Lucas - Kanade Tracking

This laboratory session is about the local tracking algorithm proposed by Lucas and Kanade in [2]. We will follow the nine steps presented in slides 59 and 65 (of 79) of the lecture notes, which are extracted from the detailed description of the tracking algorithm in [1] (shown in Figs. 1 and 2).

The Lucas-Kanade Algorithm

Iterate:

(1) Warp I with $\mathbf{W}(\mathbf{x};\mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
(2) Compute the error image $T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
(3) Warp the gradient ∇I with $\mathbf{W}(\mathbf{x};\mathbf{p})$
(4) Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x};\mathbf{p})$
(5) Compute the steepest descent images $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
(6) Compute the Hessian matrix using Equation (11)
(7) Compute $\sum_{\mathbf{x}} [\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$
(8) Compute $\Delta\mathbf{p}$ using Equation (10)
(9) Update the parameters $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$

until $\|\Delta\mathbf{p}\| \leq \epsilon$

Figure 1. The Lucas-Kanade algorithm (Lucas and Kanade, 1981) consists of iteratively applying Eqs. (10) and (5) until the estimates of the parameters $\mathbf{p}$ converge. Typically the test for convergence is whether some norm of the vector $\Delta\mathbf{p}$ is below a user specified threshold ϵ. Because the gradient ∇I must be evaluated at $\mathbf{W}(\mathbf{x};\mathbf{p})$ and the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ must be evaluated at $\mathbf{p}$, all 9 steps must be repeated in every iteration of the algorithm.

Figure 1: Fig. 1 in reference [1].

The provided MATLAB code consists of two files, which are adapted from those written by D. Kroon (Univ. of Twente):

- `LucasKanadeAffine.m` : a function that implements a template tracker for one image given a template and an initial estimate of the warping parameters.
- `TTdemo.m` : a script that loads a dataset (movie), selects several rectangles in the first image as templates and tracks them across the image sequence by calling to the function `LucasKanadeAffine.m` (one time per frame and template).

In this exercise you are required to complete the code in function `LucasKanadeAffine.m`. In the end, by running the script `TTdemo.m`, the results of the implemented tracker should be similar to those displayed in Fig. 3, where four templates are tracked across an image sequence.

Recall that the Lucas-Kanade tracker optimizes the warping parameters $\mathbf{p}$ between a template and an image using a Gauss-Newton approach, which approximates the error function locally by a quadratic model and then finds its minimizer. Since the approximate error model is quadratic, its

