Technion - Israel Institute of Technology

Hw #2 : Vision Aided Navigation (086761)

By : Daniel Engelsman, 300546173

1 Basic Probability and Bayesian Inference

1. Random variable written in covariance form $x \sim N(\mu, \Sigma)$:

$$p(x) = \frac{1}{\sqrt{det(2\pi\Sigma)}} \cdot exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)) \tag{1.1}$$

Show its **information form** (Define: $\Sigma_{m\times m}$ - Matrix, $\eta_{m\times 1}$ - Vector):

$$Since: \qquad \Lambda \doteq \Sigma^{-1}, \eta \doteq \Lambda\mu \tag{1.2}$$

$$than: \qquad \Sigma = \Sigma^T = \Lambda^{-1} = (\Lambda^{-1})^T \qquad (Symmetrical) \tag{1.3}$$

$$and: \qquad \mu = \Lambda^{-1}\eta \quad and \quad \mu^T = \eta^T(\Lambda^{-1})^T \tag{1.4}$$

The exponent content :

$$(x^T - \mu^T)_{1\times m}(\Sigma^{-1})_{m\times m}(x-\mu)_{m\times 1} \tag{1.5}$$

$$(x^T\Sigma^{-1} - \mu^T\Sigma^{-1})_{1\times m}(x-\mu)_{m\times 1} \tag{1.6}$$

$$(x^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu - \mu^T\Sigma^{-1}x + \mu^T\Sigma^{-1}\mu)_{1\times 1} \tag{1.7}$$

$$Recall: \qquad \Sigma^{-1} = \Lambda, \quad \mu = \Lambda^{-1}\eta \quad and \quad \mu^T = \eta^T(\Lambda^{-1})^T = \eta^T(\Lambda^T)^{-1} \tag{1.8}$$

$$Plug: \qquad x^T\Lambda x - x^T\Lambda\Lambda^{-1}\eta - \eta^T(\Lambda^{-1})^T\Lambda x + \eta^T(\Lambda^{-1})^T\Lambda\Lambda^{-1}\eta \tag{1.9}$$

And we get :

$$x^T\Lambda x - \eta^T x - x^T\eta + \eta^T(\Lambda^{-1})^T\eta \tag{1.10}$$

$$x^T\Lambda x - 2\eta^T x + \eta^T(\Lambda^{-1})\eta \tag{1.11}$$

Plugging inside (**Eq. 1.1**) :

$$p(x) = \frac{1}{\sqrt{det(2\pi\Lambda^{-1})}} \cdot e^{-\frac{1}{2}(x^T\Lambda x - 2\eta^T x + \eta^T(\Lambda^{-1})\eta)} \tag{1.12}$$

$$finally, \quad \underline{\underline{p(x) = \frac{e^{(-\frac{1}{2}\eta^T\Lambda^{-1}\eta)}}{\sqrt{det(2\pi\Lambda^{-1})}} \cdot e^{(-\frac{1}{2}x^T\Lambda x + \eta^T x)}}} \tag{1.13}$$

2 Standard Observation Model

(a)

As in previous HW, initial belief pdf - $p(x)$:

$$x \sim N(\hat{x}_0, \Sigma_0); \qquad \underline{\underline{p(x) = \frac{1}{det(\sqrt{2\pi\Sigma_0})} \cdot exp(-\frac{1}{2}(x-\hat{x}_0)^T\Sigma_0^{-1}(x-\hat{x}_0))}} \tag{2.1}$$

$$Mahalanobis \quad norm : p(x) = \frac{1}{det(\sqrt{2\pi\Sigma_x})} \cdot exp(-\frac{1}{2}\|x-\hat{x}_0\|^2_\Sigma) \tag{2.2}$$

Observation model $z = h(x) + v, \quad v \sim N(0, \Sigma_v)$ and measurement likelihood :

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \eta \cdot p(z|x) \cdot p(z) \tag{2.3}$$

$$v \sim N(0, \Sigma_v); \qquad p(z|x) = \eta \cdot \frac{1}{det(\sqrt{2\pi\Sigma_v})} \cdot exp(-\frac{1}{2}(z-h(x))^T\Sigma_v^{-1}(z-h(x))) \tag{2.4}$$

$$Mahalanobis \quad norm : \quad p(z|x) = \frac{1}{det(\sqrt{2\pi\Sigma_v})} \cdot exp(-\frac{1}{2}\|z-h(x)\|^2_{\Sigma_v}) \tag{2.5}$$

(b)

Write an expression for the posterior probability :

$$p(x|z=z_1) = \frac{p(z_1|x)p(x)}{p(z_1)} = \eta \cdot p(z_1|x)p(x) \propto p(z_1|x)p(x) \tag{2.6}$$

(c)

Derive expressions for the a posteriori mean $\hat{x}_1$ and covariance Σ_1 :

$$prior, \qquad p(x) = \frac{1}{\sqrt{det(2\pi\Sigma)}} \cdot exp(-\frac{1}{2}(x-\hat{x}_0)^T\Sigma^{-1}(x-\hat{x}_0)) \tag{2.7}$$

The posterior $p(z|x) \sim N(h(x), \Sigma_v)$:

$$p(z_1|x) = \eta \cdot \frac{1}{det(\sqrt{2\pi\Sigma_v})} \cdot exp(-\frac{1}{2}(z_1 - h(x))^T \Sigma_v^{-1} (z_1 - h(x))) \tag{2.8}$$

