Lecture 11: Tracking

1 Tracking

1.1 Point Tracking

Problem: Given two images, estimate the motion of a pixel from image I_0 to image I_1. Two approaches exist, depending on the amount of motion between the frames:

- **Block-based methods**
- **Differential methods**

Template Tracking: Given two images, estimate the warping that defines the motion and/or the distortion of a template from image I_0 to image I_1.

1.1.1 Block-based methods

The method can be represented by the following points:

- Search for the corresponding patch in a *neighborhood* around the point.
- Use SSD, SAD, NCC to search for corresponding patches in a local neighborhood of the point. The search region usially is a $D \times D$ squared patch. We have to perform $D \times D$ comparisons, computationally demanding.

1.1.2 Differential Methods

One looks at the local brightness changes at the **same** location. **NO patch shift** is performed. (centered in the same point!)

1.1.3 Spatial Coherency

We assume that all the pixels in the patch undergo the same motion (same u and v). Also, assume that the time interval between the two images I_0 and I_1 is small. We want to find the motion vector (u, v) that minimizes the Sum of Squared Differences (SSD). As we did for Harris, we look at the first order Taylor approximation of the sum:

$$\begin{aligned} SSD &= \sum (I_0(x,y) - I_1(x+u, y+v))^2 \\ &\approx \sum (I_0(x,y) - I_1(x,y) - I_x \cdot u - I_y \cdot v)^2 \\ &= \sum (\Delta I - I_x \cdot u - I_y \cdot v)^2, \end{aligned} \tag{1.1}$$

which is a simple quadratic function in two variables (u, v).

1.1.4 Motion Vector

To minimize the error, we differentiate with respect to (u, v) and equate to 0.

$$\begin{aligned} \frac{\partial E}{\partial u} = 0 \Rightarrow -2I_x \sum(\Delta I - I_x \cdot u - I_y \cdot v) = 0, \\ \frac{\partial E}{\partial v} = 0 \Rightarrow -2I_y \sum(\Delta I - I_x \cdot u - I_y \cdot v) = 0. \end{aligned} \tag{1.2}$$

One can get a linear system of two equations in two unknowns. We can write this in matrix form

$$\begin{aligned} \begin{pmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{pmatrix} \cdot \begin{pmatrix} u \\ v \end{pmatrix} &= \begin{pmatrix} \sum I_x \cdot \Delta I \\ \sum I_y \cdot \Delta I \end{pmatrix} \\ \begin{pmatrix} u \\ v \end{pmatrix} &= \underbrace{\begin{pmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{pmatrix}}_{M^{-1}} \cdot \begin{pmatrix} \sum I_x \cdot \Delta I \\ \sum I_y \cdot \Delta I \end{pmatrix} \end{aligned} \tag{1.3}$$

These are not matrix products, but pixel-wise products! For M to be invertible, its determinant should be different from 0. From the decomposition

$$M = R^{-1} \cdot \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \cdot R, \tag{1.4}$$

we know that $\det(M)$ is non zero when its eigenvalues are large (i.e. not a flat region and not an edge). In practice, it should be a corner or in general contain **texture**.

1.1.5 Aperture Problem

If we look at local brightness changes through a small aperture, we cannot always determine the motion direction, becuase **infinite** motion directions (solutions) may exist. The solution is to **increase** the aperture size.

1.1.6 Application of Differential Methods: Optical Flow

Optical flow is the pattern of appearent motion of objects in a visual scene, caused by the relative motion between the observer (eye or camera) and the scene. It tracks the motion of every pixel between two consecutive frames. For each pixel, we compute

- the vector direction and,
- the vector length (amount of movement).

An issue could be the choice of the right patch size.

1.1.7 Block-based vs. Differential Methods

Block-based Methods: search for the corresponding patch in a neighborhood of the point to be tracked. The search region is usually a square of $n \times n$ pixels.

- +: **robust to large motions**
- -: Can be computationally expensive ($n \times n$ comparisons for a single point track)

Differential Methods:

- +: Much more efficient than block-based methods. Thus, can be used to track the motion of every pixel in the image. It avoids searching in the neighborhood of the point by analyzing the **local intensity changes** of an image patch at a specific location (no search is performed).
- -: Works only for **small** motions (high frame rate). For larger motion, multiscale implementations are used, but are then expensive.

1.2 Transformations Review

- **Translation**: 2 DOF

$$W(x,p) = \begin{pmatrix} x + a_1 \\ y + a_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & a_1 \\ 0 & 1 & a_2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{1.5}$$

- **Euclidean**: 3 DOF

$$W(x,p) = \begin{pmatrix} x\cos(\alpha) - y\sin(\alpha) + a_1 \\ x\sin(\alpha) + y\cos(\alpha) + a_2 \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & \sin(\alpha) & a_1 \\ \sin(\alpha) & \cos(\alpha) & a_2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{1.6}$$

- **Affine**: 6 DOF

$$W(x,p) = \begin{pmatrix} a_1 x + a_3 y + a_5 \\ a_2 x + a_4 y + a_6 \end{pmatrix} = \begin{pmatrix} a_1 & a_3 & a_5 \\ a_2 & a_4 & a_6 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{1.7}$$

- **Projective (homography)**: 8 DOF

$$\begin{aligned} x' &= \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}, \\ y' &= \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}. \end{aligned} \tag{1.8}$$

The Jacobian of a function

$$F(x_1, x_2, \ldots, x_n) = \begin{pmatrix} f_1(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_m(x_1, x_2, \ldots x_n) \end{pmatrix} \tag{1.9}$$

is

$$J(F) = \nabla F = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ & \vdots & \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \tag{1.10}$$

1.3 Template Tracking

Follow a template image in a video sequence by estimating the warp. Basically, we track the **distortion** function.

