Exam in SSY097

June 10th, 2019

Allowed materials : Pen/pencil, eraser.

The exam consists of six problems. Make sure that you have them all.

- Motivate all answers carefully.
- Use a new paper for each new numbered problem.
- Write on one side of the papers only.
- Write your anonymous number on each new page.
- Avoid using a red pen.
- If you want the result registered as SSY096, write this on the cover page.

The dates for the exam review will be announced on PingPong.

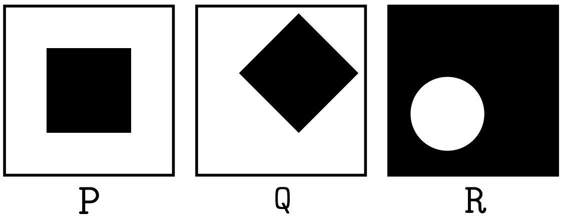
Grades

$\geq$ **8 points** Grade: 3

$\geq$ **11 points** Grade: 4

$\geq$ **14 points** Grade: 5

1 SIFT, 3 points

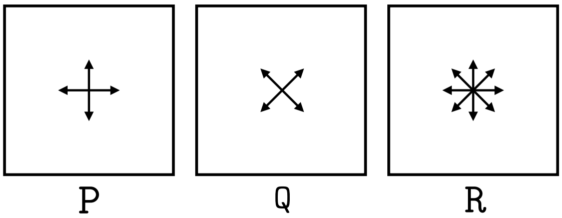


(a) Consider the three patches P, Q, and R shown above. For each patches, draw its vector bouquet representation.

Explicitly state how many entries your bouquets have.

How many entries does a normal SIFT descriptor have? Explain how you get to your solution.

(Solution) The corresponding vector bouquets are:



Each bouquet has 8 entries.
A SIFT descriptor has 128 entries. It is computed from a patch that is subdivided into 4 × 4 = 16 regions. For each region, a 8 dimensional histogram is computed. All histograms are concatenated, resulting in a $16 \cdot 8 = 128$ dimensional SIFT descriptors.

(b) In order to be robust against scale changes, SIFT builds a scale space representation and detects its keypoints as local extrema of that representation. Explain how ...

- ... the scale space representation is constructed,
- ... how the scale of each detected keypoint is computed, and
- ... how the size of the patch used to compute the SIFT descriptor is determined from the position and scale of a keypoint.

It is sufficient to describe the three steps in 2-3 sentences each, there is no need to explicitly derive / use equations.

(Solution) **Construction:** SIFT builds a scale space representation based on Difference-of-Gaussian (DoG) responses. To construct the space, it iteratively applies a Gaussian filter, resulting in a series of stronger and stronger blurred images. Subtracting adjacent images in this series then yields a series of DoG responses, which is SIFT's scale space representation.

Computing the scale: Keypoints are detected as local extrema (with respect to the neighboring pixels in the same DoG response and the two neighboring DoG responses) in the scale space. Afterwards the 3D position (image position and scale) of all keypoints that pass the non-maximum suppression stage are refined by fitting a second-order curve in the keypoints 3D neighborhood.

Computing the patch size: Assuming that a keypoint is detected at scale s, the size of the patch is given by a canonical size p that is scaled by keypoint's scale s. Thus, a patch of size $s \cdot p \times s \cdot p$ is extracted around the keypoint position.

(c) Consider the task of descriptor matching between two images $\mathcal{P}$ and $\mathcal{Q}$. The standard approach searches for each descriptor p from $\mathcal{P}$ for the nearest neighboring descriptor q_1 in $\mathcal{Q}$. However, many of the resulting matches will be wrong. In practice, it is thus common to use Lowe's ratio test[1] to filter out wrong matches.

Write down the formula for the ratio test. Make sure to explain additional variables that you introduce.

Why is the ratio test used instead of a threshold on the absolute descriptor distance $||p - q_1||_2$ to filter out wrong matches?