Since :

$$\mu_v = h(x) = \Lambda_v^{-1}\eta \quad \Leftrightarrow \quad \mu^T = \eta^T (\Lambda_v^{-1})^T \tag{2.9}$$

$$and: \qquad \Lambda_v \doteq \Sigma_v^{-1}, \quad \eta_v \doteq \Lambda_v \mu_v = \Lambda_v h(x) \tag{2.10}$$

We get the Information form :

$$p(x) = \frac{1}{\sqrt{det(2\pi\Lambda_0^{-1})}} \cdot exp(-\frac{1}{2}\eta_0^T \Lambda_0^{-1} \eta_0) \cdot exp(-\frac{1}{2}x^T \Lambda_0 x + \eta_0^T x) \tag{2.11}$$

$$Similarly, \qquad p(z_1|x) = \eta \cdot exp(-\frac{1}{2}\eta_v^T \Lambda_v^{-1} \eta_v) \cdot exp(-\frac{1}{2}z_1^T \Lambda_v z_1 + \eta_v^T z_1) \tag{2.12}$$

Putting it all together :

$$p(x|z_1) \propto p(z_1|x)p(x) \tag{2.13}$$

$$p(x|z_1) \propto exp(-\frac{1}{2}\left[\eta_v^T \Lambda_v^{-1} \eta_v + \eta_0^T \Lambda_0^{-1} \eta_0\right] - \frac{1}{2}\left[x^T \Lambda_0 x + \eta_0^T x + z_1^T \Lambda_v z_1 + \eta_v^T z_1\right]) \tag{2.14}$$

(*) **To be resolved ... I guess that it's something with** 2.10 **Equations** (*)

(d)

A second measurement, z_2 is obtained . . . **! Good Luck !**

3 Multivariate Random Variable

(a) Given the following state transition model of the variable
$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1}, \quad w_{k-1} \sim N(0, \Sigma_w)$, write an expression for the motion model :

$$knowing \quad w_{k-1} = x_k - f(x_{k-1}, u_{k-1}) \tag{3.1}$$

$$p(x_k|x_{k-1}, u_{k-1}) = w \sim N(0, \Sigma_w) = \tag{3.2}$$

$$\frac{1}{det(\sqrt{2\pi\Sigma_w})} \cdot exp(-\frac{1}{2}(x_k - f(x_{k-1}, u_{k-1}))^T \Sigma_w^{-1}(x_k - f(x_{k-1}, u_{k-1}))) \tag{3.3}$$

(b) Express the a posteriori pdf in terms of f(prior, motion, observation) model :

$$(Markov \quad Assumption \qquad p(x_1|z_1, u_0) = \frac{p(z_1|x_1, u_0)p(x_1|u_0)}{p(z_1|u_0)} \tag{3.4}$$

$$and \quad Normalizer) \qquad p(x_1|z_1, u_0) \propto p(z_1|x_1)p(x_1|u_0) \tag{3.5}$$

Prediction stage:

$$p(x_1|u_0) = \int_{x_0} p(x_1|x_0, u_0) \ dx_0 \tag{3.6}$$

Finally,

$$p(x_1|u_0) \propto p(z_1|x_1) \int_{x_0} p(x_1|x_0, u_0) \ dx_0 \tag{3.7}$$

(c) Show that MAP estimate is equivalent to solving non-linear least squares problem :

$$Letx = x_{0:1} \doteq \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, \qquad where \quad x_i \in R^n \tag{3.8}$$

$$MAP: \qquad x_k^* = arg \ max \ p(x_k|u_{0:k-1}, z_{1:k}) \tag{3.9}$$

$$\tag{3.10}$$

Since log function is monotonous, we'll use it to express the inside exponentials :

$$x_k^* = arg \ min \ (-log(p(x_k|u_{0:k-1}, z_{1:k})) \tag{3.11}$$

Knowing that a posterior equation upholds the following :

$$p(x|z) \propto p(z|x)p(x) \qquad where: \tag{3.12}$$

$$p(x) \sim N(\hat{x}_0, \Sigma_0) \quad and \quad p(z|x) \sim N(z - h(x), \Sigma_v) \tag{3.13}$$

Which can be written as follows (as seen on Hw_1) :

$$p(x|z) \propto \underline{\underline{exp(-\frac{1}{2}(\|x - \hat{x}_0\|^2_{\Sigma_0} + \left\|z - h(x)\right\|^2_{\Sigma_v}))}} \tag{3.14}$$

We'll denote the argument as a cost function **J(x)** to be optimized for a local minima :

$$J(x) = \|x - \hat{x}_0\|^2_{\Sigma_0} + \left\|z - h(x)\right\|^2_{\Sigma_v} \tag{3.15}$$

$$where, \qquad x^* = arg\ min\ \ J(x) \tag{3.16}$$

We'll now open each of **J(x)** components, and linearize about initial guess - $\bar{x}$:

$$Define, \qquad x = \bar{x} + \Delta x \tag{3.17}$$

$$prior, \qquad x - \hat{x}_0 = \bar{x} + \Delta x - \hat{x}_0 \tag{3.18}$$

$$posterior, \qquad z - h(x) = z - h(\bar{x} + \Delta x) = z - h(\bar{x}) - h(\Delta x) \tag{3.19}$$

$$Linearize \quad \Delta x: \quad h(\Delta x) = (\nabla_x h) \cdot \Delta x = H \cdot \Delta x \tag{3.20}$$

