Autonomous Vision Group
Prof. Dr.-Ing. Andreas Geiger

Eberhard Karls
Universität
Tübingen

Computer Vision

Exercise 2 – Structure-from-Motion and Stereo

1 Pen and Paper

1.1 Epipolar Geometry

a) Assume you have two cameras, both with intrinsic parameters and rotations $\mathbf{K} = \mathbf{R} = \mathbf{I}$. Then for each of the three translation vectors $\mathbf{t}_1$, $\mathbf{t}_2$ and $\mathbf{t}_3$ given below, compute the essential matrix, describe the orientation of the epipolar lines and determine location of the epipoles for the resulting camera configurations.

$$\mathbf{t}_1 = \begin{pmatrix}1\\0\\0\end{pmatrix}, \mathbf{t}_2 = \begin{pmatrix}0\\1\\0\end{pmatrix}, \mathbf{t}_3 = \begin{pmatrix}0\\0\\1\end{pmatrix},$$

The essential matrix $\tilde{\mathbf{E}}$ is given by $\tilde{\mathbf{E}} = [\mathbf{t}]_\times \mathbf{R}$. Then with

$$[\mathbf{t}_i]_\times = \begin{bmatrix} 0 & -\mathbf{t}_i^z & \mathbf{t}_i^y \\ \mathbf{t}_i^z & 0 & -\mathbf{t}_i^x \\ -\mathbf{t}_i^y & \mathbf{t}_i^x & 0 \end{bmatrix}$$

We have

$$\tilde{\mathbf{E}}_{\mathbf{t}_1} = \begin{bmatrix}0&0&0\\0&0&-1\\0&1&0\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} = \begin{bmatrix}0&0&0\\0&0&-1\\0&1&0\end{bmatrix}$$

$$\tilde{\mathbf{E}}_{\mathbf{t}_2} = \begin{bmatrix}0&0&1\\0&0&0\\-1&0&0\end{bmatrix}$$

$$\tilde{\mathbf{E}}_{\mathbf{t}_3} = \begin{bmatrix}0&-1&0\\1&0&0\\0&0&0\end{bmatrix}$$

Now that we have the essential matrices, we can reason about the corresponding epipolar lines.

For $\mathbf{t}_1$:

$$\tilde{\mathbf{l}}_2 = \tilde{\mathbf{E}}_{\mathbf{t}_1}\begin{pmatrix}x\\y\\1\end{pmatrix} = \begin{pmatrix}0\\-1\\y\end{pmatrix}, \tilde{\mathbf{l}}_1 = \tilde{\mathbf{E}}_{\mathbf{t}_1}^\top\begin{pmatrix}x\\y\\1\end{pmatrix} = \begin{pmatrix}0\\1\\-y\end{pmatrix} \rightarrow \textit{horizontal}$$

For $\mathbf{t}_2$:

$$\tilde{\mathbf{l}}_2 = \tilde{\mathbf{E}}_{\mathbf{t}_2} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -x \end{pmatrix}, \tilde{\mathbf{l}}_1 = \tilde{\mathbf{E}}_{\mathbf{t}_2}^\top \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ x \end{pmatrix} \rightarrow \textit{vertical}$$

For $\mathbf{t}_3$:

$$\tilde{\mathbf{l}}_2 = \tilde{\mathbf{E}}_{\mathbf{t}_3} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} -y \\ x \\ 0 \end{pmatrix}, \tilde{\mathbf{l}}_1 = \tilde{\mathbf{E}}_{\mathbf{t}_3}^\top \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} y \\ -x \\ 0 \end{pmatrix} \rightarrow \textit{slanted}$$

From this we can conclude that for the first two cases, because all epipolar lines must intersect at the epipole and they are parallel, the epipoles must be ideal points located at infinity.

To determine the epipole in the third case, we choose two sets of values for x and y in the epipolar line equation for each image and find the intersection point of these two epipolar lines. Consider

$$\tilde{\mathbf{l}}_2 = \begin{pmatrix} -y \\ x \\ 0 \end{pmatrix}$$

with $x = 1, y = 1$ and $x = 1, y = 2$. Then we obtain the epipole as

$$\tilde{\mathbf{e}}_2 = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} \times \begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

The epipole for the first image can be determined analogously.

b) In the third case of the previous problem, where does the baseline lie and what does that imply for the location of the epipoles?