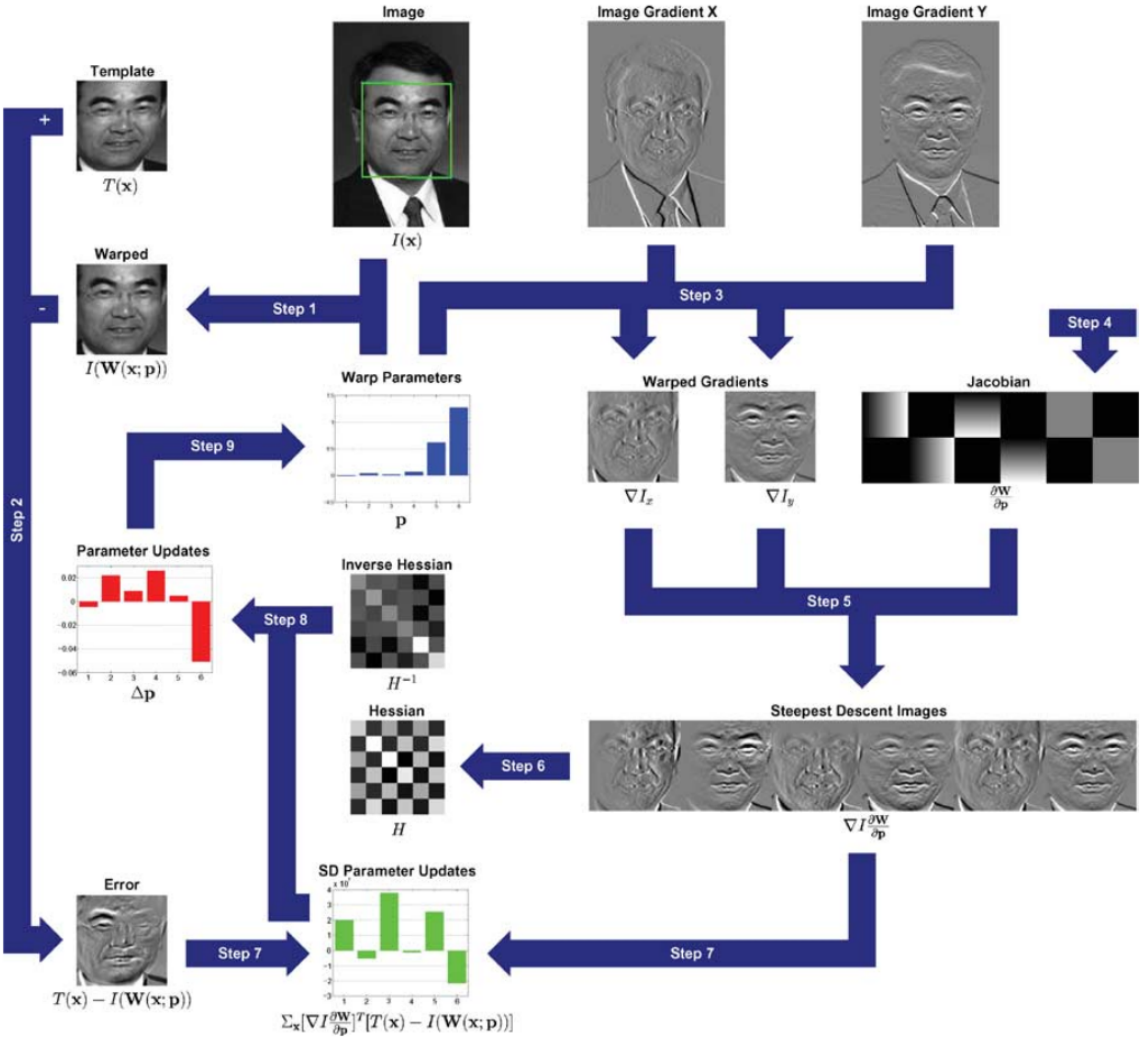

Figure 2. A schematic overview of the Lucas-Kanade algorithm (Lucas and Kanade, 1981). The image I is warped with the current estimate of the warp in Step 1 and the result subtracted from the template in Step 2 to yield the error image. The gradient of I is warped in Step 3, the Jacobian is computed in Step 4, and the two combined in Step 5 to give the steepest descent images. In Step 6 the Hessian is computed from the steepest descent images. In Step 7 the steepest descent parameter updates are computed by dot producting the error image with the steepest descent images. In Step 8 the Hessian is inverted and multiplied by the steepest descent parameter updates to get the final parameter updates $\Delta \mathbf{p}$ which are then added to the parameters $\mathbf{p}$ in Step 9.

Figure 2: Fig. 2 in reference [1].

Figure 3: Results of tracking four templates (each of them displayed with a different color) across the image sequence, which is displayed in temporal order, from left to right and from top to bottom.

first derivative (i.e., gradient) is linear, and so the minimizer, obtained by setting to zero the first derivative of the model, is given by a linear system of equations. In our case, the linear system is

$$\underbrace{\left(\sum_{\mathbf{x}\in T}\left(\nabla I\frac{\partial W}{\partial \mathbf{p}}\right)^{\top}\left(\nabla I\frac{\partial W}{\partial \mathbf{p}}\right)\right)}_{H}\Delta\mathbf{p}=\underbrace{\sum_{\mathbf{x}\in T}\left(\nabla I\frac{\partial W}{\partial \mathbf{p}}\right)^{\top}(T(\mathbf{x})-I(W(\mathbf{x};\mathbf{p})))}_{\mathbf{b}}, \tag{1}$$

that is, more compactly, $H\,\Delta\mathbf{p}=\mathbf{b}$, where H is the so-called Hessian matrix. The solution of the linear system, i.e., the computation of the update $\Delta\mathbf{p}=H^{-1}\mathbf{b}$ (step 8 of Fig. 1) is the heart of the method.

Fill in the missing code

Take a look at the files `TTdemo.m` and `LucasKanadeAffine.m`. You are asked to fill in the missing code in function `LucasKanadeAffine.m`. These follow the nine steps in Fig. 1. Specifically, you have to translate into code steps

- **2**: Compute the error image by subtracting $I(W(\mathbf{x};\mathbf{p}))$ from $T(\mathbf{x})$.
  `I_error = ...;`
- **3**: Compute the warped gradient $\nabla I\big(W(\mathbf{x};\mathbf{p})\big)$ (horizontal and vertical component). Write code similar to that provided in step 1 to compute the warped image $I(W(\mathbf{x};\mathbf{p}))$:
  `I_warped = affine_transform_2d_double(I,x,y,W_xp);`
- **6**: Compute the Hessian matrix H

  $$H=\sum_{\mathbf{x}\in T}\left(\nabla I\frac{\partial W}{\partial \mathbf{p}}\right)^{\top}\left(\nabla I\frac{\partial W}{\partial \mathbf{p}}\right),$$

  where $\left(\nabla I\frac{\partial W}{\partial \mathbf{p}}\right)$ for the j-th pixel $\mathbf{x}$ is stored in the the variable `I_steepest(j,:)`.

- **7**: Compute the right-hand-side vector of the linear system of equations

$$\mathbf{b} = \sum_{\mathbf{x} \in T} \left(\nabla I \frac{\partial W}{\partial \mathbf{p}} \right)^{\top} \left(T(\mathbf{x}) - I(W(\mathbf{x}; \mathbf{p})) \right),$$

  where $\left(\nabla I \frac{\partial W}{\partial \mathbf{p}} \right)$ for the j-th pixel $\mathbf{x}$ is stored in the the variable `I_steepest(j,:)` and $T(\mathbf{x}) - I(W(\mathbf{x}; \mathbf{p}))$ is stored in the j-th component of `I_error` (computed in step 2). Vector $\mathbf{b}$ is stored in variable `sum_xy`.

- **8**: Compute $\Delta\mathbf{p}$ by solving the linear system of equations (1). In MATLAB, the best method to solve linear systems is to use the backslash command \. That is, to solve $Ax = b$, we use the command `x=A\b;` See also:
  `>> doc mldivide`

- Finally, copy the lines of code from steps 6, 7 and 8 to the corresponding lines after the '`else`' statement, which implement the optimization in the simple and commonly used type of affine transformation: translations.

Run the script `TTdemo.m`, which is provided to test the Lucas-Kanade tracker. Feel free to select different regions of the image as templates to be tracked and/or to change the parameter passed to function `LucasKanadeAffine`, such as the number of iterations (translation, affine transformation), the amount of smoothing of the derivatives, etc.

References

[1] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *Int. J. Computer Vision*, 56(3):221–255, 2004.

[2] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 674–679, 1981.