Plugging inside (**Eq. 3.15**) :

$$J(\bar{x} + \Delta x) = \left\|\Delta x + (\bar{x} - \hat{x}_0)\right\|^2_{\Sigma_0} + \left\|z - h(\bar{x}) - H\Delta x\right\|^2_{\Sigma_v} \tag{3.21}$$

Using useful relation from Hw_1 ($a \equiv$ vector):

$$\|a\|^2_\Sigma = \left\|\Sigma^{-1/2} a\right\|^2 = (\Sigma^{-1/2}a)^T(\Sigma^{-1/2}a) = a^T\Sigma^{-1}a \qquad here: \tag{3.22}$$

$$J(\bar{x} + \Delta x) = \left\|\Sigma_0^{-1/2}(\Delta x + (\bar{x} - \hat{x}_0))\right\|^2 + \left\|\Sigma_v^{-1/2}(z - h(\bar{x}) - H\Delta x)\right\|^2 \tag{3.23}$$

Collect Jacobian martices and right hand side vectors :

$$\|a\|^2_\Sigma = \left\| \begin{pmatrix} \Sigma_0^{-1/2} \\ \Sigma_v^{-1/2} H\Delta x \end{pmatrix} \Delta x + \begin{pmatrix} -\Sigma_0^{-1/2}(\bar{x} - \hat{x}_0) \\ -\Sigma_v^{-1/2}(z - h(\bar{x})) \end{pmatrix} \right\|^2 \tag{3.24}$$

And hence,

$$\Delta x^* = arg\ \ min\ \ \left\| A\Delta x + (-b) \right\|^2 \quad \Rightarrow \quad A\Delta x = b \tag{3.25}$$

Since A is not necessarily invertible, we'll do as follows ($A \doteq$ - Information matrix) :

$$\Delta x^* = (A^T A)^{-1}(A^T) \cdot b \tag{3.26}$$

And we get :

$$A^T A \doteq \Lambda \quad \Rightarrow \quad Convergence \quad \Rightarrow \quad p(x|z) = N(\mu|\Sigma) \tag{3.27}$$

4 Hands-on Section

(a) General expression for the projection matrix - $P(x, X^G) \doteq K \cdot [R_G^C \; t_G^C] \cdot X^G$:

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R(1,1) & R(1,2) & R(1,3) & t_G^C(x) \\ R(2,1) & R(2,2) & R(2,3) & t_G^C(y) \\ R(3,1) & R(3,2) & R(3,3) & t_G^C(z) \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.1}$$

(b) Putting the numerical values of the given data (**Note**: $t_G^C = R_G^C \cdot t_C^G$):

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{bmatrix} 480 & 0 & 320 \\ 0 & 480 & 270 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5363 & -0.8440 & 0 & -458.9384 \\ 0.8440 & -0.5363 & 0 & -243.0052 \\ 0 & -0 & 1 & 400 \end{bmatrix} \begin{bmatrix} 350 \\ -250 \\ -35 \\ 1 \end{bmatrix} \tag{4.2}$$

And we get :

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{bmatrix} 87,888 \\ 59,344 \\ 365 \end{bmatrix} \Rightarrow \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \tilde{u}/\tilde{w} \\ \tilde{v}/\tilde{w} \end{bmatrix} = \begin{bmatrix} 240.789 \\ 162.586 \end{bmatrix} \quad [pxl] \tag{4.3}$$

(c) Calculate the re-projection error :

$$\Delta err = z - \pi(x, X^G) = \begin{bmatrix} 241 \\ 169 \end{bmatrix} - \begin{bmatrix} 240.789 \\ 162.586 \end{bmatrix} = \begin{bmatrix} 0.211 \\ 6.415 \end{bmatrix} \quad [pxl] \tag{4.4}$$

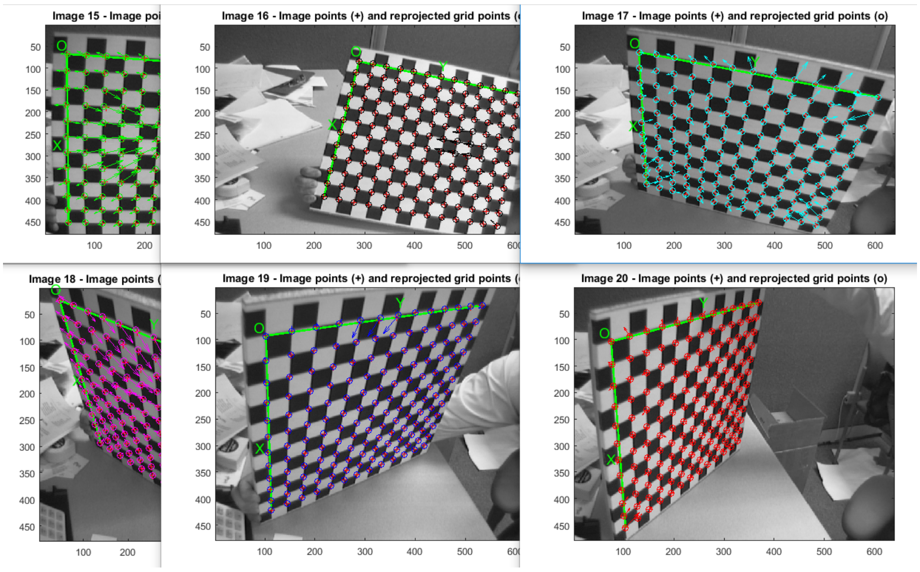
$$Norm : \quad \left\| \begin{bmatrix} 0.211 \\ 6.415 \end{bmatrix} \right\| = 6.4185 \quad [pxl] \tag{4.5}$$

```
% ———— 1.a/b Camera's Projection Matrix ———— %
% Calibration Matrix
K       = [480  0   320;
             0   480  270;
             0    0    1];
% Rotation Matrix
R_G_C = [    0.5363   -0.8440   0  ;
             0.8440    0.5363   0  ;
             0         0          1];
% Translation from Camera to Global
t_C_G = [-451.2459 257.0322  400]';
% Translation from Global to Camera
t_G_C = R_G_C*t_C_G;
% Camera Pose
Proj = K*[R_G_C t_G_C];
% 3D Global Point
l_G    = [350, -250, -35]';
% Homogenous Coordinates
P = Proj*[l_G; 1]
[u_t, v_t, w_t] = deal(P(1), P(2), P(3))
% Conversion to Pixels
[u, v] = deal(u_t/w_t, v_t/w_t)

% ——————— 1.c - Re-projection Error ——————— %
z = [241  169]';
p_calc = [u    v   ]';
err = z - p_calc;
norm(err)
```