(Solution) Let q_1 and q_2 be the first and second neighbor of p in image $\mathcal{Q}$. Lowe's ratio test is then defines as

$$\frac{||p - q_1||}{||p - q_2||} < \tau \ ,$$

where τ is typically in the range $[0.6, 0.8]$. The ratio test measures how much closer p is to its first neighbor in relation to its second neighbor. If a match is ambiguous, i.e., if both neighbors have a similar distance to p, then the ratio will be close to one. If the nearest neighbor is much closer to p than the second nearest neighbor, then the ratio will be close to 0.

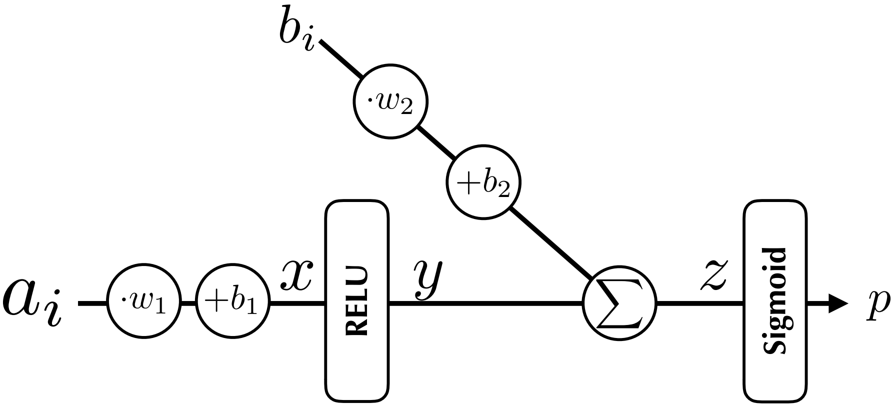
SIFT descriptors are not uniformly distributed in the SIFT descriptor space. Instead, there are regions that are more densely populated than others (for example corresponding to repeating patterns in an image). Consequently, there is no single threshold on the absolute distance that can be used to detect wrong matches. In contrast, the ratio test normalizes the distance to the nearest neighbor by the distance to the second nearest neighbor and thus removes the effect of different densities in different parts of the descriptor space.

[1] Also known as the SIFT ratio test.

2 Statistical Learning, 3 points

Consider the neural network shown below that performs binary classification on a tuple (a_i, b_i) of two scalar input values $a_i, b_i \in \mathbb{R}$. The RELU and Sigmoid functions are given as $\text{RELU}(x) = \max(0, x)$ and $\text{Sigmoid}(z) = \frac{e^z}{1+e^z}$. The derivative of the RELU function is given as

$$\frac{\partial \text{RELU}(x)}{\partial x} = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise} \end{cases} .$$



(a) Provide the loss function L_i for a positive example (a_i, b_i). The loss should depend on the probability p estimated by the Sigmoid layer. As in the lecture and the lab, use the negative log-likelihood loss.

(Solution) Let L_i be the loss for a positive example. We want to maximize p, i.e., the probability computed by the classifier that the example is a positive one. The negative log-likelihood loss is thus given by $L_i = -\ln(p)$.

(b) Given a positive example $(a_i, b_i) = (10, -5)$, compute the derivatives

$$\frac{\partial L_i}{\partial w_1}, \quad \frac{\partial L_i}{\partial b_1}, \quad \frac{\partial L_i}{\partial w_2}, \quad \frac{\partial L_i}{\partial b_2}$$

through the backpropagation algorithm. To this end, first perform a forward pass to compute values for x, y, z, and p. Next, use the chain rule to derive formulas for the derivatives in the backward pass. Compute the actual values for the derivatives using your equations and the values for x, y, z, and p computed during the forward pass.
The current values for w_1, b_1, w_2, and b_2 are

$$w_1 = 10, \qquad b_1 = 50, \qquad w_2 = 5, \qquad b_2 = -125 \ .$$

(Solution) We have the following relations:

$$\begin{aligned} x &= w_1 \cdot a_i + b_1 \\ y &= \max(0, x) \\ z &= w_2 \cdot b_i + b_2 + y \\ p &= \frac{e^z}{1+e^z} \end{aligned}$$

