

# Visual Aided Navigation

## Homework #2

Alexander Shender: 328626114

Tzvi Horowitz: 021672910

### Question 1-1

In the covariance form the distribution is:

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

We develop the matrix multiplication inside into:

$$-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu) = -\frac{1}{2}x^T \Sigma^{-1}x + x^T \Sigma^{-1}\mu - \frac{1}{2}\mu^T \Sigma^{-1}\mu$$

Since all this in exponential, we can split them, taking the term not dependent on x outside, so that we get:

$$p(x) = \frac{\exp\left(-\frac{1}{2}\mu^T \Sigma^{-1}\mu\right)}{\sqrt{\det(2\pi\Sigma)}} \exp\left(-\frac{1}{2}x^T \Sigma^{-1}x + x^T \Sigma^{-1}\mu\right)$$

Now, by knowing that  $\begin{cases} \Sigma = \Lambda^{-1} \\ \mu = \Lambda^{-1}\eta \end{cases}$  we can replace the symbols in the equation and get:

$$p(x) = \frac{\exp\left(-\frac{1}{2}\Lambda^{-1}\eta^T \eta\right)}{\sqrt{\det(2\pi\Lambda^{-1})}} \exp\left(-\frac{1}{2}x^T \Lambda x + x^T \eta\right)$$

## Question 1-2

Given:

$$z = h(x) + v \quad ; v \sim N(0, \Sigma_v)$$

Initial belief:  $x \sim N(\tilde{x}_0, \Sigma_0)$

**(a)** The prior  $p(x)$  is the mere Normal distribution with known mean and covariance

$$p(x) = \frac{1}{\sqrt{\det(2\pi\Sigma_0)}} \exp\left(-\frac{1}{2}(x - \tilde{x}_0)^T \Sigma_0^{-1}(x - \tilde{x}_0)\right)$$

The measurement likelihood  $p(z|x)$  will also be Gaussian, since error is present. With mean in  $h(x)$ :

$$p(z|x) = N(h(x), \Sigma_r) = \frac{1}{\sqrt{\det(2\pi\Sigma_r)}} \exp\left(-\frac{1}{2}(z - h(x))^T \Sigma_r^{-1}(z - h(x))\right)$$

**(b)** Measurement  $z_1$  acquired. Posterior probability is given by the Bayes Rule:

$$p(x|z_1) = \frac{p(z_1|x) p(x)}{p(z_1)}, \quad \text{or} \quad \eta p(z_1|x) p(x),$$

*where  $\eta = p(z_1)^{-1}$ , not known for sure*

**(c)**  $p(x|z_1) = N(\tilde{x}_1, \Sigma_1)$  – derive expressions for this to hold for  $\tilde{x}_1, \Sigma_1$

We're about to find the MAP estimate for  $\tilde{x}_1$ .

$$\begin{aligned} \operatorname{argmax}_x p(x|z_1) &= \operatorname{argmax}_x p(z_1|x)p(x) = \operatorname{argmin}_x -|\log p(x|z_1)| \\ &= \operatorname{argmin}_x ||z - h(x)||_{\Sigma_r}^2 + ||x - \tilde{x}_0||_{\Sigma_0}^2 = \text{cost function} \end{aligned}$$

To reach the minimum we have to seek for the case there both of those term will be equal zero.  $h(x)$  is nonlinear, we have to linearize it around  $x_0$ , hence

$$h(\tilde{x}_1 + \Delta x) = h(\tilde{x}_1) + \Delta x \frac{\delta h}{\delta x} \Big|_{\tilde{x}_1}$$

$$z - h(x) = z - h(\tilde{x}_1 + \Delta x) = z - h(\tilde{x}_1) - \Delta x \frac{\delta h}{\delta x} \Big|_{\tilde{x}_1}$$

$$\text{we denote it } H = \Delta x \frac{\delta h}{\delta x} \Big|_{\tilde{x}_1} - \text{The Hessian Matrix}$$

$$x - x_0 = \tilde{x}_1 + \Delta x - x_0 = \Delta x + (x - x_0)$$

so we can rewrite

$$\operatorname{argmin}_x ||\Sigma_0^{-\frac{1}{2}}(\Delta x + (\tilde{x}_1 - x_0))||^2 + ||\Sigma_r^{-\frac{1}{2}}(H\Delta x + h(\tilde{x}_1) + z)||^2$$

Collecting the Jacobian matrix together we get the equation system:

$$A\Delta x - b = 0$$

Where

$$A = \begin{bmatrix} \Sigma_0^{-\frac{1}{2}} \\ \Sigma_r^{-\frac{1}{2}} H \end{bmatrix} ; \quad b = \begin{bmatrix} \Sigma_0^{-\frac{1}{2}} (\widetilde{x}_1 - x_0) \\ \Sigma_r^{-\frac{1}{2}} (h(\widetilde{x}_1) - z) \end{bmatrix}$$

Matrix A is 2X1, not invertible, so that for calculation we perform:

$$A\Delta x = b$$

$$A^T A \Delta x = A^T b$$

$$\Delta x = (A^T A)^{-1} A^T b$$

By calculating the  $\Delta x$  we update the  $\widetilde{x}_1$  by adding it to  $\widetilde{x}_1$ .