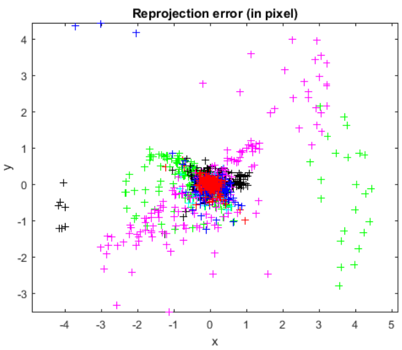
5

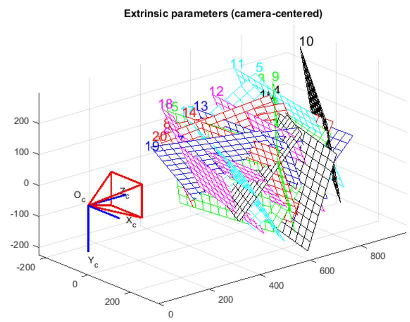
5. Camera Cailbration :

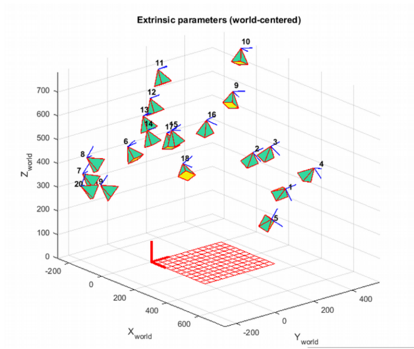
5. a/b/c First, we'll download the images' library and define the window and square's sizes Each of the images corners are extracted manually (#1 … #20) to ensure calibration relative to origin at top left of image. That way we create a calibrated set of the data.



Initial reprojection error : Extrinsic_1 : Extrinsic_2 :





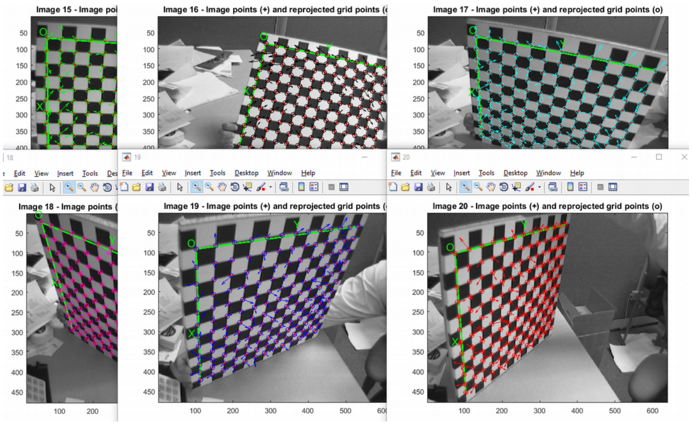


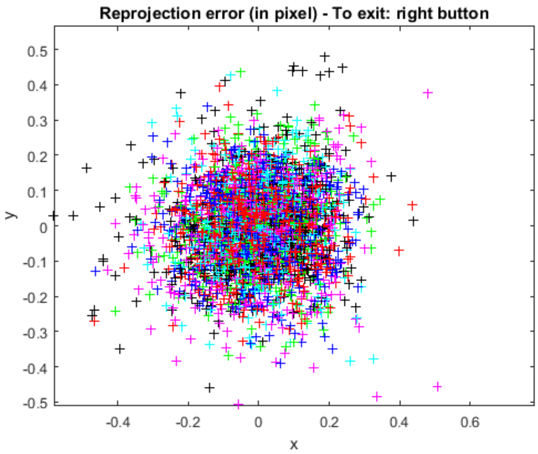
Applying **Recomp. corners** in the **Camera calibration tool** and then **Calibrating :**

```
Aspect ratio optimized (est_aspect_ratio = 1) -> both components of fc are estimated (DEFAULT).
Principal point optimized (center_optim=1) - (DEFAULT). To reject principal point, set center_optim=0
Skew not optimized (est_alpha=0) - (DEFAULT)
Distortion not fully estimated (defined by the variable est_dist):
     Sixth order distortion not estimated (est_dist(5)=0) - (DEFAULT) .

Main calibration optimization procedure - Number of images: 20
Gradient descent iterations: 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17..
Estimation of uncertainties...done


Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 657.64372   658.04109 ] +/- [ 0.40242   0.43055 ]
Principal point:       cc = [ 303.19237   242.55567 ] +/- [ 0.81857   0.74878 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +/- 0.00000
Distortion:            kc = [ -0.25610    0.13089    -0.00019    0.00004   0.00000 ] +/- [ 0.00314    0.01251
Pixel error:          err = [ 0.15297    0.13964 ]

Note: The numerical errors are approximately three times the standard deviations (for reference).
```

This ensures optimization of the manual corner extraction, that inevitably suffers from human error.