The forward pass yields the following values:

$$\begin{aligned} x &= 150 \\ y &= 150 \\ z &= 0 \\ p &= 0.5 \end{aligned}$$

We need the following derivatives for the Sigmoid and the RELU:

$$\begin{aligned} \frac{\partial p}{\partial z} &= \frac{e^z(1+e^z)-(e^z)^2}{(1+e^z)^2} \\ &= \frac{e^z}{1+e^z}\cdot\frac{1}{1+e^z} = p\cdot(1-p) \\ \frac{\partial y}{\partial x} &= \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

The chain rule provides the following relationships for the gradients:

$$\begin{aligned} \frac{\partial L_i}{\partial z} &= \frac{\partial L_i}{\partial p}\frac{\partial p}{\partial z} = -\frac{1}{p}\cdot p\cdot(1-p) = (p-1) \\ \frac{\partial L_i}{\partial b_2} &= \frac{\partial L_i}{\partial z}\frac{\partial z}{\partial b_2} = (p-1)\cdot 1 \\ \frac{\partial L_i}{\partial w_2} &= \frac{\partial L_i}{\partial z}\frac{\partial z}{\partial w_2} = (p-1)\cdot b_i \\ \frac{\partial L_i}{\partial y} &= \frac{\partial L_i}{\partial z}\frac{\partial z}{\partial y} = (p-1) \\ \frac{\partial L_i}{\partial x} &= \frac{\partial L_i}{\partial z}\frac{\partial y}{\partial x} = (p-1) \quad \text{since } x > 0 \text{ in our example} \\ \frac{\partial L_i}{\partial b_1} &= \frac{\partial L_i}{\partial x}\frac{\partial x}{\partial b_1} = (p-1)\cdot 1 \\ \frac{\partial L_i}{\partial w_1} &= \frac{\partial L_i}{\partial x}\frac{\partial x}{\partial w_1} = (p-1)\cdot a_i \end{aligned}$$

Using the values computed during the forward pass then yields

$$\begin{aligned} \frac{\partial L_i}{\partial b_2} &= \frac{\partial L_i}{\partial z}\frac{\partial z}{\partial b_2} = -0.5 \\ \frac{\partial L_i}{\partial w_2} &= \frac{\partial L_i}{\partial z}\frac{\partial z}{\partial w_2} = 2.5 \\ \frac{\partial L_i}{\partial b_1} &= \frac{\partial L_i}{\partial x}\frac{\partial x}{\partial b_1} = -0.5 \\ \frac{\partial L_i}{\partial w_1} &= \frac{\partial L_i}{\partial x}\frac{\partial x}{\partial w_1} = -5 \end{aligned}$$

(c) Using a learning rate of 0.01, what is the next value for w_1?

(Solution) Using a learning rate of 0.01, the next value for w_1 is given as

$$w_1 = w_1 - 0.01\cdot\frac{\partial L_i}{\partial w_1} = 10 + 0.05 = 10.05$$

3 Robust Model Fitting & Statistical Learning, 3 points

In this question, we consider the problem of fitting a polynomial into a set of 2D data points: Given a set of N 2D points $\{(x_i, y_i) | i \in [1, \cdots, N]\}$, we are looking to estimate the parameters $\mathbf{w} = (w_0, ..., w_M)$ of a polynomial $y(x|\mathbf{w}) = \sum_{j=0}^{M} w_j \cdot x^j$ of degree M. We denote the residual between the observed y-coordinate y_i and the predicted y-coordinate $y(x_i|\mathbf{x})$ for the i-th datapoint as $r_i(\mathbf{w}) = y_i - y(x_i|\mathbf{w})$.

(a) A standard approach to estimate the parameters $\mathbf{w}$ of the polynomial is to minimize a sum of squared errors:

$$\min_{\mathbf{w}} \sum_{i=1}^{N} (r_i(\mathbf{w}))^2 \quad .$$

Show that minimizing the sum of squared errors is the same as maximizing the likelihood

$$\max_{\mathbf{w}} \prod_{i=1}^{N} p(r_i(\mathbf{w})) \quad ,$$

where $p(x)$ is a Normal / Gaussian distribution with zero mean and a standard deviation of σ (the actual value of σ is not important here).