Since we have pure translation in z-direction, the baseline lies on the optical/principal axis of the two cameras. Therefore the epipoles coincide with the principal point.

c) When is the fundamental matrix equal to the essential matrix? Discuss your reasoning.

Hint: *Think about the relationship between camera- and image coordinates.*

We arrive at the essential matrix $\tilde{\mathbf{E}}$ from the epipolar constraint

$$\tilde{\mathbf{x}}_2^\top \tilde{\mathbf{E}} \, \tilde{\mathbf{x}}_1 = 0$$

where $\tilde{\mathbf{x}}_1$, $\tilde{\mathbf{x}}_1$ are the local ray directions in camera coordinates. We can relate these to the image coordinates (in pixels) via the inverse perspective transformation of the camera model

$$\tilde{\mathbf{x}}_i = \mathbf{K}_i^{-1} \, \bar{\mathbf{x}}_i$$

This gives us

$$\tilde{\mathbf{x}}_2^\top \tilde{\mathbf{E}} \, \tilde{\mathbf{x}}_1 = \bar{\mathbf{x}}_2^\top \mathbf{K}_2^{-\top} \tilde{\mathbf{E}} \, \mathbf{K}_1^{-1} \bar{\mathbf{x}}_1 = \bar{\mathbf{x}}_2^\top \tilde{\mathbf{F}} \bar{\mathbf{x}}_1 = 0$$

where $\tilde{\mathbf{F}} = \mathbf{K}_2^{-\top} \tilde{\mathbf{E}} \, \mathbf{K}_1^{-1}$ is the fundamental matrix. From this we can see that if

$$\tilde{\mathbf{x}}_i = \bar{\mathbf{x}}_i$$

for both cameras, then $\tilde{\mathbf{F}} = \tilde{\mathbf{E}}$. In other words, if the homogenous image coordinates and the augmented pixel coordinates coincide, the fundamental matrix becomes the essential matrix. This is the case if the perspective transformation for both cameras is the identity transform

$$\mathbf{K}_1 = \mathbf{K}_2 = \mathbf{I}$$

In terms of instrinsic parameters this means we assume a camera with a focal length $f_x = f_y = 1$ and $s = 0$ and don't shift the origin to avoid negative pixel coordinates.

1.2 Triangulation

a) Consider a system of two cameras with the following intrinsics and extrinsics:

$$\mathbf{K}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{K}_2 = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix},$$
$$\mathbf{R}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{R}_2 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$
$$\mathbf{t}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \qquad \mathbf{t}_2 = \begin{pmatrix} -2 \\ 0 \\ 1 \end{pmatrix}$$

Furthermore, assume that you have observations for a point in both cameras:

$$\tilde{\mathbf{x}}_1^s = \begin{pmatrix} 1/4 \\ 1/2 \\ 1 \end{pmatrix}, \tilde{\mathbf{x}}_2^s = \begin{pmatrix} -1/5 \\ 1/5 \\ 1 \end{pmatrix}$$

For the given system, triangulate the 3D point $\tilde{\mathbf{x}}_w$ in world coordinates that corresponds to the observations. You can assume that the observations are exact.

We start by constructing the projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$. With

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}$$

we have

$$\mathbf{P}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \mathbf{P}_2 = \begin{bmatrix} -2 & 0 & 1 & -3 \\ 0 & -2 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

From these and the observations $\tilde{\mathbf{x}}_1^s$ and $\tilde{\mathbf{x}}_2^s$ we then construct a linear system as follows:

$$\underbrace{\begin{bmatrix} x_i^s \, \mathbf{p}_{i3}^\top - \mathbf{p}_{i1}^\top \\ y_i^s \, \mathbf{p}_{i3}^\top - \mathbf{p}_{i2}^\top \end{bmatrix}}_{\mathbf{A}_i} \tilde{\mathbf{x}}_w = \mathbf{0}$$

Where $\mathbf{p}_{ik}^\top$ is the k-th row of the i-th camera's projection matrix. Then

$$\mathbf{A} = \begin{bmatrix} -1 & 0 & 1/4 & 0 \\ 0 & -1 & 1/2 & 0 \\ 2 & 0 & -6/5 & 14/5 \\ 0 & 2 & -4/5 & -4/5 \end{bmatrix}$$

Solving $\mathbf{A}\tilde{\mathbf{x}}_w = 0$, for example via Gaussian Elimination, and setting the last element $\tilde{w}$ of the

homogenous vector to 1 we obtain

$$\tilde{\mathbf{x}}_w = \begin{pmatrix} 1 \\ 2 \\ 4 \\ 1 \end{pmatrix}$$

1.3 Stereo Vision

a) Show that for a stereo camera system the depth measurement error grows quadratically with depth.