Once again, reprojection of images :

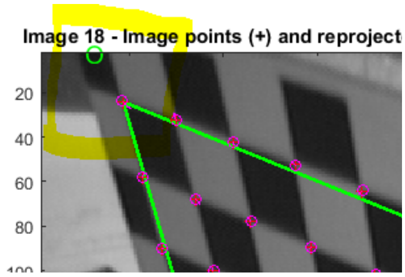




One can see that the new error is **dramatically** smaller than previous due to optimization (↑).

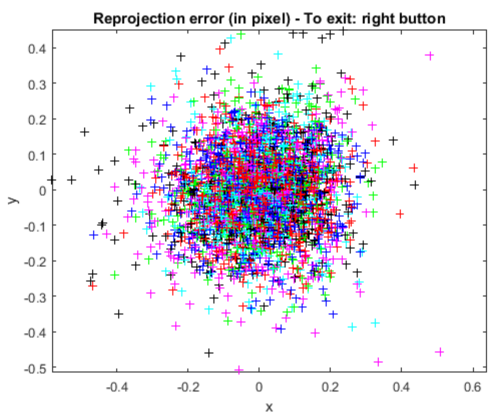
Yet, there are still a difference between “normal” pixel error (Image 8) and extremest (Image 20).

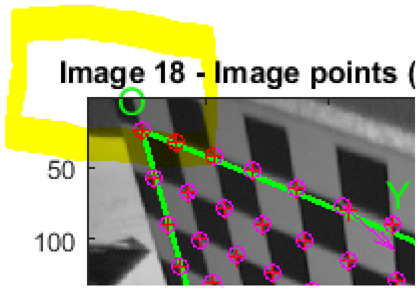
```
Selected image: 8
Selected point index: 42
Pattern coordinates (in units of (dX,dY)): (X,Y)=(5,9)
Image coordinates (in pixel): (309.30,204.58)
Pixel error = (-0.07545,-0.05456)
Window size: (wintx,winty) = (5,5)
done
Pixel error:          err = [ 0.15297   0.13964] (all active images)


Selected image: 18
Selected point index: 85
Pattern coordinates (in units of (dX,dY)): (X,Y)=(0,5)
Image coordinates (in pixel): (215.72,78.79)
Pixel error = (4.64081,3.35877)
Window size: (wintx,winty) = (5,5)
```



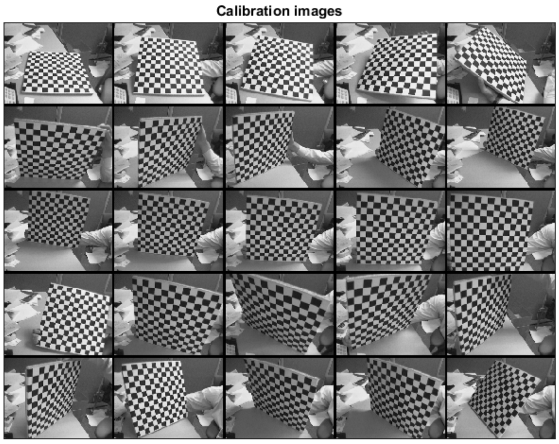
In order to improve out results, we’ll change partially images’ sizes, reproject on all images and obtain slightly better error:

wintx_4	9
wintx_5	7
wintx_6	9
wintx_7	7
wintx_8	7
wintx_9	9
wintx_default	5
winty	5
winty_1	9
winty_10	9
winty_11	9
winty_12	9
winty_13	9
winty_14	9

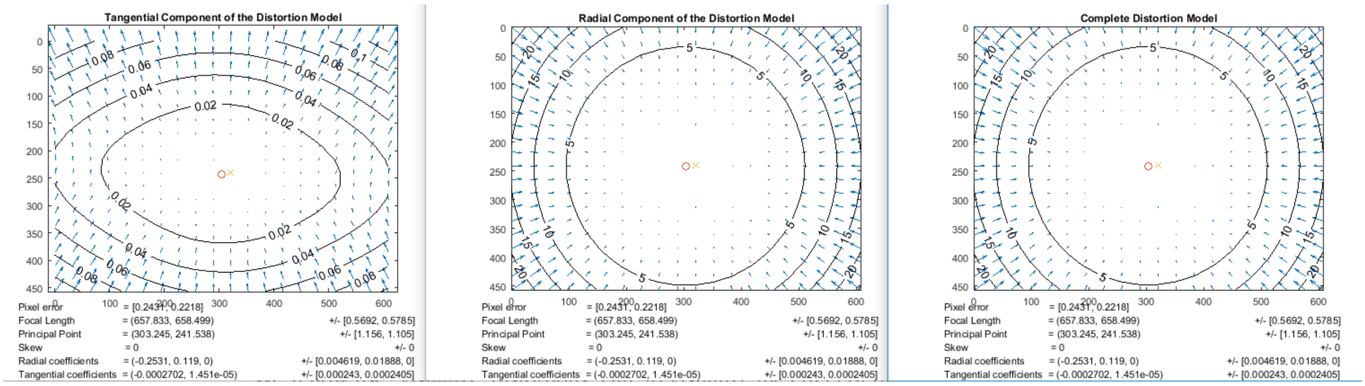




Adding 5 new Images in mosaic formation, and re-calibrate :



and then we'll use 'visualize_distortions' call, to visualize the effect of distortions on the pixel image, and the importance of the radial component versus the tangential component of distortion :



