Version: January 27, 2016

Linköping University

Name of student ______________________

Personal number ______________________

Date ______________________

Teaching assistant ______________________

The Harris Operator and the LK Tracker

Lab Exercise 1

1 Introduction

During this exercise, you will implement the original version of the widely used Lucas-Kanade Feature Tracker (LK Tracker). The LK tracking algorithm is computationally heavy and a Matlab implementation is likely to be slow if we try to track a region around every pixel in the image. In this computer exercise we will make the restriction to track only one single pixel, i.e. only one region centered around one pixel. This means that we need to choose which pixel to track. Due to the close relationship between the LK tracker and the Harris operator we will use the Harris operator to detect good features to track.

1.1 Preparations

Before the exercise you should have read through this exercise guide and completed the home exercises. They are all clearly marked with a pointing finger. To be able to do this you should have read, and understood, the simplified paper on LK tracking that can be found at the course homepage. When we are asking questions on a specific expression or variable, e.g. $\mathbf{Z}$, we are using the same notation as in the simplified paper supplied together with this document [1]. It is important to have an understanding of the different steps in the algorithm to be able to complete the computer exercises in time.
Expected preparation time: 4h

1.2 Start Matlab

The first thing we need to do in Matlab is to add the course to the path. This needs to be done each time you start Matlab.

```
>> initcourse TSBB15
```

2 Algorithm Outline

It is important to have an understanding of the different steps in the algorithm to be able to complete the computer exercises in time. Before coming to this computer exercise you should have a clear strategy on how to approach the problem.

Preparation Question: Draw a block diagram that outlines the main steps in the algorithm:

1. dtot = 0;
2. Lös Z och E för bilden J och gradienten. Uppdatera d. (från Zd=e)
3. if d ≈ small_value
    goto next_level_of_pyramid_and_interpolate;
  else create_new_image_and_grad_J;
4. goto 2.

Preparation Question: Write the final equation that is needed to update the displacement:

Zt_bar = b_bar. Lucas-Kanade

Preparation Question: Using the notation in [1], explain what **Z** is:

Z is the structure tensor.

Preparation Question: What is required of **Z**, for the final equation to be solved?

Full rank (= same amount of eigenvalues as rows)
Object can't move out of screen.
Object be intrinsically 2dimensional.

The LK tracking algorithm is computationally heavy and a Matlab implementation is likely to be slow if we try to track every pixel in the image. In this computer exercise we will make the restriction to only a single pixel. This means that we need to choose one pixel to track.

Preparation Question: Given your knowledge about the aperture problem and the answer to the question above, what should characterize a pixel we choose to track?

Distinkt mot bakgrunden. Ska ligga i rätt område.

3 The LK Tracker

Here you will implement the blocks needed to create your LK tracker.

3.1 Gradient Function

To be able to estimate the orientation tensor needed by the LK tracker and the Harris operator we need a function that estimates the horizontal and vertical derivatives for all pixels. Implement a function that takes a filter size and a standard deviation as inputs, and returns the regularized derivatives for an image. **Hint:** This was discussed in the computer lesson.

3.2 Estimating Z

You will need to implement a function that estimates an orientation tensor for a specified region. Make this function as general as possible, i.e. make it possible to use different window sizes and also non-square windows.

3.3 Difference Function

To update the displacement with the LK equations we need to estimate **e** for a specified region. Implement a function that does this. Make this function as general as possible, i.e. make it possible to use different window sizes and also non-square windows.

3.4 Interpolation Function

During the gradient descent iterations, it is apparent that we need to obtain intensity values for non-integer pixel values. You need to implement a function to access these values. Create a function that returns the interpolated intensity values for all sub-pixels specified by a region of interest. You might want to implement more than one interpolating function but that is not required to pass the computer exercise. Depending on your implementation, your region of interest might be the whole image. **Hint:** See the Matlab function `interp2`.

3.5 Finalizing the LK Tracker

Now when you have all the building blocks (the gradient function, the orientation tensor function, the difference function and the interpolating function) you should be able to finalize the LK tracker. What you basically need is to solve the displacement equation and a control loop that updates the displacement vector until some stopping criterion is met, e.g. small enough error or maximum number of iterations.

Check that your implementation is working correctly. To do this, use the function `getCameraman.m` to obtain **I**,**J** and **dTrue**. **I** is the original image, **J** is a shifted version of the same image, and **dTrue** is a vector describing the displacement between the images. Track a region with [height,width]=[70,40] centered around [row,col]=[85,120]. When testing your implementation your estimated displacement **d** should come close to **dTrue** even after the first iteration. Make sure that the estimated displacement is improved when using several iterations.

Question: What are the estimated displacements after the first and second iteration? What is the true displacement?

4 The Harris Operator

For the previous exercise we manually selected a region to track, but we would like to be able to automatically detect good features to track. We will use the Harris operator for this due to the close relationship between the Harris Operator and the LK tracker. Look at the answer to the previous preparation question. You should have come to the conclusion that good features to track are basically the same features that we can detect by using the Harris operator.
Preparation Question: Write down the expression used to calculate the Harris response.

☞

You will implement a function to estimate the Harris response for each pixel in the image. We will later use this function to extract a good feature to track with our LK tracker.

4.1 Orientation Tensor

Implement a function that returns the orientation tensor for all pixels in an image. **Hint:** This was discussed in lesson 2.

4.2 Implementation of the Harris Operator

Now when we have the orientation tensor you should be able to estimate the Harris response for all pixels. Implement a function that estimates the Harris response for all pixels in an image. Test this function on the image `cornertest.png`. By choosing a threshold for the Harris response you should be able to detect good features to track.

Question: When raising the threshold for the Harris response, you will end up with some distinct features. Are these the expected features?

You may end up with clusters of several neigbouring points above the threshold. Such clusters can be removed by 2D non-max suppression. **Hint:** In Matlab, the function `ordfilt2` can do this for you, e.g:

```
>> Imm = ordfilt2(Im,9,ones(3)); [row,col] = find(Im==Imm);
```

5 Combining LK and the Harris Operator

Now when you are sure that the LK tracker and the Harris operator are working, it is time to combine them. Use the Harris operator to find the best K features in `chessboard_1.png` that should be tracked in the subsequent frames `chessboard_2,chessboard_3,...` Avoid tracking features near the border of the image, since these might cause trouble.

Question: Show your lab assistant when you got a working implementation of this. Track the best $K = 5$ features and make sure that you are displaying the result obtained between each frame.

6 Regularised LK

Extra

This exercise should be done if there is time left.
You may have noticed that the LK tracker can get stuck in local minima. To get a more robust tracker we can perform the tracking using gradually reduced regularisation. That is, we start by using a large standard deviation while extracting the gradients from the image. By reducing the standard deviation each time our LK tracker has converged, we might be able to avoid local minima. Try this with $\sigma = 4, 3, 2, 1$.

Question: Did this improve the performance?

References

[1] Björn Johansson. Derivation of the Lucas-Kanade tracker. 2007.