Assuming that we accurately know the focal length f and baseline b from calibration, the depth z is a function of only the disparity d. We thus want to examine how z responds to small errors Δ_d in d, for example due to inaccuracies in the matching algorithm or the discretization to pixels.

The depth z is given by the relationship

$$z(d) = \frac{bf}{d}$$

Using $\frac{\delta}{\delta x}$ instead of $\frac{d}{dx}$ to denote the derivative for clarity, we can then obtain a first-order Taylor expansion

$$\begin{aligned} z(d + \Delta_d) &= z(d) + \Delta_d \frac{\delta}{\delta d} \frac{bf}{d} \\ &= z(d) - \Delta_d \frac{bf}{d^2} \\ &= z(d) - \Delta_d \frac{z^2}{bf} \end{aligned}$$

where we see that the error term grows quadratically with the depth z.

b) You can also think of this relationship as the depth resolution of the stereo camera system. How can we change the system setup to get a better depth resolution? What disadvantages might this have?

From the error term $\Delta_d \frac{z^2}{bf}$ we can see that increasing the baseline b between both cameras leads to a decreased error term and a better depth resolution in principle. However, the wider the baseline, the more challenging the matching problem becomes, potentially leading to larger errors in the estimated disparity.

1.4 Block Matching

a) Consider two $K \times K$ windows of pixels flattened to vectors $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^{K^2}$. Show that the *Zero Normalized Cross-Correlation (ZNCC)* is invariant to changes in brightness in these windows. For this you can assume changes in brightness to be linear transformations of the form $\mathbf{w}_i' = \alpha_i \mathbf{w}_i + \mathbf{1}\beta_i$, where $\mathbf{1} \in \mathbb{R}^{K^2}$ is a vector of all ones.

Let's examine the terms for the first vector $\mathbf{w}_1$ in isolation:

$$\begin{aligned}
&\frac{\left(\mathbf{w}_1' - \bar{\mathbf{w}'}_1\right)^T}{\|\mathbf{w}_1' - \bar{\mathbf{w}'}_1\|_2} \\
&= \frac{\left(\alpha_1\mathbf{w}_1 + \mathbf{1}\beta_1 - \frac{1}{K^2}\sum_k^{K^2}\left(\alpha_1\mathbf{w}_1 + \mathbf{1}\beta_1\right)\right)^T}{\|\alpha_1\mathbf{w}_1 + \mathbf{1}\beta_1 - \frac{1}{K^2}\sum_k^{K^2}\left(\alpha_1\mathbf{w}_1 + \mathbf{1}\beta_1\right)\|_2} \\
&= \frac{\left(\alpha_1\mathbf{w}_1 + \mathbf{1}\beta_1 - \frac{\alpha_1}{K^2}\sum_k^{K^2}\mathbf{w}_1 - \mathbf{1}\beta_1\right)^T}{\|\alpha_1\mathbf{w}_1 + \mathbf{1}\beta_1 - \frac{\alpha_1}{K^2}\sum_k^{K^2}\mathbf{w}_1 - \mathbf{1}\beta_1\|_2} \\
&= \frac{\alpha_1\left(\mathbf{w}_1 - \bar{\mathbf{w}_1}\right)^T}{\|\left(\alpha_1\mathbf{w}_1 - \alpha_1\bar{\mathbf{w}_1}\right)\|_2} \\
&= \frac{\alpha_1\left(\mathbf{w}_1 - \bar{\mathbf{w}_1}\right)^T}{\sqrt{\alpha_1^2\left(\mathbf{w}_1 - \bar{\mathbf{w}_1}\right)^2}} \\
&= \frac{\left(\mathbf{w}_1 - \bar{\mathbf{w}_1}\right)^T}{\|\mathbf{w}_1 - \bar{\mathbf{w}_1}\|_2}
\end{aligned}$$

