Master MIR – Planar Homography Tracking

Yosef GUEVARA

April 17, 2023

General objective

The aim of this practical is to get some hands-on experience with a direct planar homography tracking algorithm to track planar patches across sequences of images. This could be useful for stabilising a AUV with respect to its environment for tasks like underwater inspection.

The code base is provided and you will be required to complete some core functions for the planar Homography-tracking algorithm. This will allows to sequentially determine the location of a planar patch throughout an image sequence via a non-linear iterative minimisation procedure.

The direct planar tracking algorithm functions as follows:

1. Select a patch in the first image as a reference patch using a mouse.
2. Initialise the first estimate of the Homography $H = I$ (identity).
3. Warp the current image using the current estimate of H.
4. Compute the error between the reference patch and the warped patch.
5. Compute the update of H according to the pseudo inverse of the Jacobian times the error.
6. If the update is still large, then repeat to 3.

Code and Sequence:

Obtain a copy of the *Cyclopes* source code and sequences. Two sequences are provided :

1. color monocular underwater sequence found in the directory : $IMAGES_small RGB$
2. A greyscale stereo sequence found in the directory : $Versailles_canyon$.

The main file to execude the code is:

cyclopes/examples/TrackImageSL3/mainTrackImageSL3.m

Inside this file is a function test() at the end of the file which contains all the parameters for this program. Modify the paths of the sequence and the source code to suit your installation.

Questions

1. At the core of a planar Homography tracker is the geometric warping function:

$$p_2 = Hp_1$$

The aim of this section is to produce the necessary computations to obtain the homography H for a set of points p_1, to find the transformation p_2. The $WarpSL3$ function, the points' matrix, is created by taking the U, V coordinates of the points to be warped from the reference image and adding a column of ones. Then, the points' matrix is multiplied with the homography matrix H to get the warped points in a matrix called $points_warped$ to implement a planar homography transformation.

Where p_1 refers to the set of points to be warped, in this case **u** and **v**, **ReferenceImage.P.U** y **ReferenceImage.P.V**, respectivamente. Lo cuales representan las coordenadas de la imagen. mienstras que w presenta la coordenada homogena, the purpose of the homogeneous coordinate is to represent the position of the pixel on a plane at a specific distance from the camera. The matrix **H** is used to compute the new homogeneous coordinates **(u, v, w)** of each pixel in the transformed image, where u and v are the transformed image coordinates and w is the homogeneous coordinate.

Listing 1: **u** and **v** coordinate transformation using **H**

```
1     % TO DO
2     % 1. Warp patch coordinates (u, v, w);
3     % Small u,v for indexed vector form. Capital U,V for Matrix/Image
      form.
4     u = (H(1,1)*ReferenceImage.P.U(ReferenceImage.index)+H(1,2)*
      ReferenceImage.P.V(ReferenceImage.index)+H(1,3));
5     v = (H(2,1)*ReferenceImage.P.U(ReferenceImage.index)+H(2,2)*
      ReferenceImage.P.V(ReferenceImage.index)+H(2,3));
6     w = (H(3,1)*ReferenceImage.P.U(ReferenceImage.index)+H(3,2)*
      ReferenceImage.P.V(ReferenceImage.index)+H(3,3));
```

After computing the transformed homogeneous coordinates, it is necessary to normalize the coordinates, by dividing by **w**. This normalization step ensures that the transformed pixel coordinates are properly scaled and within a reasonable range, also removes the homogeneous coordinates from the transformed pixel coordinates, resulting in a vector of two elements **(u, v)** that represent the regular image coordinates of the transformed pixel position.

Listing 2: Normalizing **u** and **v** coordinates by diving bt **w**

```
1     % Normalise
2     u = u./w;
3     v = v./w;
```

The indices of the visible pixels in the **ReferenceImage** are then determined by selecting only the pixels in the **ReferenceImage** that correspond to the visible pixels in the **WarpedImage**.The variable **ReferenceImage.index** contains the indices of all pixels in the **ReferenceImage**, and the variable **WarpedImage.visibility-index** contains the indices of the visible pixels in the **WarpedImage**.