(Solution) Using $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$, we have

$$\max_{\mathbf{w}} \prod_{i=1}^{N} p(r_i(\mathbf{w})) = \max_{\mathbf{w}} \frac{1}{\sqrt{2\pi\sigma^2}} \prod_{i=1}^{N} e^{-\frac{r_i(\mathbf{w})^2}{2\sigma^2}} \quad .$$

Here, we can safely ignore the constant factor $\frac{1}{\sqrt{2\pi\sigma^2}}$ as it does not affect the solution $\mathbf{w}$ that maximizes the likelihood.

Maximizing the likelihood is the same as minimizing the negative log-likelihood (as discussed in the lecture), yielding

$$\min_{\mathbf{w}} - \ln \left(\prod_{i=1}^{N} e^{-\frac{r_i(\mathbf{w})^2}{2\sigma^2}} \right) = \min_{\mathbf{w}} - \sum_{i=1}^{N} - \frac{r_i(\mathbf{w})^2}{2\sigma^2} = \min_{\mathbf{w}} \frac{1}{2\sigma^2} \sum_{i=1}^{N} r_i(\mathbf{w})^2 \quad .$$

Again, dropping the constant factor $\frac{1}{2\sigma^2}$ does not change the result of the minimization. Consequently, we arrive at a sum of squared error loss function.

(b) The problem of assuming that the residuals follow a Normal distribution (and thus least-squares fitting) is that a Normal distribution cannot model outliers. Consequently, outlier measurements will distort the estimated parameters.

Explain how robust cost functions can be used to reduce the impact of outliers.

Name one robust cost function discussed in the lecture (not RANSAC).

(Solution) The motivation behind robust cost functions is to decrease the cost of outliers in the loss function, thus reducing their impact on the fitted parameters. Compared the the sum of squared residuals loss, this means using a probability distribution to model the residuals under which outliers (i.e., large residuals) are more likely compared to the Normal distribution.

Two robust cost functions discussed in the lecture are using the L1 norm instead of the L2 norm and the Huber cost function.

(c) In a slight modification of the problem setting from above, we are given a set of N 2D points $\{(x_i, y_i)|i \in [1, \cdots, N]\}$ that can be used for training and an additional set of K 2D points $\{(x_i, y_i)|i \in [1, \cdots, K]\}$. We still want to fit a polynomial function $y(x|\mathbf{w})$ to the data-points, but do not know which degree M to use.

Describe an approach that determines a suitable degree M and fits the polynomial function $y(x|\mathbf{w})$ to the data while trying to avoiding overfitting.

(Solution) Overfitting occurs if our model (in this case, the polynomial function) has too many parameters (in this case, the degree M is too large). As such, we would like to find a model with as few parameters (a small M) as necessary to explain the data well. In order to measure how well a model fits the data, we use the second set as a validation dataset.
The resulting approach works like this: For all potential values of M (M has to be from the range $[0, N]$ since a polynomial of degree N is sufficient to explain N data points), we fit the parameters of the resulting polynomial to the N data points. We then measure the performance of the polynomial on the K validation points. Finally, we select the smallest value for M (and the corresponding parameters) such that the validation loss is either close to the minimum or minimal.

4 RANSAC, 3 points

(a) Consider two minimal solvers for the same problem with the following properties:

- **Solver 1**: The solver requires $m = 3$ data points, takes 1 time unit to run, always returns 4 solutions[2], and evaluating one model on a single data point requires 0.01 time units.
- **Solver 2**: The solver requires $m = 4$ data points, takes 1 time unit to run, returns 1 solution if all data points are inliers and 0 solutions otherwise, and evaluating one model on a single data point requires 0.01 time units.

Derive equations for the average run-time of RANSAC with each solver for a given inlier ratio ε and N data points.

Hint: For a given inlier ratio ε and a solver that needs to sample m data points in each iteration, RANSAC on average requires $\frac{1}{\varepsilon^m}$ iterations to find an all-inlier sample.