1.3.1 Problem Formulation

Template Warping: Given the template image $T(x)$, take all the pixels from the template image and warp them using the function $W(x,p)$ parameterized in terms of parameters p. The goal of template-based tracking is to find the set of warp parameters p such that

$$I(W(x,p)) = T(x). \tag{1.11}$$

This is solved by determining p that minimizes the Sum of Squared Differences

$$E = SSD = \sum_{x \in T} \left[I(W(x,p)) - T(x) \right]^2. \tag{1.12}$$

Assumptions are:

- No errors in the template image boundaries: only the appearance of the object to be tracket appears in the template image.
- No occlusion: the entire template is visible in input image.
- Brightness consistency assumption: the intensity of the object appearance is always the same across different views.

1.3.2 Lucas-Kanade Tracker

Uses the Gauss-Newton method for minimization, i.e.

- Applies a first order approximation of the warp,
- Attempts to minimize the SSD iteratively.

Derivation: Starting from

$$E = SSD = \sum_{x \in T} \left[I(W(x,p)) - T(x) \right]^2, \tag{1.13}$$

we assume that an initial estimate of p is **known**. Then, we want to find the increment Δp that minimizes

$$\sum_{x \in T} \left[I(W(x,p+\Delta p)) - T(x) \right]^2. \tag{1.14}$$

The first order Taylor approximation of the term in brackets reads

$$I(W(x,p+\Delta p)) \approx I(W(x,p)) + \nabla I \frac{\partial W}{\partial p} \Delta p. \tag{1.15}$$

By replacing that in the equation we get

$$E = \sum_{x \in T} \left[I(W(x,p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) \right]^2. \tag{1.16}$$

In order to minimize it, we differentiate and equate to 0, i.e.

$$\frac{\partial E}{\partial \Delta p} = 0. \tag{1.17}$$

It holds

$$\begin{aligned} \frac{\partial E}{\partial \Delta p} &= 0 \\ 2\sum_{x\in T}\left(\nabla I\frac{\partial W}{\partial p}\right)^T\left[I(W(x,p)) + \nabla I\frac{\partial W}{\partial p}\Delta p - T(x)\right] &= 0 \\ \Delta p = H^{-1}\sum_{x\in T}\left(\nabla I\frac{\delta W}{\delta p}\right)^T &(T(x) - I(W(x,p))), \end{aligned} \tag{1.18}$$

where $H = \sum_{x\in T}\left(\nabla I\frac{\partial W}{\partial p}\right)^T\left(\nabla I\frac{\partial W}{\partial p}\right)$ is the second moment matrix of the warped image (Hessian). Summing up, the algorithm reads

1. Warp $I(x)$ with $W(x,p)$.
2. Compute the error.
3. Compute **warped** gradients, ∇I evaluated at $W(x,p)$.
4. Evaluate the Jacobian of the warping $\frac{\partial W}{\partial p}$.
5. Compute the inverse Hessian H^{-1}.
6. Multiply steepest descent with error.
7. Comptue Δp.
8. Update parameters $p \leftarrow p + \Delta p$.
9. Repeat until $\Delta p < \varepsilon$.

The algorithm follows a predict-correct cycle. A prediction $I(W(x,p))$ of the warped image is computed from an initial estimate. The correction parameter Δp is computed. The larger the error, the larger the correction. Challenges are

- How to get initial p?
- If the initial estimate is too far, the linear approximation does not longer hold. Solution are **pyramidal implementations**
- Illumination changes, object deformations,
- Occlusions,
- A solution can be to update the template with newest image.

Coarse to Fine estimation - Pyramidal implementations

Because the small motion assumption, regular optical flow methods work bad if the object we are tracking moves a long distance. Building image pyramids for each image and doing optical flow on each layer of the pyramid (to get rid of small motion constraints). Decrease resolution to get smaller image.

Generalization of Lucas-Kanade

The same concept (predict/correct) can be applied to tracking of 3D objects. In order to deal with wrong prediction, it can be implemented in a Particle-Filter fashion.

Tracking by detection of local image features

1. Keypoint detection and matching. Invariant to scale, rotation or perspective.
2. Geometric verification (RANSAC)

Issues are:

- How to segment the object to track from background?
- How to initialize the warping?
- How to handle occlusions?
- How to handle illumination changes andn on modeled effects?

1.4 Understanding Check

Are you able to answer the following questions?

- *Are you able to illustrate tracking with block matching?*
- *Are you able to explain the underlying assumptions behind differential methods, derive their mathematical expression and the meaning of the M matrix?*
- *When is this matrix invertble and when not?*
- *What is the aperture problem and how can we overcome it?*
- *What is optical flow?*
- *Can you list pros and cons of block-based vs. differential methods for tracking?*
- *Are you able to describe the working principle of KLT?*
- *Are you able to derive the main mathematical expression for KLT?*
- *What is the Hessian matrix and for which warping function does it coincide to that used for point tracking?*
- *Can you list Lukas-Kanade failure cases and how to overcome them?*
- *How does one guess the initial guess?*
- *Could you illustrate the coarse-to-fine Lucas-Kanade implementation?*
- *Could you illustrate alternative tracking procedures using point features?*