Listing 3: Visible pixels identification

```
1    % Determine pixels inside of image and update indexes
2    WarpedImage.visibility_index = find(u<ReferenceImage.sIu & u>1 & v<
     ReferenceImage.sIv & v>1);
3    WarpedImage.index = ReferenceImage.index(WarpedImage.
     visibility_index);
```

The valid transformed pixel coordinates are then assigned to the appropriate indices in these matrices using the **index** field of the **WarpedImage** structure.Finally, a binary mask matrix to indicate which pixels in the **WarpedImage** are visible after the transformation. The matrix is created to be the same size as the **WarpedImage**, with a value of 1 assigned to the indices that correspond to visible pixels in the "index" field of the **WarpedImage** structure.

Listing 4: WarpedImage generation

```
1    % Stored in full image size matrix
2    WarpedImage.P.U = zeros(size(ReferenceImage.I));
3    WarpedImage.P.V = zeros(size(ReferenceImage.I));
4    WarpedImage.P.U(WarpedImage.index) = u(WarpedImage.visibility_index)
5    WarpedImage.P.V(WarpedImage.index) = v(WarpedImage.visibility_index)
6
7    % Propagate and update Mask and indexes
8    WarpedImage.Mask = zeros(ReferenceImage.sIv,ReferenceImage.sIu);
9    WarpedImage.Mask(WarpedImage.index) = 1;
```

As a result, a new **WarpedImage** with the same size as the **ReferenceImage** and update the pixel locations and mask of the **WarpedImage** using the updated pixel indices.

2. The planar Homography tracker also requires an intensity warping function:

$$I_2 = I_1(p_2)$$

Where:

- I_2 **(WarpedImage):** Which is the transformed image after warping, obtained by mapping the intensity values of the original image **I1** to new intensity values at transformed pixel coordinates p_2 using an intensity warping function.
- p_2**:** The transformed pixel coordinates of the image, which are obtained by applying the homography transformationto the pixel coordinates of the original image. These transformed pixel coordinates specify the new locations of the pixels in the transformed image **I2**.
- I_1 **(CurrentImage):** Which is the original image that is being transformed

Listing 5: Bilinear interpolation of the image

```
1     % Bilinear interpolation of image
2     WarpedImage.I = zeros(ReferenceImage.sIv, ReferenceImage.sIu);
3     WarpedImage.I(WarpedImage.index) = interp2(double(CurrentImage.I),
    WarpedImage.P.U(WarpedImage.index), WarpedImage.P.V(WarpedImage.
    index), 'linear');
```

As a result, the **WarpedImageSL3** implements an intensity warping function by performing bilinear interpolation of the **CurrentImage** onto the transformed pixel coordinates of the **ReferenceImage**, and returns a **WarpedImage** containing the transformed image coordinates, interpolated pixel values.The **Bilinear interpolation** is a method that estimates a value at an arbitrary point within the bounds of the known grid values.

3. Implement the stopping criterion for the non-linear iterative minimisation. This requires modifying the file track/TrackImageSL3.m

Listing 6: Stop criteria

```
    % Convergence criterion
    bool_res = 1;
    tol = 1e-4;
    % Stop criteria
    while(((norm(x)> tracking_param.max_x) || (norm(rescale(residues))<
   tracking_param.max_err)&&(bool_res == 1)) && (iter <= tracking_param
   .max_iter)) ;
    .
    .
    .

    % Check the convergence criterion
        if (iter > 1)
            if abs(norm_res(iter) - norm_res(iter-1)) < tol
                bool_res = 0;
            end;

        end;
    end;
```

To ensure that the iterative minimization loop continues until the **homography** matrix has been accurately estimated, and the tracking error has been reduced to a small enough value, four conditions have been used, referring to the **norm of x**, the **rescaled norm of the residuals**, and the **maximum number of iterations**.