2.d As seen, we obtained the most accurate calibration results after the optimization :

```
Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 657.83275   658.49912 ] +/- [ 0.56919   0.57850 ]
Principal point:       cc = [ 303.24503   241.53789 ] +/- [ 1.15552   1.10528 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]  => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion:            kc = [ -0.25310   0.11896   -0.00027   0.00001  0.00000 ] +/- [ 0.00462   0.01888   0.00024
Pixel error:          err = [ 0.24309   0.22185 ]

Note: The numerical errors are approximately three times the standard deviations (for reference).
```

Which will now be plugged in our intrinsic parameters matrix (Calibration matrix **Part 1**) :

```
%% 2.d     ------ Intrinsic : Perspective Projection (X_c to Pixels) ----- %
u0 = 303.245; v0 = 241.5379;               % Principal Point [Pixels]
Kx = -0.2531; Ky = 0.11896;                % Defining Square pixel k=1
fx = 657.83275; fy = 658.5;                % Focal Lengths
alpha_x = Kx*fx; alpha_y = Ky*fy;          % focal length [Pixels]
        % ------------------ Calibration Matrix ------------------ %
K = [alpha_x 0 u0; 0 alpha_y v0; 0 0 1]    % add N index applied
```

```
Command Window
K =
  -166.4975         0   303.2450
          0   78.3352   241.5379
          0         0     1.0000
```

*<u>To conclude, the Flowchart of</u> **<u>Calibration Algorithm</u>** :*

Image Names & Read > Extract Grid Corner > Recompute Corners > (Re)-Calibration

6 Basic Image Feature Extraction

(a) Let us clarify some related terms :

1. **Detection :** Identify interest points (features) in the image
2. **Description :** Extract vector feature descriptor around each interest point
3. **Matching :** Determine correspondence between descriptors in two images.

Use a camera to capture 2 images of yourself :

Figure 1: Myself in a ”casual” posture

(b) Extract SIFT features in each image. Indicate feature scale and orientation :