(Solution) **Solver 1**: In each iteration of RANSAC, Solver 1 needs to run (1 unit) and the resulting 4 models need to be evaluated on all N data points ($4 \cdot N \cdot 0.01$ time units). On average, using Solver 1 results in $\frac{1}{\varepsilon^3}$ RANSAC iterations, yielding a total time of

$$\text{time(Solver 1)} = \frac{1}{\varepsilon^3} \cdot (1 + 4 \cdot N \cdot 0.01) \text{ time units.}$$

Solver 2: On average, using Solver 2 results in $\frac{1}{\varepsilon^4}$ RANSAC iterations before an all-inlier sample is found. Out of these iterations, $\frac{1}{\varepsilon^4} - 1$ iterations sample at least one outlier. Thus Solver 2 returns 0 solutions for these iterations, resulting in a run-time of $\left(\frac{1}{\varepsilon^4} - 1\right) \cdot 1$ time units. In one iteration, the random sample will contain only inliers, resulting in 1 model returned by Solver 2. This iteration thus has a run-time of $1 + N \cdot 0.01$. Overall, the average time required when using Solver 2 is thus

$$\text{time(Solver 2)} = \left(\frac{1}{\varepsilon^4} - 1\right) \cdot 1 + 1 + N \cdot 0.01 = \frac{1}{\varepsilon^4} + N \cdot 0.01 \text{ time units.}$$

(b) Using your equations from (a), determine which solver is on average faster for $\varepsilon = 1$ and $N = 100$, $\varepsilon = 0.1$, and $N = 100$, and $\varepsilon = 0.1$ and $N = 1000$.

Hint: $\frac{1}{0.1} = 10$.

(Solution) **Case $N = 100$, $\varepsilon = 1$**: In this case, the run-time of the two solvers is time(Solver 1) $=$ 5 and time(Solver 2) $= 1 + 1 = 2$. Thus, using Solver 2 is faster.

Case $N = 100$, $\varepsilon = 0.1$: In this case, the run-time of the two solvers is time(Solver 1) $= \frac{1}{0.1^3} \cdot 5 = 10^3 \cdot 5 = 5000$ and time(Solver 2) $= \frac{1}{0.1^4} \cdot 1 + 1 = 10^4 \cdot 1 + 1 = 10000 + 1$. Thus, using Solver 1 is faster.

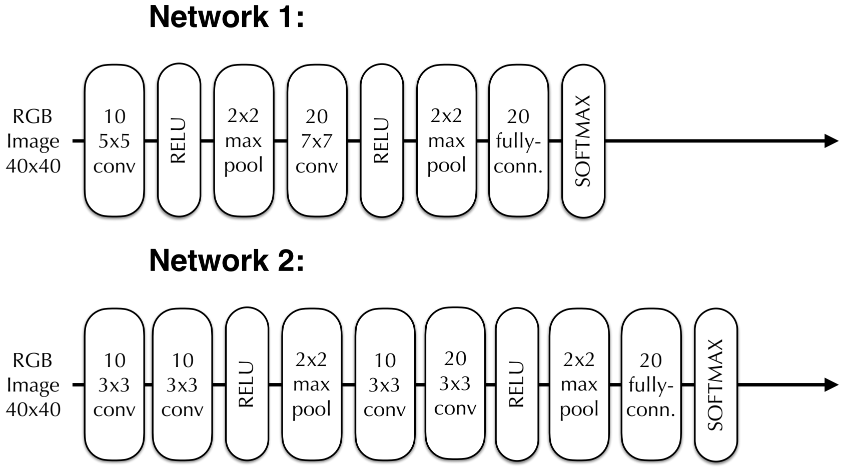
Case $N = 1000$, $\varepsilon = 0.1$: In this case, the run-time of the two solvers is time(Solver 1) $= \frac{1}{0.1^3} \cdot 41 = 10^3 \cdot 41 = 41000$ and time(Solver 2) $= \frac{1}{0.1^4} \cdot 1 + 1 = 10^4 \cdot 1 + 10 = 10000 + 10 = 10010$. Thus, using Solver 2 is faster.