- The first condition, $x\_norm > tracking\_param.max\_x$, checks whether the norm of the parameter vector x is larger than a predefined maximum value tracking-param.max-x.

Listing 7: Stop criteria for the norm(x)

```
    (norm(x)> tracking_param.max_x);
```

- The second condition, $norm(rescale(residues)) > tracking\_param.maxerr$, checks whether the difference between the pixel intensities of the warped image and the reference image, where the rescale is used to normalize the weighted difference between the pixel intensities of the warped image and the reference image to a range between 0 and 1.

Listing 8: Stop criteria for the error

```
        (norm(rescale(residues))< tracking_param.max_err));
```

- The Convergence criterion verifies when $norm(rescale(residues))$ doesn't change enough from the previous iteration, and changes the value of the $bool\_res$ variable to 0.

Listing 9: Stop criteria for the convergence

```
        % Check the convergence criterion
        if (iter > 1)
            if abs(norm_res(iter) - norm_res(iter-1)) < tol
                bool_res = 0;
            end;

        end;
        end;
```

- Finally, the last criterion checks the number of iterations performed by the loop.

Listing 10: Stop criteria for the number of iteractions

```
        (iter <= tracking_param.max_iter))
```

In summary, the first two criteria are evaluated using a logical **and** operation. If the value of the **rescaled norm of the residuals** is larger than the established value and has not yet **converged**, this logical operation will continue to function. After this, if the **norm of the vector x** is larger than the maximum allowable value, **or** the criterion for the **rescaled norm of the residuals** is false, this logical operation will continue to be a logical one. This process is carried out to ensure that the convergence of either x or the residuals stops the process early, without necessarily going through all the iterations. Then, this set of criteria is again evaluated using a logical **and** operation with the **maximum number of iterations**. If the number of iterations exceeds the expected value, the while loop ends.

4. Test the tracking of a patch up to the end of the sequence using different options.

   - Try tracking using CurrentJacobian, ReferenceJacobian and ESM algorithm. What differences can you note?

     After evaluating the three types of Jacobians under the same conditions, we found that the algorithm that produced the best results was the one using the reference Jacobian. It should be noted that:

     – The **reference Jacobian** (first-order Jacobian) describes how the transformation of the reference image affects the coordinates of points in the reference image.

     – The **current Jacobian** (first-order Jacobian) describes how the transformation of the reference model affects the coordinates of points in the current image.

     – The **second-order Jacobian** (Extended Similarity Measure) is a matrix that describes how the transformation of the reference model affects the pixel intensities in the current image.

     The first-order Jacobians are simpler and faster to compute, but may not capture all the variation in the image. The ESM considers more information about the image, but is more computationally expensive.

	Frames	Video link
Reference Jacobian	40	Link RJ
Current Jacobian	10	Link CJ
ESM	20	Link ESM

Table 1: Comparison using different tracking algoritms

As can be seen, the **Reference Jacobian** algorithm was able to track the object until frame 40. On the other hand, the **ESM Jacobian** became unstable after the pedestrian passed in front of the camera. Additionally, the **Current Jacobian** was only able to track up to 10 frames, and the captured image disappeared as soon as the pedestrian passed in front of the camera. Therefore, it was decided to use the **Reference Jacobian** as the base algorithm.

Listing 11: Testing Tracking parameters

```
1    tracking_params.max_iter = 100;
2    tracking_params.max_err = 60;
3    tracking_params.max_x = 0.1;
4    tracking_params.robust_method='tukey';
```

- Try tracking using the **M-estimator**, How does this affect the minimisation? What does the choice of the Tukey or Huber weighting functions do?

  The M-estimator is used to estimate the unknown parameters of the tracking model while considering the presence of outliers in the image data. The M-estimator task is to by down weighting the influence of the outliers, to prevent the tracking algorithm from stuck on the wrong feature and improve the accuracy.