**(d)** Using the Markov assumption we can write:

$$p(x|z_1 z_2) = \eta p(z_2|x) p(x|z_1)$$

Where the expression for  $p(x|z_1)$  was found in (c). And expression for  $p(z_2|x)$  is obtained from regular observation model. So we get:

$$p(x|z_1 z_2) = \eta \frac{1}{\det(2\pi\Sigma_v)} \frac{1}{\det(2\pi\Sigma_1)} \exp\left(-\frac{1}{2} (\|z - h(x)\|_{\Sigma_v}^2 + \|x - \widetilde{x}_1\|_{\Sigma_1}^2)\right)$$

By multiplying 2 Normal Distributions we get a Normal distribution with its own  $\widetilde{x}_2$  and  $\Sigma_2$ .

$$p(x|z_1 z_2) \sim N(\widetilde{x}_2, \Sigma_2) \quad [-5]$$

and where are the expressions for the mean and cov?

### Question 1-3

Given a state transition model:

$$x_{k+1} = f(x_k, u_k) + w_k ; \quad w_k \sim N(0, \Sigma_w)$$

And regular observation model:

$$z = h(x) + v ; \quad v \sim N(0, \Sigma_v)$$

$$x \sim N(x_0, \Sigma_0)$$

**(a)** Expression for the motion model:

The covariance for  $x_{k+1}$  is derived:

$$\Sigma(x_{k+1}|x_k, u_k) = \Sigma(f(x_k, u_k)|x_k, u_k) + \Sigma(w_k|x_k, u_k) = 0 + \Sigma_w$$

Hence, the distribution is normal with the derived covariance:

$$p(x_k|x_{k-1}, u_{k-1}) \sim N(f(x_{k-1}; u_{k-1}), \Sigma_w)$$

Or

$$p(x_k|x_{k-1}, u_{k-1}) = \frac{1}{\sqrt{\det(2\pi\Sigma_w)}} \exp\left(-\frac{1}{2}(x_k - f(x_{k-1}; u_{k-1}))^T \Sigma_w^{-1}(x_k - f(x_{k-1}; u_{k-1}))\right)$$

**(b)** from the Bayes Rule:

$$p(x_1|z_1, u_0) = \frac{p(z_1|x_1, u_0) p(x_1|u_0)}{p(z_1|u_0)} = \eta p(z_1|x_1) p(x_1|u_0)$$

Expanding:

$$p(x|u_0) = \int^{x_0} p(x_1|u_0, x_0) p(x_0) dx$$

So that together we get:

$$p(x_1|z_1, u_0) = \eta p(z_1|x_1) \int^{x_0} p(x_1|u_0, x_0) p(x_0) dx$$

Where:

$$\begin{cases} p(z_1|x_1) \sim N(h(x_1), \Sigma_v) - \text{observation model} \\ p(x_1|u_0, x_0) \sim N(f(x_0; u_0), \Sigma_w) - \text{motion model} \\ p(x_0) \sim N(x_0, \Sigma_0) - \text{prior distribution} \end{cases}$$

**(c)** denote a general form: (using (b))

$$x^* = \operatorname{argmax}_{x_k} p(x_k | z_k, u_{k-1}) = \operatorname{argmax}_{x_k} \eta p(z_k|x_k) p(x_k|u_{k-1})$$

$$= \operatorname{argmax}_{x_k} \eta p(z_k|x_k) \int^{x_0} p(x_k|u_{k-1}, x_{k-1}) p(x_{k-1}) dx$$

Expanding the distribution forms (leaving the constant behind):

$$p(z_k|x_k) = \exp\left(-\frac{1}{2}\left(z - h(x_k)\right)^T \Sigma_v^{-1} \left(z - h(x_k)\right)\right)$$

$$\int^{x_0} p(x_k | u_{k-1} x_{k-1}) p(x_{k-1}) dx$$

$$= \int^{x_0} \exp\left(\left\| -\frac{1}{2}(x_k - f(x_{k-1}, u_{k-1})) \right\|_{\Sigma_v}^2 + \exp\left(\left\| -\frac{1}{2}(x - x_{k-1}) \right\|_{\Sigma_v}^2 \right) dx$$

To turn the problem into least squares, perform the log on the argmax expression:

$$\begin{aligned} \operatorname{argmax}_{x_k} p(x_k | z_k u_{k-1}) &= \operatorname{argmin}_{x_k} (-\log(p(x_k | z_k u_{k-1}))) \\ &= \operatorname{argmin}_{x_k} \left( \left\| -\frac{1}{2}(z - h(x_k)) \right\|_{\Sigma_v}^2 \right. \\ &\quad \left. + \int^{x_0} \exp\left(\left\| -\frac{1}{2}(x_k - f(x_{k-1}, u_{k-1})) \right\|_{\Sigma_v}^2 + \exp\left(\left\| -\frac{1}{2}(x - x_{k-1}) \right\|_{\Sigma_v}^2 \right) dx \right) \end{aligned} \quad [-6]$$

It can be seen that we're got the non-linear least squares problem.