[2] Out of the 4 solutions, at most one will be correct. The other solutions are introduced by the way the solver models the problem. However, you do not know which model is correct without evaluating it on all data points.

(c) Given code for Solver 1, describe how you can use this code to construct a solver with similar properties as Solver 2: Your solver should use $m = 4$ data points as input, return at most 1 model, require 1.01 time units to run, and evaluating one model on a single data point requires 0.01 time units.

(Solution) Given the 4 data points, the new solver uses 3 of the data points to run Solver 1. It then selects the model out of the 4 models returned by Solver 1 that minimizes the error of the last data point. The resulting time for this is 1.01 time units.
An even better solver can be obtained by checking whether the 4th point is an inlier to the best model and return no model at all if this is not the case.

5 Deep Learning, 3 points



(a) Consider the two convolutional neural networks shown above. Which of the two networks has fewer parameters? Assume that the convolutional layers use padding and that no bias terms are used.

Explain your answer!

(Solution) Both networks use the same number of max pooling units. Since padding is used and the number of outputs is identical for the last fully convolutional layer in each network, the input to the fully convolutional layer is of size $10 \times 10 \times 20$ for both networks. Thus, both fully convolutional layers have the same number of parameters. The difference in the number of parameters is thus only in the fully convolutional layers.

The first convolutional layer of Network 1 has $5 \cdot 5 \cdot 3 \cdot 10 = 750$ parameters ($5 \cdot 5$ from using a 5×5 filter, $\cdot 3$ as the input dimensionality is 3 (RGB), and $\cdot 10$ for the 10 filters that are applied). The second layer has $7 \cdot 7 \cdot 10 \cdot 20 = 9800$ parameters. This results in 10550 parameters.

The fully convolutional layers of Network 2 have $3 \cdot 3 \cdot 3 \cdot 10 = 270$, $3 \cdot 3 \cdot 10 \cdot 10 = 900$, $3 \cdot 3 \cdot 10 \cdot 10 = 900$, and $3 \cdot 3 \cdot 10 \cdot 20 = 1800$. This results in a total of 3870 parameters.
Even though Network 2 has more layers, it has less trainable parameters than Network 1.

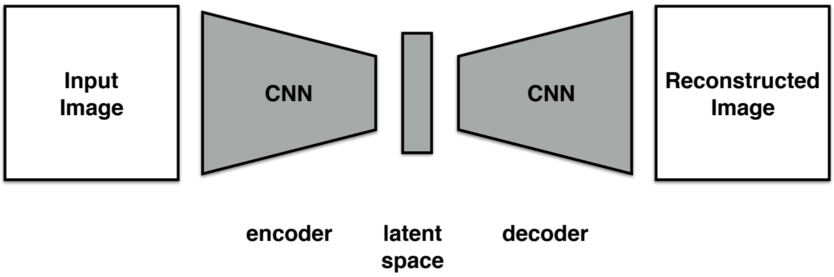
(b) Assume that you want to convert one of the two networks into a fully convolutional neural network. In order to do so, you will need to replace the fully connected layer with one convolutional layer. How many convolutional filters are needed and what is their size such that center pixel of a 40×40 input image produces the same result? In other words, you will need to apply N $k \times k$ filters. What are the values for N and k?

Explain your answer!

(Solution) In both the original Network 1 and the original Network 2, the input to the fully connected layer is a $10 \times 10 \times 20$ volume and its output is a 20 dimensional vector. In order to reproduce the results of the original networks using a convolutional layer instead of a fully connected one, the layer has to use a filter size of 10×10. As we need an output of 20 dimensions, we thus need 20 filters. Thus, we have $N = 20$ and $k = 10$.

(c) In the context of generative modelling, draw the general architecture of a Variational Autoencoder (VAE). Explain the purpose of all its components. Which parts are only needed during testing?

(Solution) The general architecture of a VAE is shown below:



A VAE consists of an encoder, which takes an image as input and maps it to a point in the latent space, and a decoder. The decoder takes a point in latent space as input and reconstructs an image from it.