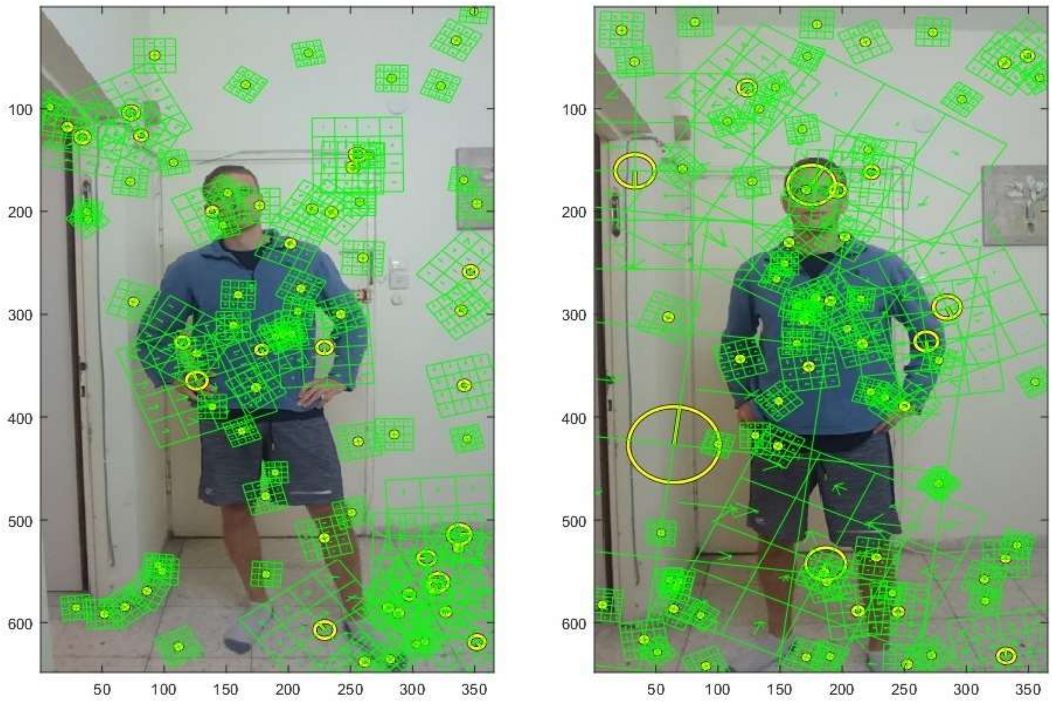


Figure 2: Same images pair with the extracted SIFT features

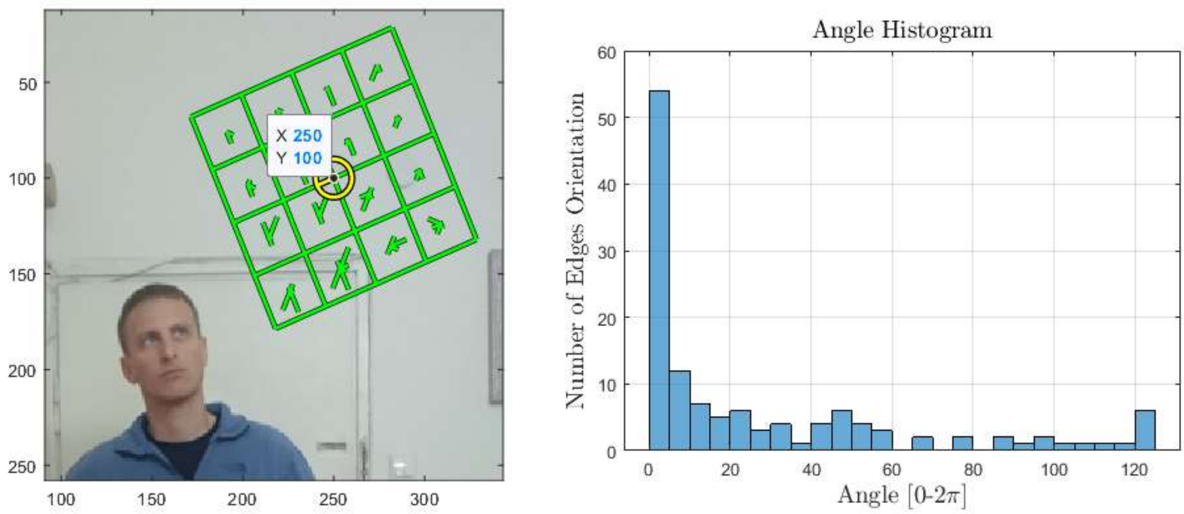


Figure 3: Representative feature's **position** @ [100, 250] of **scale** 10 and **orientation** -$\pi/8$

(c) Calculate putative matches by matching SIFT descriptors :

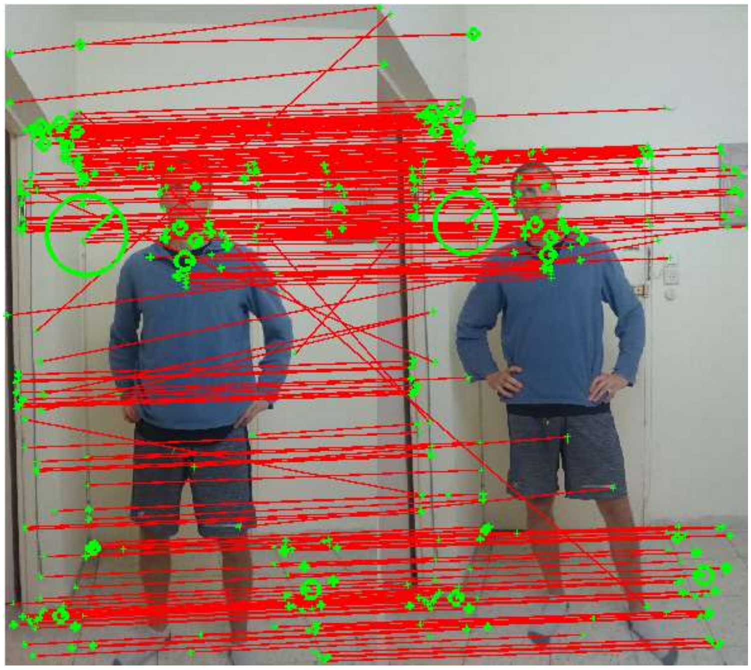


Figure 4: Threshold gradient magnitude = 3

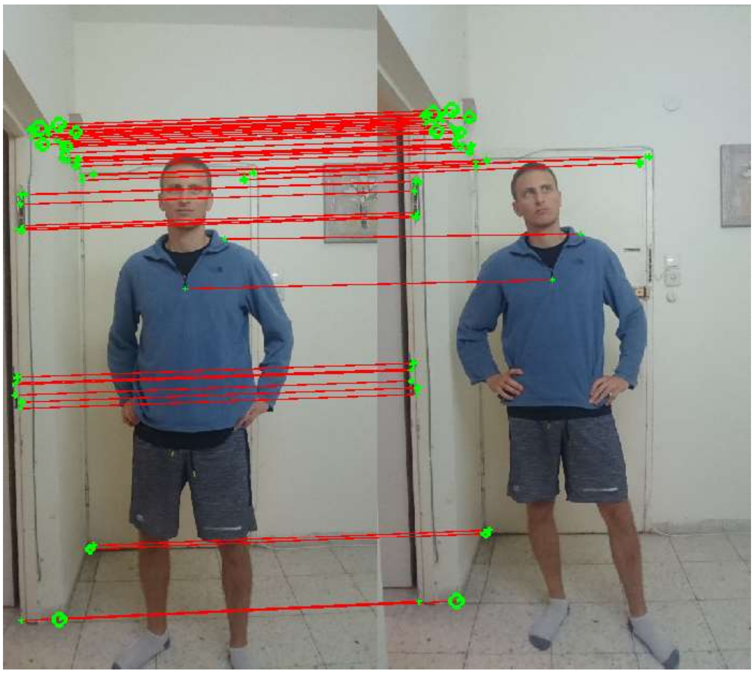


Figure 5: Threshold gradient magnitude = 15

Indicate representative inlier and outlier matches :