**(d)** We are given the covariance matrix  $\Sigma_{0:1}$ . We can observe that to marginalize  $x_1$  we only need the covariance member which is responsible for the  $x_1$  covariance out of it, which means:

$$\Sigma'_1 = \Sigma_{11}$$

To get the information form marginalization more operations are performed: (from the lecture slide)

$$I'_1 = I_{11} - I_{10} I_{00}^{-1} I_{01}$$

what about the dim?

[-1]

## Question 2-1

(a) Projection Matrix is defined as

$$P = K [R | t_{c \rightarrow G}^c]$$

Which in our case is:

$$K = \begin{bmatrix} f_x & s & u_0 \\ & f_y & v_0 \\ & & 1 \end{bmatrix} = \begin{bmatrix} 480 & 0 & 320 \\ & 480 & 720 \\ & & 1 \end{bmatrix}$$

We are given  $t_{c \rightarrow G}^G$ , we need to change it to the Camera Frame of Coordinates. So that:

$$t_{c \rightarrow G}^c = R t_{c \rightarrow G}^G = \begin{bmatrix} 0.5363 & -0.8440 & 0 \\ 0.8440 & 0.5363 & 720 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -451.5259 \\ 257.0322 \\ 400 \end{bmatrix} = \begin{bmatrix} -458.9384 \\ -243.0052 \\ 400 \end{bmatrix}$$

So that the camera pose matrix is given by:

$$[R | t_{c \rightarrow G}^c] = \begin{bmatrix} 0.5363 & -0.8440 & 0 & -458.9384 \\ 0.8440 & 0.5363 & 0 & -243.0052 \\ 0 & 0 & 1 & 400 \end{bmatrix}$$

And all together the Projection Matrix is:

$$P = K [R | t_{c \rightarrow G}^c] = \begin{bmatrix} 480 & 0 & 320 \\ & 480 & 720 \\ & & 1 \end{bmatrix} \begin{bmatrix} 0.5363 & -0.8440 & 0 & -458.9384 \\ 0.8440 & 0.5363 & 0 & -243.0052 \\ 0 & 0 & 1 & 400 \end{bmatrix}$$

$$P = \begin{bmatrix} 2.6 & -4.1 & 3.2 & -922.9 \\ 4.1 & 2.6 & 7.2 & 1713.6 \\ 0 & 0 & 0 & 4 \end{bmatrix} * 10^2 \quad [-2]$$

(b)

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = P x^G$$

By inserting the point in the global frame  $x^G$  we find its location on the screen:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{bmatrix} 2.6 & -4.1 & 3.2 & -922.9 \\ 4.1 & 2.6 & 7.2 & 1713.6 \\ 0 & 0 & 0 & 4 \end{bmatrix} * 10^2 \begin{pmatrix} 350 \\ -250 \\ -35 \end{pmatrix} = \begin{pmatrix} 87888 \\ 223593.5 \\ 365 \end{pmatrix}$$

its not augmented....

By normalizing by the third (last member) which has to be equal 1 we get the true representation on the screen:

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 240.789 \\ 612.585 \\ 1 \end{pmatrix}$$

×

(c) We are given that the actual point on the screen is (241.5, 169). Finding the re-projection error:

$$Error = true - calculated = \begin{pmatrix} 241.5 \\ 169 \\ 1 \end{pmatrix} - \begin{pmatrix} 240.789 \\ 612.585 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.711 \\ -443.585 \\ 0 \end{pmatrix}$$

Third term is out of our interest anyway.