  - The **Tukey estimator**, which uses a quadratic function to weight the influence of the data points.
  - The **Huber estimator**, which is a combination of a quadratic function and a linear function that is more robust to outliers than the Tukey estimator.

For this particular case by multiple trial and error, the best results were obtained by deactivating the M-estimator and only using the Reference Jacobian and the calculation of the pseudo-inverse. Under these conditions, tracking was achieved up to frame 100 as the video shows.

Video link Reference Jacobian results without M-estimator.

Figure 1: Reference Jacobian Jacobian without M-estimator results.

5. This question concerns a sequence of stereo images. You will find in the Versailles Canyons sequence both Left and Right images of a stereo pair.

   In the previous exercise, a monocular sequence was used to perform image-to-image tracking/registration. This sequence was acquired at 30Hz and only a small displacements are observed between successive images. The initial guess of the Homography was therefore the Identity matrix.

   - Track the same patch in both left and right images. Is this Homography the same for both Left and Right images?

     The homography is not necessarily the same for left and right images. The H matrix stores the homography for each image frame, they are separate homographies stored for the left and right cameras. To achieve the sift the following code was made.

Listing 12: Left and right switching

```
1     if(tracking_param.left_right)
2             %image_num_string = sprintf(['%0', num2str(
    capture_params.string_size), 'd'], k);
3             if(left)
4                 image_num_string = sprintf(['%0', num2str(
    capture_params.string_size), 'd'], image_number_left);
5                 file_I = [capture_params.data_dir, capture_params.
    prefix, image_num_string, capture_params.suffix];
6                 left = 0;
7                 image_number_left = image_number_left + 1;
8                 %disp('left')
9             else
10                image_num_string = sprintf(['%0', num2str(
    capture_params.string_size), 'd'], image_number_right);
11                file_I = [capture_params.data_dir_right,
    capture_params.prefix, image_num_string, capture_params.suffix];
12                left = 1;
13                image_number_right = image_number_right + 1;
14                %disp('right')
15            end;
16    else
17        image_num_string = sprintf(['%0', num2str(capture_params.
    string_size), 'd'], k);
18        file_I = [capture_params.data_dir, capture_params.prefix,
    image_num_string, capture_params.suffix];
19    end;
```

   By iterating over the sequence of images, the **HtrackHtrack** variable is set to the previous homography estimate. If $tracking\_param.left\_rightistrue$, then Htrack alternates between the HLL and HLR matrices depending on the current image number.

- Modify the code to "register" or "track" a patch from the Left image to the Right image for each image pair in the sequnce.

  Based on the expected displacement between the left and right images, it is determined through trial and error that the initial homography can be defined as:

$$H_L^R = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

- Is the computed Left-to-Right Homography the same for each stereo pair of images in the sequence?

  The homography is computed separately for each pair of images, based on the displacement between the left and right images in that pair. Therefore, the homography may be different for each pair of images, depending on the specific displacement between the left and right images in each pair.