During training, the task of the decoder is to reconstruct the original input image. In addition, there is a loss that encourages the points in the latent space to follow a predefined probability distribution (e.g., a Normal distribution).

During testing, one uses the predefined probability distribution to sample a point in the latent space. This point is used as input to the encoder to generate a novel image. The encoder is thus not required at test time.

6 Triangulation, 3 points

Consider two images with corresponding projection matrices $\mathtt{P}_1$ and $\mathtt{P}_2$. Using SIFT features, we have matched the 2D keypoint u_1 in image 1 with the 2D keypoint u_2 in image 2. We now want to use triangulation to compute the position of the corresponding 3D point X.

(a) Assuming that u_1 and u_2 perfectly correspond to the projection of X into image 1 and 2, respectively. Derive the geometric relations between u_1, u_2, X, $\mathtt{P}_1$, and $\mathtt{P}_2$ that need to hold.

Explain your answer and make sure that you explain all variables that you introduce.

(Solution) The following geometric relationship has to hold for both images

$$\lambda_i \begin{pmatrix} u_i \\ 1 \end{pmatrix} = \mathtt{P}_i \begin{pmatrix} X \\ 1 \end{pmatrix} \quad .$$

Here, $\mathtt{P}_i X$ projects the 3D point X to a homogenous point in image i, i.e., a point identical to $\begin{pmatrix} u_i \\ 1 \end{pmatrix}$ up to some scaling factor. λ_i is that scaling factor, corresponding to the depth of the 3D point in the local camera coordinate system.

(b) Starting from the relations in (a), derive a minimal linear solver for triangulation: Describe how to setup a system of linear equations $\mathtt{M}\theta = b$ such that solving the system for θ yields the 3D coordinates of the 3D point X.

Explain your answer and make sure that you explain all variables that you introduce. It is sufficient to explain how you obtain the linear system and what the unknown are that you need to solve for. You do not need to write down the actual equations.

(Solution) The geometric relationship derived in (a) yields 6 equations (3 per image) in five unknowns: X (3 unknowns corresponding to the 3D point position), λ_1, and λ_2. In order to obtain a minimal solver, we can thus drop one equation to get 5 equations in 5 unknowns.
Let $\mathtt{P}_i = \begin{pmatrix} a_i \\ b_i \\ c_i \end{pmatrix}$, i.e., a_i is the 1st row of the projection matrix of image i. We thus obtain 3 equations for image i:

$$\lambda_i \cdot u_i^x = a_i \begin{pmatrix} X \\ 1 \end{pmatrix} \quad ,$$

$$\lambda_i \cdot u_i^y = b_i \begin{pmatrix} X \\ 1 \end{pmatrix} \quad ,$$

$$\lambda_i = c_i \begin{pmatrix} X \\ 1 \end{pmatrix} \quad .$$

Here, u_i^x and u_i^y are the x- and y-coordinate of the 2D point u_i.

All equations are linear in the unknowns and can thus be written in the form $M_i\theta = b_i$. Stacking the matrix M_1 obtained from image 1 and the first two rows of the matrix M_2 obtained for image 2 (and the same for the right hand side) thus yields the linear system we are looking for.

(c) Consider the case of three images with corresponding projection matrices $\mathtt{P}_1$, $\mathtt{P}_2$, $\mathtt{P}_3$ and 2D point positions u_1, u_2, u_3. You know that each pair u_i and u_j $(i \neq j)$ fulfils the epipolar constraint, i.e., u_j lies on the epipolar line of u_i in image j and vice versa.
RANSAC-based triangulation using the minimal solver from (b) returns exactly 2 inliers. Can you trust the solution?

Justify your answer.

(Solution) The epipolar line for feature u_i in image j is the projection of the viewing ray of feature u_i, i.e., the ray on which the 3D point corresponding to u_i has to lie on, into image j. Thus, the viewing ray corresponding to any arbitrary point on this epipolar line will intersect the ray of u_i. Consequently, using any pair of the 3 points u_1, u_2, u_3 will always result in a triangulated point with at least 2 inliers. Thus, if RANSAC only finds 2 inliers, you cannot be sure whether the triangulated 3D point is correct.