



Multiple View Geometry: Exercise Sheet 5

Prof. Dr. Daniel Cremers, Christiane Sommer, Rui Wang
Computer Vision Group, TU Munich

<http://vision.in.tum.de/teaching/ss2017/mvg2017>

Exercise: June 8th, 2017

Part I: Theory

This part of the exercises should be **solved at home**.

1. The Lucas-Kanade method

The weighted Lucas-Kanade energy $E(\mathbf{v})$ is defined as

$$E(\mathbf{v}) = \int_{W(\mathbf{x})} G(\mathbf{x} - \mathbf{x}') \left\| \nabla I(\mathbf{x}', t)^\top \mathbf{v} + \partial_t I(\mathbf{x}', t) \right\|^2 d\mathbf{x}'.$$

Assume that the weighting function G is chosen such that $G(\mathbf{x} - \mathbf{x}') = 0$ for any $\mathbf{x}' \notin W(\mathbf{x})$.

- (a) Prove that the minimizer \mathbf{b} of $E(\mathbf{v})$ can be written as

$$\mathbf{b} = -M^{-1}\mathbf{q}$$

where the entries of M and \mathbf{q} are given by

$$m_{ij} = G * (I_{x_i} \cdot I_{x_j}) \quad \text{and} \quad q_i = G * (I_{x_i} \cdot I_t)$$

- (b) Show that if the gradient direction is constant in $W(\mathbf{x})$, i.e. $\nabla I(\mathbf{x}', t) = \alpha(\mathbf{x}', t)\mathbf{u}$ for a scalar function α and a 2D vector \mathbf{u} , M is not invertible.
Explain how this observation is related to the aperture problem.
- (c) Write down explicit expressions for the two components b_1 and b_2 of the minimizer in terms of m_{ij} and q_i .

2. The Reconstruction Problem

The bundle adjustment (re-)projection error for N points $\mathbf{X}_1, \dots, \mathbf{X}_N$ is

$$E(R, \mathbf{T}, \mathbf{X}_1, \dots, \mathbf{X}_N) = \sum_{j=1}^N \left(\|\mathbf{x}_1^j - \pi(\mathbf{X}_j)\|^2 + \|\mathbf{x}_2^j - \pi(R\mathbf{X}_j + \mathbf{T})\|^2 \right)$$

- (a) What dimension does the space of unknown variables have if ...
- ... R is restricted to a rotation about the camera's y -axis?
 - ... the camera is only rotated, not translated?
 - ... the points \mathbf{X}_j are known to all lie on one plane?

In contrast to the projection error, the 8-point algorithm decouples the rigid body motion from the coordinates \mathbf{X}_j .

- (b) Which constrained optimization problem does the 8-point algorithm solve? Write down a cost function $E_{8\text{-pt}}(R, \mathbf{T})$ and according constraints using $\mathbf{x}_1^j, \mathbf{x}_2^j, R$ and \mathbf{T} .
- (c) Can the 8-point algorithm be used if ...
- ... R is restricted to a rotation about the camera's y -axis?
 - ... the camera is only rotated, not translated?
 - ... the points \mathbf{X}_j are known to all lie on one plane?

Part II: Practical Exercises

This exercise is to be solved **during the tutorial**.

1. The Structure Tensor

In order to be able to detect corners in an image and compute optical flow, the structure tensor M shall be computed in this exercise. Write a function `[M11, M12, M22] = getM(I, sigma)` that computes the entries of the structure tensor M (see Theory, Ex.1(a)) for every pixel (x, y) of an image I . Proceed as follows:

- Compute the image gradients I_x and I_y using central differences.
- As weighting function use a two-dimensional Gaussian Kernel with a standard deviation σ (see Exercise Sheet 1). Use a kernel size (and hence integration window size) $k = 2 \cdot 2\sigma + 1$.
- Use the result from Theory, Ex.1(a) and the Matlab function `conv2` to compute `M11`, `M12` and `M22`. Why is it not necessary to compute `M21`?

2. Corner Detection

In this exercise you will implement the Harris corner detector. Download `ex5.zip` and extract its content.

- Fill in the first missing part in `getHarrisCorners.m`: compute the scoring function $C := \det(M) - \kappa \text{trace}^2(M)$ for each pixel (x, y) .
- Fill in the second part: visualize the scoring function. It is a good idea to display a non-linearly transformed scoring function, e.g. $\text{sign}(C) \cdot |C|^{\frac{1}{4}}$.
- Complete `getHarrisCorners.m`: find all pixels (x, y) for which $C(x, y) > \theta$, and which are a local maximum of the scoring function, i.e. all four adjacent pixel have a lower score (non-maximum suppression).
- Run `getHarrisCorners` for `img1.png` with $\sigma = 2$, $\kappa = 0.05$ and $\theta = 10^{-7}$. Display the found corners using the provided function `drawPts`.
- Try different values for σ . What do you observe?

3. Dense Optical Flow

In this exercise you will implement the Lucas-Kanade method to compute optical flow. To this end, complete the missing parts in `getFlow.m`.

- Write a function `[M11, M12, M22, q1, q2] = getMq(I1, I2, sigma)` that computes the entries of the structure tensor M as well as the vector \mathbf{q} for every pixel (x, y) . The easiest way to do this is to copy `getM.m` and modify it accordingly.
- Compute the local velocity (v_x, v_y) of each pixel using the formula derived in Theory, Ex.1(a). Use the results from Theory Ex.1(c) to avoid loops and thus make your code efficient.
- Run `getFlow` for the two images `img1.png` and `img2.png` and $\sigma = 2$.
- Create a figure with three subplots. Visualize the two velocities separately using `imagesc`. In the third subplot, display a quiver plot of the velocities. Use `help quiver` if you do not know the syntax.
- Again, try different values for σ . What do you observe now?