It then follows that this must hold for the terms of the second vector and thus the whole expression as well.

b) You are given the following pair of 5×7 stereo images, where part of the background is occluded by a thin structure (represented by the column of 10s). Similar to the *Sum of Squared Differences (SSD)*, which you know from the lecture, we can use the *Sum of Absolute Differences (SAD)* as a similarity metric, which is defined as:

$$SAD(x, y, d) = |w_L(x, y) - w_R(x - d, y)|.$$

Using the SAD with a 3×3 window, determine the *Winner-Takes-All (WTA)* disparity $d \in \{0, 1, 2\}$ for the background pixel marked in Cyan. Discuss your result.

1	2	3	10	5	6	7
1	2	3	10	5	6	7
1	2	3	10	5	6	7
1	2	3	10	5	6	7
1	2	3	10	5	6	7

(a) Left image

2	10	4	5	6	7	8
2	10	4	5	6	7	8
2	10	4	5	6	7	8
2	10	4	5	6	7	8
2	10	4	5	6	7	8

(b) Right image

Do determine the disparity, we need to compare the reference window $\mathbf{w}_L$ located at $(3, 5)$ in the left image with three target windows $\mathbf{w}_R^{d=0}$, $\mathbf{w}_R^{d=1}$ and $\mathbf{w}_R^{d=2}$ in the right image. These are located at $(3, 5)$, $(3, 4)$ and $(3, 3)$, respectively.

- $d = 0$: $SAD(3, 5, 0) = 3 * (|10 - 5| + |5 - 6| + |6 - 7|) = 21$
- $d = 1$: $SAD(3, 5, 1) = 3 * (|10 - 4| + |5 - 5| + |6 - 6|) = 18$
- $d = 2$: $SAD(3, 5, 2) = 3 * (|10 - 10| + |5 - 4| + |6 - 5|) = 6$

According to the computed matching costs, $d = 2$ would be the optimal disparity. However, if we inspect the images, we can clearly see that the background only moved by one pixel ($d = 1$).

This is because the occluding structure in the foreground is closer to the camera rig than the background and thus moves differently, violating the fronto-parallel assumption.

c) Below we have the full disparity maps for the images from the previous exercise computed from the left- to the right image and from the right- to the left image, respectively. Perform a left-right consistency test for the pixels marked in Cyan, Green, Red, and Orange. Which of the points pass the test? Is the test succesful in determining incorrect disparity estimates?

Remark: *The disparities were computed in the same way as in the previous exercise. To compute disparities along the image boundaries, the images were padded with zeros.*

0	1	2	2	2	1	0
0	1	2	2	2	1	0
0	1	2	2	2	1	0
0	1	2	2	2	1	0
0	1	2	2	2	1	0

(a) Left → Right

2	2	2	2	1	0	0
2	2	2	2	1	0	0
2	2	2	2	1	0	0
2	2	2	2	1	0	0
2	2	2	2	1	0	0

(b) Right → Left

- Cyan: Start at (2 , 6) in (a) → As $d_{L\to R}(2,6) = 1$, move to (2, 6 - 1) in (b) → Observe that $d_{R\to L}(2,5) = 1$ and move to (2, 5 + 1) in (a). The disparities are **consistent**.
- Green: Start at (3 , 4) in (a) → As $d_{L\to R}(3,4) = 2$, move to (3, 4 - 2) in (b) → Observe that $d_{R\to L}(3,2) = 2$ and move to (3, 2 + 2) in (a). The disparities are **consistent**.
- Red: Start at (5 , 3) in (a) → As $d_{L\to R}(5,3) = 2$, move to (5, 3 - 2) in (b) → Observe that $d_{R\to L}(5,1) = 2$ and move to (5, 1 + 2) in (a). The disparities are **consistent**.
- Orange: Start at (1 , 2) in (a) → As $d_{L\to R}(1,2) = 1$, move to (1, 2 - 1) in (b) → Observe that $d_{R\to L}(1,1) = 2$ and move to (1, 1 + 2) in (a). The disparities are **not consistent**.

  Even though it passes the test, Red is actually an incorrect disparity estimate resulting from bleeding artifacts of the thin foreground structure. Orange's disparity is flagged as incorrect because its window content changes significantly, violating the fronto-parallel assumption - exposing how block matching fails at depth discontinuities.