×



## Question 2-2

### 1. loading images

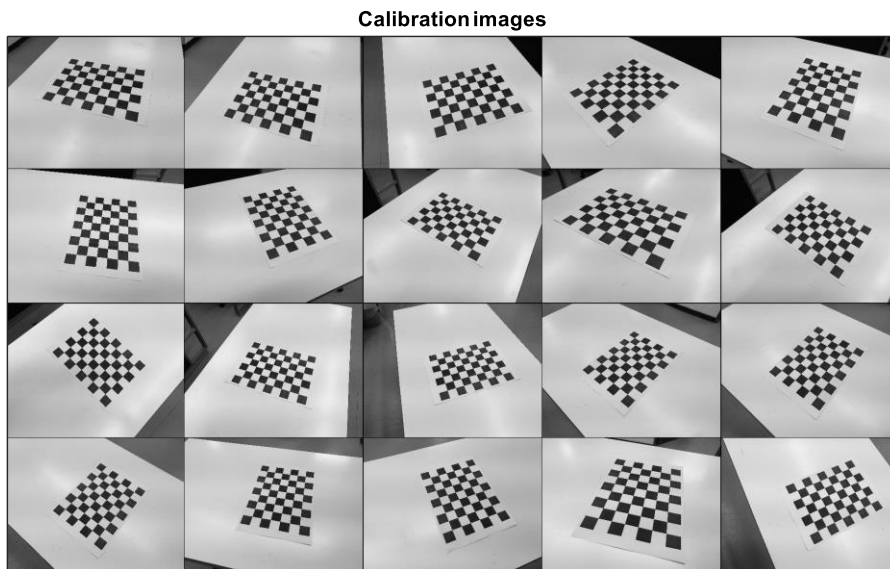
```
>> clear all
>> calib

.      Image15.jpg Image4.jpg
..     Image16.jpg Image5.jpg
Image1.jpg Image17.jpg Image6.jpg
Image10.jpg Image18.jpg Image7.jpg
Image11.jpg Image19.jpg Image8.jpg
Image12.jpg Image2.jpg  Image9.jpg
Image13.jpg Image20.jpg
Image14.jpg Image3.jpg

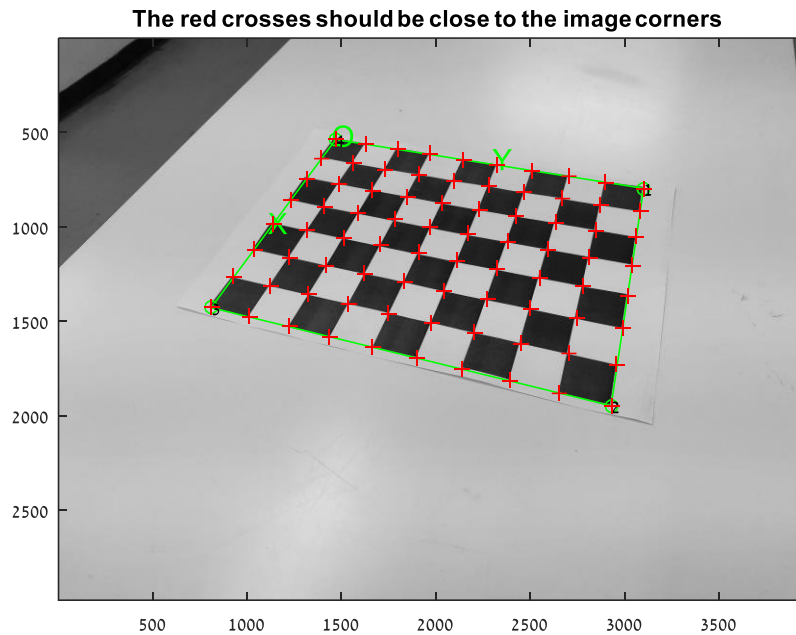
Basename camera calibration images (without number nor suffix): Image
Image format: (['r']='ras', 'b']='bmp', 't']='tif', 'p']='pgm', 'j']='jpeg', 'g']='jpeg', 'm']='ppm') j

Checking directory content for the calibration images (no global image loading in memory efficient mode)
Found images: 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...
done
```

### 2. mosaic view of the images



### 3. Extracting grid corners



### 4. Main calibration

Calibration parameters after initialization:

```
Focal Length:      fc = [ 3168.71279  3168.71279 ]
Principal point:    cc = [ 1983.50000  1487.50000 ]
Skew:              alpha_c = [ 0.00000 ] => angle of pixel = 90.00000 degrees
Distortion:        kc = [ 0.00000  0.00000  0.00000  0.00000  0.00000 ]
```

Main calibration optimization procedure - Number of images: 20

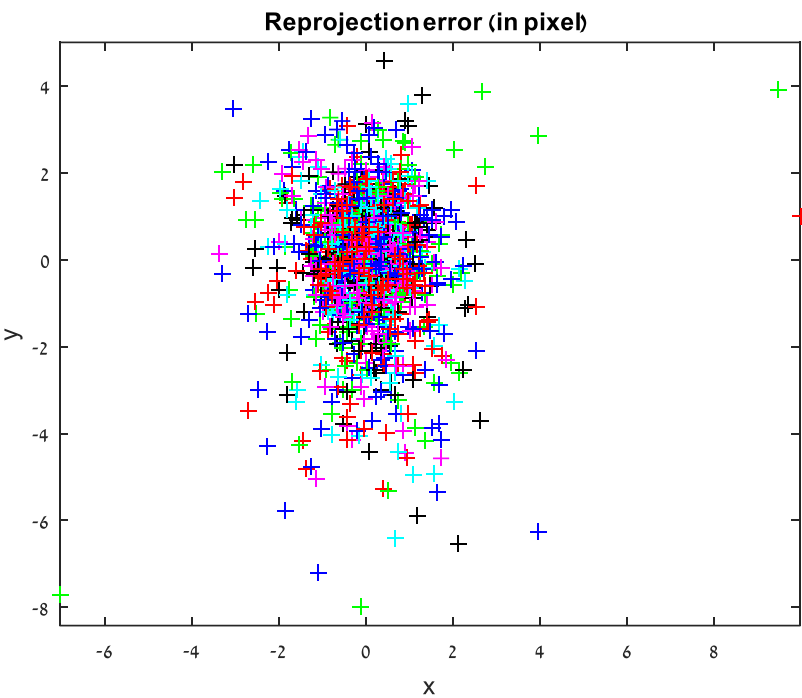
```
Gradient descent iterations: 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...done
Estimation of uncertainties...done
```