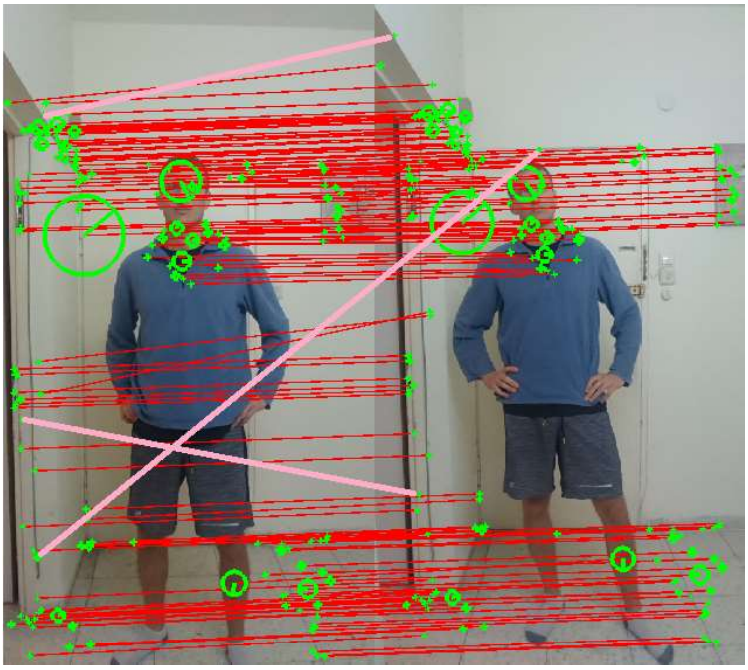


Figure 6: Outliers in pink

Hereby is attached the Matlab Code for section **6.b 6.c**:

```
%% ---------------------------- Part 2 ---------------------------- %
clc; clear all; close all;
% ------------ a. Capture 2 Images of yourself ------------ %
% I = imread('C:\Users\Daniel\Desktop\Vision Aided Navigation\Hw_i
    \Hw_2\Self_Pics\Img_0.jpg');
I = imread('C:\Users\Daniel\Desktop\Vision Aided Navigation\Hw_i\
    Hw_2\Self_Pics\Img_1.jpg');
I = imrotate(I, 90);                    % Rotate Image
scale = 0.25;                           % New Image scale
I = imresize(I, scale);                 % Scale down Image
figure('Color','white','rend','painters','pos',[2000 20 500 700]);
image(I);

%% ----- b. Compute the SIFT frames and descriptors ----- %
I = single(rgb2gray(I));
[f, d] = vl_sift(I);
```

```
% Generate random scatter on image pixels
perm = randperm(size(f,2)) ;
sel = perm(1:200) ;
h1 = vl_plotframe(f(:,sel)) ;
h2 = vl_plotframe(f(:,sel)) ;
set(h1,'color','k','linewidth',3) ;
set(h2,'color','y','linewidth',2) ;
h3 = vl_plotsiftdescriptor(d(:,sel),f(:,sel)) ;
set(h3,'color','g') ;

%% ------------------------------ Custom Frames ------------------------------ %
% Compute the descriptor of a SIFT frame at given position
I = single(rgb2gray(I));
[f, d] = vl_sift(I);
fc = [250; 100; 10; -pi/8];
[f,d] = vl_sift(I, 'frames', fc, 'orientations') ;
h3 = vl_plotsiftdescriptor(d,f) ; set(h3,'color','k', 'linewidth'
    ,3) ;
h4 = vl_plotsiftdescriptor(d,f) ; set(h4,'color','g', 'linewidth'
    ,2) ;
h1 = vl_plotframe(f) ; set(h1,'color','k','linewidth',4) ;
h2 = vl_plotframe(f) ; set(h2,'color','y','linewidth',2) ;

%%
figure; cols = 80; histogram(d, cols)

%% Basic matching - SIFT descriptors
clc; clear all; close all;
% ------------- Upload 2 images of the same object ------------- %
Ia = imread('C:\Users\Daniel\Desktop\Vision Aided Navigation\Hw_i\
    Hw_2\Self_Pics\Img_0.jpg');
Ib = imread('C:\Users\Daniel\Desktop\Vision Aided Navigation\Hw_i\
    Hw_2\Self_Pics\Img_1.jpg');

Ia = imrotate(Ia, 90);
Ib = imrotate(Ib, 90);
```

```matlab
scale = 1;
Ia = imresize(Ia,scale);
Ib = imresize(Ib,scale);
[fa, da] = vl_sift(im2single(rgb2gray(Ia)));
[fb, db] = vl_sift(im2single(rgb2gray(Ib)));

[matches, scores] = vl_ubcmatch(da, db, 3);

[drop, perm] = sort(scores, 'descend');
matches = matches(:, perm);
scores = scores(perm);

figure(1);
imagesc(cat(2, Ia, Ib));
axis image off ;
vl_demo_print('sift_match', 1);

figure(2);
imagesc(cat(2, Ia, Ib)) ;

xa = fa(1,matches(1,:)) ;
xb = fb(1,matches(2,:)) + size(Ia,2) ;
ya = fa(2,matches(1,:)) ;
yb = fb(2,matches(2,:)) ;
hold on ;
h = line([xa ; xb], [ya ; yb]) ;
set(h,'linewidth', 1, 'color', 'r') ;
vl_plotframe(fa(:,matches(1,:))) ;
fb(1,:) = fb(1,:) + size(Ia,2) ;
vl_plotframe(fb(:,matches(2,:))) ;
axis image off ;
```

–fin–