1.5 Learned Stereo and End-to-End Models

a) Recent approaches in end-to-end disparity estimation often build a disparity cost volume and apply 3D convolutions to it to estimate the final disparity map. However, 3D convolutions are computationally expensive, limiting both the resolution and maximum disparity that can be used.

Below we consider two sequences of two 2D- and 3D convolutions, respectively, applied to the same input tensor. We describe the layer configurations as ConvND(C_{in}, C_{out}, k), where input channels are denoted by C_{in}, output channels by C_{out} and k is the kernel size. For simplicity you can assume square kernels, as well as appropriate padding and a stride of one, such that the spatial dimensions remain unchanged. Shapes are specified by the number of channels followed by the spatial dimensions, so (Channels, Height, Width) for 2D convolutions and (Channels, Depth, Height, Width) for 3D convolutions.

For both sequences, calculate the total amount of memory required to store the activations and trainable parameters. For this you can fill in the blank fields in the table below. You can assume that all activations and trainable parameters are stored as 32-bit floating point numbers.

Layer	Input Shape	Output Shape	# Trainable Parameters	Memory [MiB]
Conv2D$(16, 32, 3)$ Conv2D$(?, 128, 3)$	(16, 128, 128)			
Total(2D)				
Conv3D$(1, 32, 3)$ Conv3D$(?, 64, 3)$	(1, 32, 128, 128)			
Total(3D)				

Layer	Input Shape	Output Shape	# Trainable Parameters	Memory [MiB]
Conv2D$(16, 32, 3)$	$(16, 128, 128)$	$(32, 128, 128)$	$3^2 \times 16 \times 32 + 32 = 4640$	Activations: $(32 \times 128 \times 128 \times 4)/1024^2 = 2.0$ MiB Params: $(4,640 \times 4)/1024^2 = 0.02$ MiB Total: 2.0 MiB + 0.02 MiB = 2.02 MiB
Conv2D$(32, 128, 3)$	$(32, 128, 128)$	$(128, 128, 128)$	$3^2 \times 32 \times 128 + 128 = 36992$	Activations: $(128 \times 128 \times 128 \times 4)/1024^2 = 8.0$ MiB Params: $(36992 \times 4)/1024^2 = 0.14$ MiB Total: 8.0 MiB + 0.14 MiB = 8.14 MiB
Total (2D)			41632 parameters	Activations: 10.0 MiB Params: 0.16 MiB Total: 10.0 MiB + 0.16 MiB = **10.16 MiB**
Conv3D$(1, 32, 3)$	$(1, 32, 128, 128)$	$(32, 32, 128, 128)$	$3^3 \times 1 \times 32 + 32 = 896$	Activations: $(32 \times 32 \times 128 \times 128 \times 4)/1024^2 = 64.0$ MiB Params: $(896 \times 4)/1024^2 = 0.0032$ MiB Total: 64.0 MiB + 0.0034 MiB = 64.0034 MiB
Conv3D$(32, 64, 3)$	$(32, 32, 128, 128)$	$(64, 32, 128, 128)$	$3^3 \times 32 \times 64 + 64 = 55360$	Activations: $(64 \times 32 \times 128 \times 128 \times 4)/1024^2 = 128.0$ MiB Params: $(55,360 \times 4)/1024^2 = 0.22$ MiB Total: 128.0 MiB + 0.22 MiB = 128.22 MiB
Total (3D)			56256 parameters	Activations: 192.0 MiB Params: 0.22 MiB Total: 192.0 MiB + 0.22 MiB = **192.22 MiB**

Remark: As we are only interested in the activations and trainable parameters, we neglect the memory required to store the input tensor to the first layer of both sequences.

b) You are working with a GC-Net-style architecture to solve a disparity estimation problem. For two particular pixels p_1 and p_2 in the cost volume, the network estimates the following matching costs:

- p_1: $c_\theta(d) = [1.0, 3.0, 10.0, 3.0, 1.0]$
- p_2: $c_\theta(d) = [10.0, 2.0, 1.0, 2.0, 10.0]$

where $d \in \{0, 1, 2, 3, 4\}$. For both pixels, calculate the expectation of the disparity and discuss the result. GC-Net uses the discrepancy between the expected and the ground-truth disparity as a loss function during training. What kind of behaviour does this encourage?