Calibration results after optimization (with uncertainties):

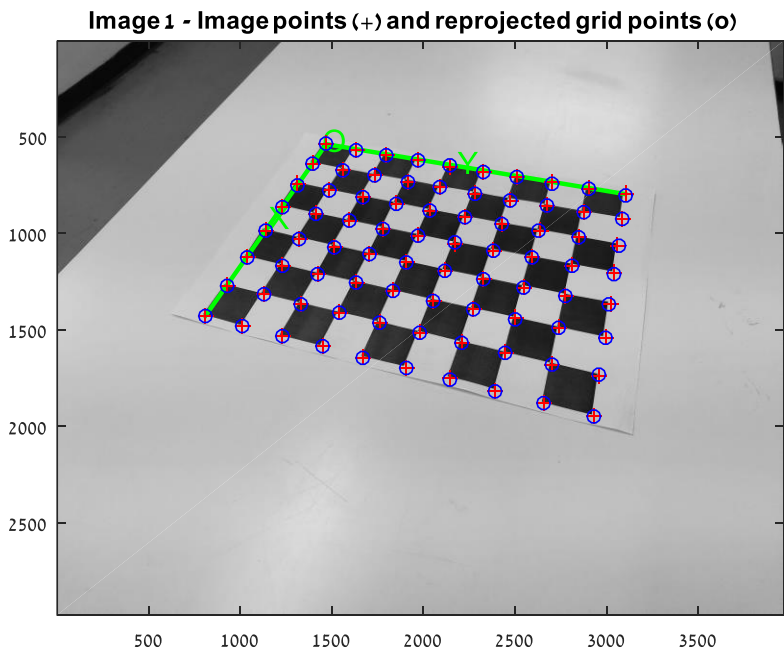
```
Focal Length:      fc = [ 3065.34745  3094.62676 ] +/- [ 20.07427  18.46297 ]
Principal point:    cc = [ 1992.17991  1494.17902 ] +/- [ 7.54370  20.68307 ]
Skew:              alpha_c = [ 0.00000 ] +/- [ 0.00000 ] => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion:        kc = [ 0.18641  -0.64835  -0.01089  0.00120  0.00000 ] +/- [ 0.00932  0.04192  0.00100  0.00098  0.00000 ]
Pixel error:       err = [ 0.91903  1.41297 ]
```