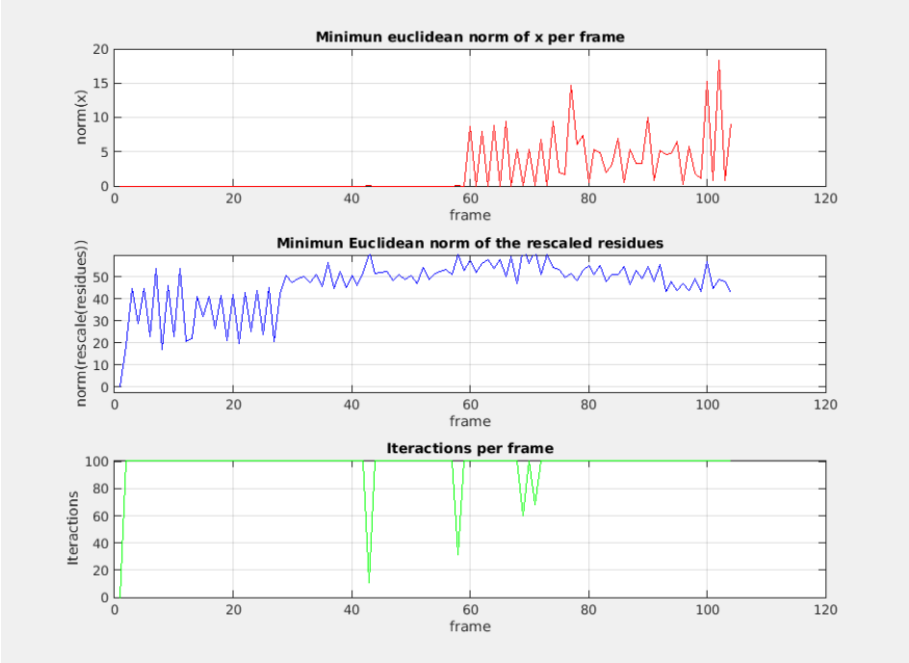

Figure 2: Results, Stereo tracking.

Video link results, stereo tracking.

As a consequence, the failure to update the homography matrix after frame 60 results in errors that prevent the norm of x and the norm of the residuals from converging properly. The values for the norm of x exceed 5 when computing tracking from the right camera and approach 0 when computing from the left camera, while the values for the norm of the residuals show a similar behavior. Eventually, the image disappears in the wrapping.

It is important to note that the full number of frames necessary for calculating the transformation is always reached, indicating that the tracking process is continuing despite the errors. However, the failure to update the homography matrix leads to a loss of the tracked image, as the mapped points no longer correspond to the correct points in the other image.

6. Investigate the effect of changing the reference patch each image or keeping it for as long as possible

   Changing the reference patch for each image can decrease accuracy and stability in planar homography tracking due to the need for recomputing the homography estimation and introducing errors. Additionally, the new reference patch may not be similar enough to the previous one, leading to incorrect homography estimation and tracking. This can be observed in the following image, where changing the reference patch for each image using the same parameters as previous cases results in the loss of tracking in less than **10 frames**, the **norm of x** increases exponentially after the third frame.

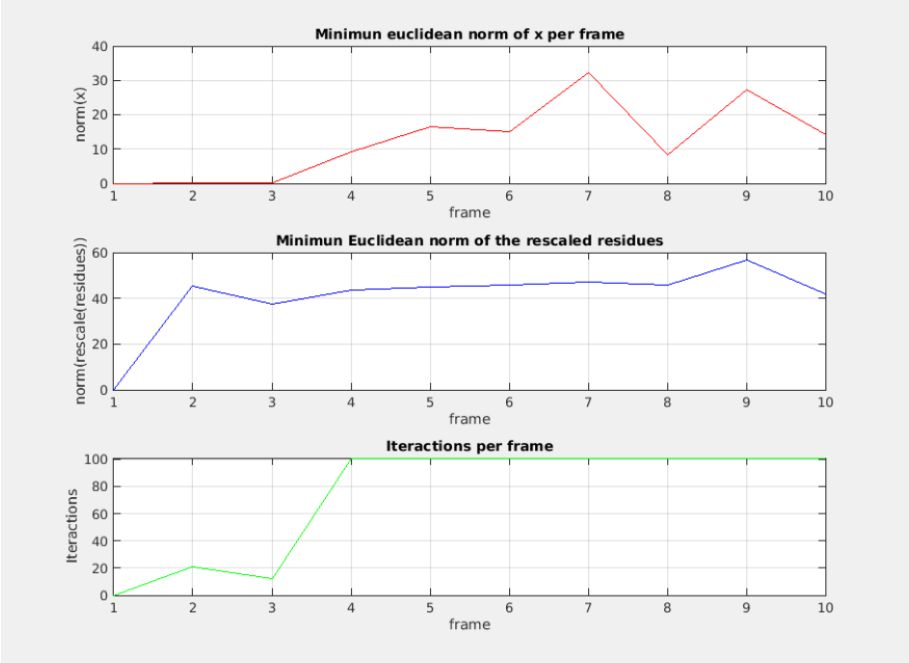

Figure 3: Effect of changing the reference patch in each image.

