of point correspondences needed to estimate a homography between the two views? Does adding more points improve the result? If so why?

Now use the function `homography_stls` to estimate a homography $\mathbf{H}_{12}$ from image 1 to image 2.

```
>> H12=homography_stls(x1,x2);
```

Suppose that `x1` is a list of four marked points in the first image,

$$x1 = \begin{bmatrix} x1a & x1b & x1c & x1d \\ y1a & y1b & y1c & y1d \end{bmatrix},$$

then `[x1;ones(1,size(x1,2))]` gives a matrix

$$\begin{bmatrix} x1a & x1b & x1c & x1d \\ y1a & y1b & y1c & y1d \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

that can be mapped through $\mathbf{H}_{12}$.

QUESTION 4: Write a function `map_points.m` that maps a list of coordinates, such as `x1` through a homography, i.e. `x2 = map_points(H,x1);`. **Note:** Avoid using for-loops in the function. `map_points.m` will later be called by `image_resample.m` and for-loops slow down the computation.

QUESTION 5: How can you use your function to verify that the homography is correct?

QUESTION 6: Does it matter which points you use to estimate the homography? How should they preferably be located in the image?

Now, define a list of coordinates for the four corners of the image:

```
>> [rows,cols,ndim]=size(img1);
>> imbox=[1 cols cols 1 1;1 1 rows rows 1];
```

QUESTION 7: Why is it a list of 5 points, do you think?

Map these points through $\mathbf{H}_{12}$, giving a new coordinate list `imbox12`, and plot this on top of image 2:

```
>> imbox12 = map_points(H12,imbox);
>> figure(1);imshow(img2); hold on
>> plot(imbox12(1,:),imbox12(2,:),'g')
>> axis image
```

QUESTION 8: Does the result look as expected? Based on how the images looked, can you say whether the result looks reasonable?
Hint: As previously mentioned, our panorama images have been obtained through a purely horizontal rotation of the camera.

QUESTION 9: How can we also find the image box from image 2 to image 1?

Now, we will use the function `image_resample` to resample the two images to a common registration grid. First we need to find the required extents of this grid. This can be done using the coordinates in `imbox` and `imbox12`.

```
>> cl=[imbox imbox12];
>> cmin=floor(min(cl,[],2));
>> cmax=ceil(max(cl,[],2));
>> rows=cmax(2)-cmin(2);
>> cols=cmax(1)-cmin(1);
```

Now generate a homography that moves the images into this reference grid:

```
>> Ht=[1 0 1-cmin(1);0 1 1-cmin(2);0 0 1];
```

Finally, we can resample the two images to the reference grid:

```
>> pano1=image_resample(img1,Ht*H12,rows,cols);
>> pano2=image_resample(img2,Ht,rows,cols);
```

QUESTION 10: Look at the two resampled images (using e.g. `imshow`) and verify that they appear to fit before proceeding.

In order to blend the two images, we will generate weight masks in the reference grid and combine the images using weighted averaging.

```
>> alpha0=ones(size(img1(:,:,1)),'uint8')*255;
>> alpha1=image_resample(alpha0,Ht*H12,rows,cols);
>> alpha2=image_resample(alpha0,Ht,rows,cols);
```

Consequently, `alpha1` is the mask for the first image in the reference grid and `alpha2` is the mask for the second image in the reference grid.

Finally we combine the images using weighted averaging:

```
>> pano=zeros(rows,cols,3,'int32');
>> asum=int32(alpha1)+int32(alpha2);
>> asum(asum==0)=1;
>> for k=1:3,
>>  pano(:,:,k)=pano(:,:,k)+int32(pano1(:,:,k)).*int32(alpha1);
>>  pano(:,:,k)=pano(:,:,k)+int32(pano2(:,:,k)).*int32(alpha2);
>>  pano(:,:,k)=pano(:,:,k)./asum;
>> end
```

QUESTION 11: Why do we divide with asum and why do we need the row `asum(asum==0)=1`?

QUESTION 12: Look at the resulting panorama `pano` using e.g. `imshow` (cast to `uint8` first or scale to a maximum of 1). Describe the result.

Now tidy up your script that performs all the steps in section 3 and **show the result to the teaching assistant before proceeding further.**

QUESTION 13: Was the teacher content with your planar panorama?

Extra

Extra exercise In order to get a better blending of the images, the weight mask `alpha0` should be replaced with a mask that linearly decays from the centre of the mask, from 255 to 1. Write down how to make such a mask below:

Recompute the panorama using your new blending function and verify that the result looks better.

Extra **Extra exercise** Modify your script such that it also blends a **third** image into the panorama. Did you manage?

---

4 Stitching in Spherical Coordinates

The images used in this lab have been grabbed with a camera that has the following calibration matrix:

$$\mathbf{K} = \begin{pmatrix} 1420 & -3 & 808 \\ 0 & 1420 & 605 \\ 0 & 0 & 1 \end{pmatrix} . \tag{1}$$

Using this prior information, we can resample the images to a spherical coordinate system.

QUESTION 14: First extract the horizontal and vertical fields of view from $\mathbf{K}$, as well as the offset between the geometrical centre and the projection of the optical centre. Write down the measures in radians and degrees.

These can be used to define a sampling range in spherical coordinates. For instance you may try the ranges:

```
>> hr=hfov/2*[-1.8 1.8]+hoff;
>> vr=vfov/2*[-1.1 1.1]+voff;
```

You will also need to specify a sample density on the sphere. Try starting with 0.05°.