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

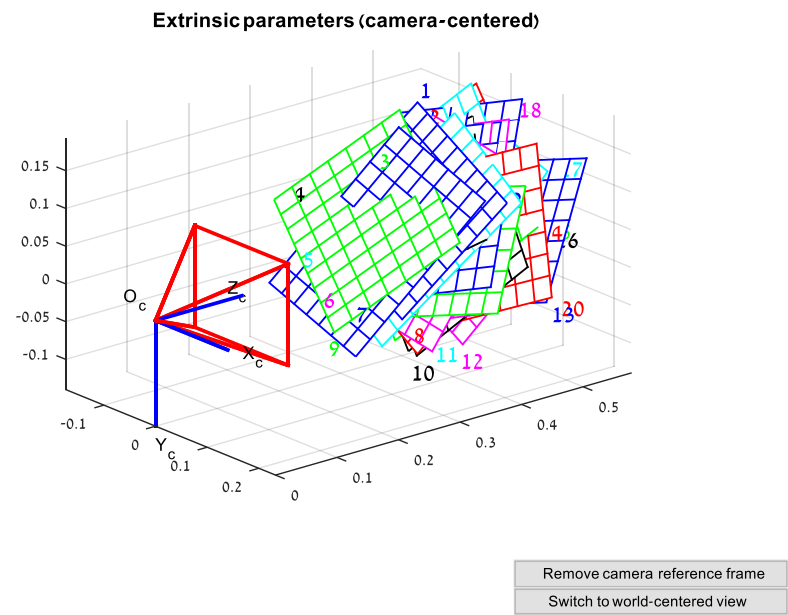
5. Reprojection error



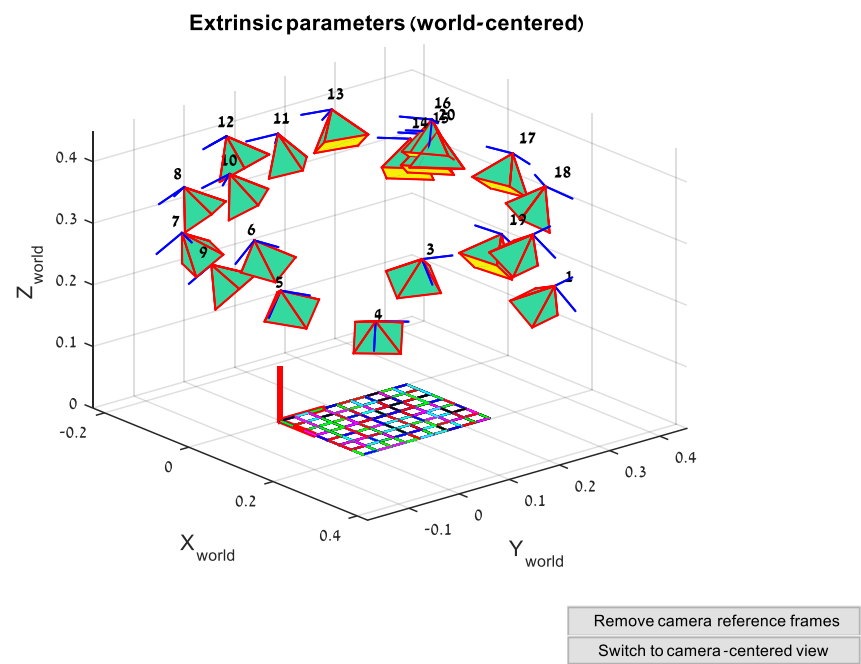
One of images:



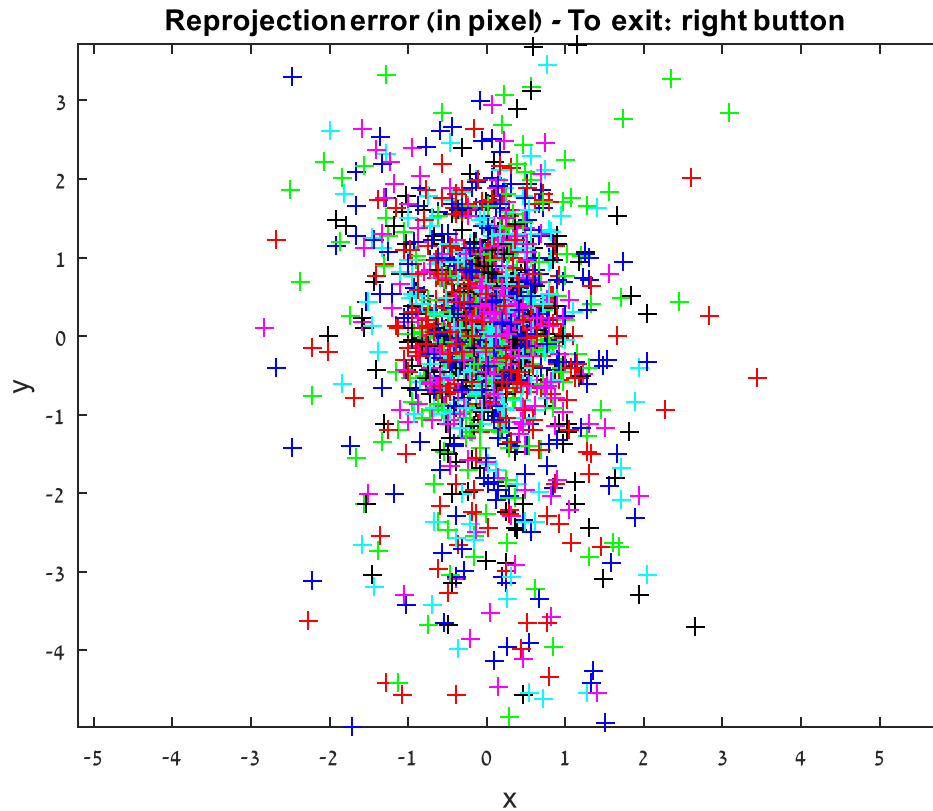
6. The extrinsic



7. The extrinsic world centered



8. Recomputing the corners automatically as told in the manual didn't give much improvement in the error.

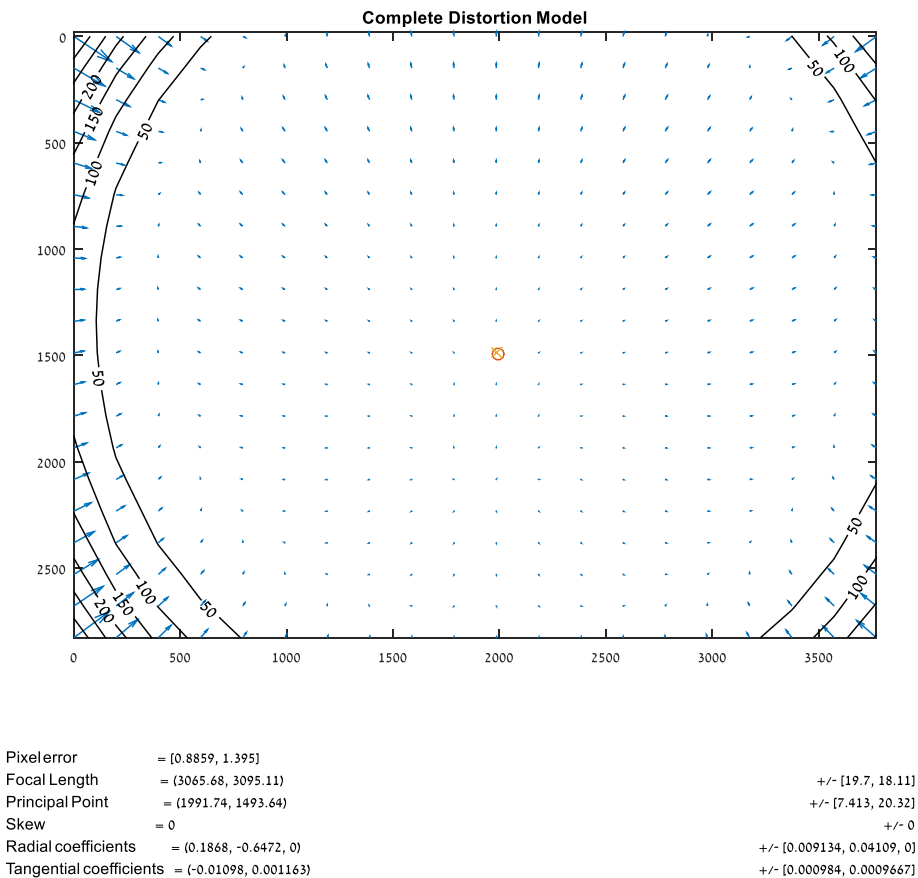


Decreasing the wintx winty area decreases the error, but the corners don't match the corners in the actual images.

Playing with distortion also didn't bring much improvement, without distortion it's still the best match. My guess is that the paper didn't lay flat enough and some edges got a bit "wavy". This should be done more precisely of course.

9. Getting familiarized with Distortion model, obtaining 3 graphs. Apparently my phone camera has a following distortion model: (complete)

I believe it isn't a complete circle but rather an ellipse because my cellphone has actually 2 lenses placed horizontally near one another, so that it elongates the shape of the model, while still not symmetrically. I have also went through other things explained in the tutorial.



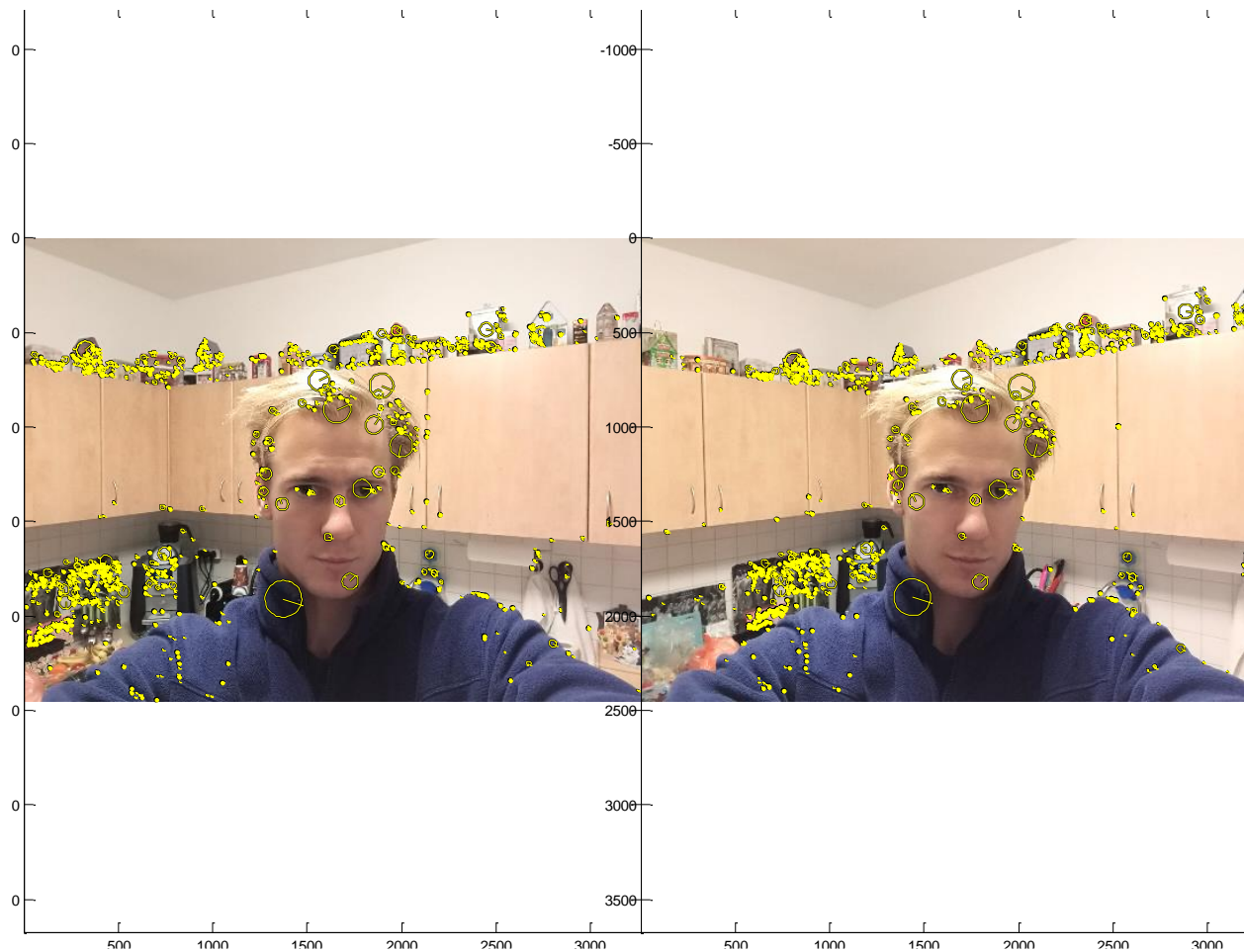
where is K ?