Hint: *Compare the distributions over disparities implied by the cost vectors with the final result.*

Recall that the softmax is defined as : $\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

Then for p_1:
$\sigma(-c_\theta) = [0.44, 0.06, 0.00, 0.06, 0.44]$
$E[d] = \sum_{d=0}^{D} \sigma(-c_\theta) \cdot d = 2$

And for p_2:
$\sigma(-c_\theta) = [0.00, 0.21, 0.58, 0.21, 0.00]$
$E[d] = \sum_{d=0}^{D} \sigma(-c_\theta) \cdot d = 2$

For p_1, the matching cost is similarly low for two disparity values (it is *bimodally* distributed). This could be the case at depth discontinuities/object boundaries, where the disparities of both the foreground and background are plausible solutions. However, when taking the expectation, this bimodal structure is lost. Thus, using the discrepancy between

the expected- and the ground-truth disparity will encourage the network predictions to be *unimodally* distributed.

2 Coding Exercises

The following coding exercises are split into two sections: A structure-from-motion and a stereo section, with corresponding jupyter notebooks `code/sfm/sfm.ipynb` and `code/stereo/stereo.ipynb`. As in the last exercise, the notebooks are self-contained but you can also use this document as guidance. If you are stuck, you can find *Hints* in the notebooks themselves which are written upside-down.

2.1 Structure-From-Motion

a) Implement the function `compute_fundamental_matrix` which takes sets of corresponding keypoints $\bar{\mathbf{x}}_i$ from the first and second image as input and returns the fundamental matrix $\tilde{\mathbf{F}}$ using the 8-point algorithm.

b) Implement the function `compute_fundamental_matrix_normalized` which again takes sets of corresponding keypoints $\bar{\mathbf{x}}_i$ from the first and second image as input and returns the fundamental matrix $\tilde{\mathbf{F}}$, but this time using the *normalized* 8-point algorithm.

c) Implement the function `compute_essential_matrix` which takes the fundamental matrix $\tilde{\mathbf{F}}$ as well as the intrinsics $\mathbf{K}_i$ for the first and second image as input and returns the essential matrix $\tilde{\mathbf{E}}$.

d) Implement the function `triangulate_point`. This function takes keypoints $\bar{\mathbf{x}}_i$, intrinsics $\mathbf{K}_i$ as well as the relative rotation $\mathbf{R}$ and translation $\mathbf{t}$ from two images as input and returns a triangulated 3D point $\bar{\mathbf{x}}_w$.

2.2 Stereo

a) Implement the function `sad`, which given a window size and maximum disparity D, takes a stereo image pair as input and returns a disparity map, computed from the left to the right image. If you are interested you can also implement the bonus function `sad_convolve`, which should be significantly faster than `sad`.

b) Create a visualization of the computed disparites by implementing the function `visualize_disparity`. It's a good idea to also visualize the input images to see if the results are sensible.

c) Experiment with different window sizes (for example 3, 7, 15) and report which one leads to better visual results and why? In case you were not able to solve the previous exercises, you can use the provided disparity maps in the `code/stereo/examples/` folder.

d) Why do you think the block matching approach fails to lead to good estimations around homogeneous regions such as the road?

e) Develop a Siamese Neural Network architecture. For this you can use the `StereoMatchingNetwork` class. In particular, implement the `__init__` and `forward` methods to initialize the layers and define the forward pass. Details on the architecture can be found in the jupyter notebook.

f) From the lecture you know two classes of Siamese Neural Network architectures - which class does the architecture you implemented in `StereoMatchingNetwork` belong to?

g) Implement the function `calculate_similarity_score`, which takes an instance of the Siamese Neural Network as well as patches $\mathbf{w}_L$ and $\mathbf{w}_R$ from the left and right image as input and returns the similarity of those patches. The similarity should be computed based on features extracted by the Siamese Network.

h) Try to improve the network by finding better hyperparameters. For example, you can vary the number of training iterations or the number of filters in the convolutional layers. Explain your findings.

i) Compare the visualization of the disparity maps from the Siamese Neural Network to the ones obtained by the block matching algorithm. Which predictions are better and why? Can you find regions in the scenes where the differences in predictions are most dominant? (If you were not able to solve the previous exercises, you can use the provided disparity maps in the `code/stereo/examples/` folder.)