In the other hand, keeping the same reference patch for as long as possible can improve accuracy and stability by reducing computational cost and ensuring consistent tracking of the same patch throughout the sequence. However, there are scenarios where changing the reference patch is necessary, such as significant changes in appearance of the target object or when the initial reference patch is no longer visible.

Video link results, changing the reference patch.

The change in the reference image can be achieved using the following code:

Listing 13: Change reference

```
    if (tracking_param.changereference)
        ReferenceImage.I = CurrentImage.I;
        ReferenceImage.polygon = WarpedImage.polygon;
        ReferenceImage.index = WarpedImage.index;
        ReferenceImage.Mask = WarpedImage.Mask;
    end;
```

7. Devise a method for detecting when to change the reference patch.

   To accomplish this task, a patch is placed over frame 280 where a clear image of the feature to be tracked is available. The patch successfully tracked until frame 772, as shown in the video. In order to identify when it is necessary to change the reference patch, the results of the norm of x are taken into account. When it increases to 3, the reference patch is changed.

   It is important to note that the error norm significantly reduces after applying the change in the reference image. However, the error tends to remain with minimal fluctuations after the reference image change. Additionally, the maximum number of iterations per frame is reached for most cases. In the video, it is also observed that despite the presence of disturbances, the patch is able to recover after a few iterations.

   Video link Results, RGB tracking.

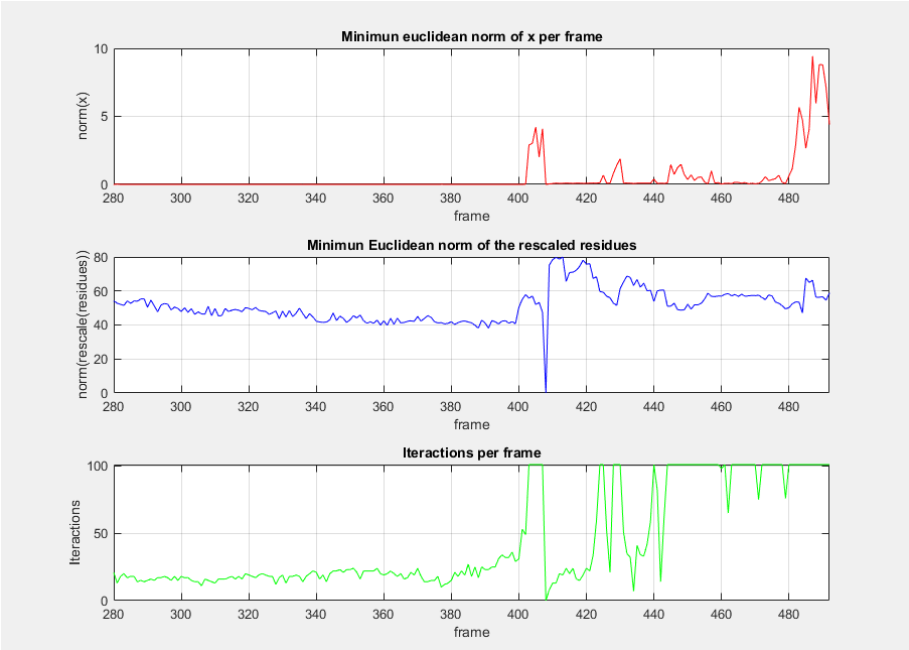

Figure 4: Reference Jacobian without M-estimator results.

This tracking task is particularly challenging due to the presence of numerous disturbances in the image sequence, such as sudden intensity changes and objects passing in front of the tracking patch. The variation in intensity caused by these disturbances, which leads to deformation and eventual failure of the planar tracking.