[-5]

### Question 2-3. SIFT feature extracting and matching

#### (a) The images are captured

**(b,c)** Each image separately gave me about 3300-3600 features, by using the peak threshold of 5. By running the matching algorithm I have found 1188 matches, which are indicated on the image below. Each feature is described by its location  $(x,y)$ , scale  $(\sigma)$  and orientation  $(\theta)$ , all of which may be seen on the images (rotation denoted by radius direction).



Between those matches many are outliers, which means they were indicated as matching by their scale and orientation, by their translation  $(x,y)$  differs from the translation of the majority of the points. Using methods like RANSAC we can eliminate those. I have indicated the outliers in red rectangles in the image below, there are of course many more which require zoom to identify (or RANSAC to eliminate). Or course matches which also match the major translation vector of the majority of the matches are the inliers. For example, most of the features on my face.



Outliers location

how do you know which is which?

[-5]



## MATLAB CODE:

```
%% Q2 - 1

%given
f=480; % [px]
u_v_0=([320,720]); %principal point
skew = 0; % assumption

R_gc=[0.5363 -0.8440 0;0.8440 0.5363 0;0 0 1];
%translation from C to G in G
t_cg=transpose([-451.2459,257.0322,400]);

%point coords
X_G = transpose([350,-250,-35]);
X_G(4) = 1;

u_v_obs = ([241.5,169]);

%SOLUTION

%calibration matrix
cal_mx = ([f 0 u_v_0(1); 0 f u_v_0(2); 0 0 1]);

%getting translation from C to G in C
t_cg_C = R_gc*t_cg;

P_mx = horzcat(R_gc,t_cg_C);

screen = cal_mx*P_mx*X_G;
screen_norm = screen/screen(3);
```

```

%% Q2 - 3

img1 =
imread('C:\Users\Alexander\Documents\MATLAB\VAN\hw2\Image1.jpg')
;
img1_gray = single(rgb2gray(img1));
img2 =
imread('C:\Users\Alexander\Documents\MATLAB\VAN\hw2\Image2.jpg')
;
% image(rgb2gray(img2));
img2_gray = single(rgb2gray(img2));

peak_thresh =5;

[fa,da] = vl_sift(img1_gray,'PeakThresh',peak_thresh);
[fb,db] = vl_sift(img2_gray,'PeakThresh',peak_thresh);

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

figure(1)
loc1=subplot(1,2,1)
image(img1)
set(loc1,'Position',[0.01 0.01 0.5 1]);

h1 = vl_plotframe(fa(:,matches(1,:)));
h2 = vl_plotframe(fa(:,matches(1,:))) ;
set(h1,'color','k','linewidth',2) ;
set(h2,'color','y','linewidth',1) ;
axis equal

% figure(2)
loc2=subplot(1,2,2)
image(img2)
set(loc2,'Position',[0.51 0.01 0.5 1]);

h3 = vl_plotframe(fb(:,matches(2,:)));
h4 = vl_plotframe(fb(:,matches(2,:))) ;
set(h3,'color','k','linewidth',2) ;
set(h4,'color','y','linewidth',1) ;
axis equal

```