These ranges will give you a little extra space above and below the vertical FOV and rather much space to the left and the right of the horizontal FOV, i.e. one additional image may be placed to the left and another to the right.

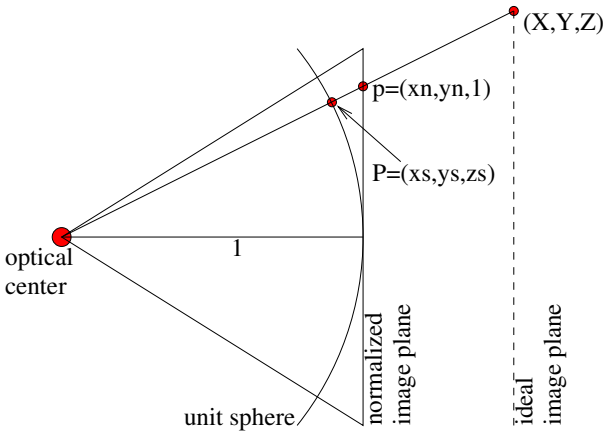

Figure 1: Camera geometry with extra unit sphere.

See Figure 1 that shows the camera geometry. Also, the normalized image plane as well as the unit sphere are indicated in the figure. A point $(X, Y, Z)^T$ can be projected to the normalized image plane $(x_n, y_n, 1)^T$ and then transformed to a point $(x, y, 1^T)$ on the real image grid through the **K**-matrix according to

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} . \tag{2}$$

QUESTION 15: Load the correspondences, `x1` and `x2`, that you selected for an image pair from file. Then transform these to coordinates on the normalized image plane, `p1` and `p2`, using your function `map_points.m`. How is this done?

QUESTION 16: See Figure 1. Give an equation how to calculate a 3D point on the unit sphere (x_s, y_s, z_s) from a point on the normalized image plane $(x_n, y_n, 1)$.

QUESTION 17: Write a function `procrustes.m` that finds the best rotation matrix that transforms the mapped points `p1`, to `p2`.
Hint: First convert to 3D points on the unit sphere, `P1`, and `P2`. Then use `[U,D,V]=svd(P2*P1');` and finally `R=...;`.

Use your `procrustes.m` function to find the best rotation matrix between the two point lists.

QUESTION 18: Show the teacher that the norm of all your `P1` and `P2` points are 1. (If you use the command `norm`, you must be careful here.) Was the teacher content?

QUESTION 19: Extract the rotation axis and angles from your rotation matrices. What values do you get? Are they reasonable? Explain why!
Hint: Use the command `[V D] = eig(R);`

Once we have $\mathbf{K}$ and all rotation matrices $\mathbf{R}_{kl}$ between the view pairs in the panorama, we can blend all images in the spherical coordinate system. To do this, you need to use the function `image_resample_sphere` instead of `image_resample`. Now write a MATLAB script that does this for three images. Choose image i to put in the center, then rotate image $i-1$ and put to the left and finally rotate image $i+1$ and put to the right.

QUESTION 20: Which panorama, planar or spherical, is best to use for a wide field-of-view (FOV), e.g. $[-45°, 45°]$? Which panorama, planar or spherical, is best to use for a full 360° FOV?

QUESTION 21: Was the teacher content with your spherical panorama?

When we have filled the whole sphere with a panorama, we are free to project a patch of the sphere to a normal flat image in an arbitrary direction. Visit `https://www.google.se/maps/` and search for:
Västerlånggatan 22, Stockholm
Move the yellow person in the lower right corner to the address. Localize the shop "Mats Jonasson Målerås".

QUESTION 22: Turn around (pan) 180°. A picture of the face of a historical person has been blurred - who?
Then, look up (tilt) 90°. How was the weather that day?

QUESTION 23: Also visit `http://360gigapixels.com/petrin-prague-photo`
Which text can you read if you look down, i.e. a tilt angle of −90°?

QUESTION 24: Explore the resolution by using the zoom function. There are numerous statues along one bridge. This is the famous Karlsbron (Charles bridge). Are there more or less than 8 statues?

Extra

Extra exercise Expand your panorama by blending in more images until you get bored. Also try to generate panoramas from other image sets.

5 Final Notes

Read this if you are interested in implementing a full panorama generating program yourself. If this sounds interesting you can do this as a project in *Images and Graphics, Project Course CDIO,* TSBB11.

- In this assignment we used manually selected correspondences to compute the homographies relating the views. In computer vision applications, this is done automatically, using automatic correspondence finding algorithms like SIFT. See [1] for details.
- For best results, the images should be aligned using a joint optimisation over the calibration parameters $\mathbf{K}$ and γ, and the rotation matrices $\mathbf{R}_{kl}$. This is called *bundle adjustment* (BA) and is typically done using non-linear least-squares (e.g. `lsqnonlin` in MATLAB can be used to do this). You also need to use a smooth and compact representation of the rotations, e.g. *truncated quaternions* [3]. See [1] for details on how to set up the BA problem.
- The blending we used in this assignment causes image blur when the alignment is not perfect. Blending can be done in a more advanced fashion, using Laplacian pyramids[1], and in an even more advanced fashion using Energy Minimisation techniques [4].

References

[1] Matthew Brown and David Lowe. Recognising panoramas. In *IEEE International Conference on Computer Vision*, 2003.

[2] Per-Erik Forssén. Panorama stitching: Supplementary notes. Technical report, Linköping University, 2008.

[3] M.I.A. Lourakis and A.A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the Levenberg-Marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Aug. 2004.

[4] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6), June 2008.