Lecture 15, Oct 24, 2025

Optical Flow and Motion Tracking

- *Dense motion estimation* or *optical flow* is the process of estimating a motion vector (velocity) for every possible pixel in the image from a series of images
  - This can be applied in video compression, e.g. MPEG, H.263/4
  - Often regularization is needed to fill in missing pixels
- We need to define a error metric, similar to feature matching, e.g. SAD, SSD
  - Often subpixel accuracy is important (since we are defining very small vectors), so we use interpolation
  - Then use a search technique to identify motion in a pair of images
- Applications of optical flow includes motion estimation, moving object tracking, UAVs, optical mice, etc
- The most challenging version of the problem is to compute motion at each pixel independently over time to obtain an *optical flow field*
  - We rely on the *brightness constancy assumption*: over a time Δt, there is another pixel in the next image that has the same brightness, i.e. $I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$
    - $\Delta x, \Delta y$ is the optical flow vector that we are looking for; the smaller the Δt, the closer this assumption is to reality
    - Using a Taylor expansion, $I(x + \Delta x, y + \Delta y, t + \Delta t) \approx I(x, y, t) + \dfrac{\partial I}{\partial x}\Delta x + \dfrac{\partial I}{\partial y}\Delta y + \dfrac{\partial I}{\partial t}\Delta t$
    - Therefore $\dfrac{\partial I}{\partial x}\dfrac{\Delta x}{\Delta t} + \dfrac{\partial I}{\partial y}\dfrac{\Delta y}{\Delta t} + \dfrac{\partial I}{\partial t}\dfrac{\Delta t}{\Delta t} = 0$
  - The flow equation is $\dfrac{\partial I}{\partial x}v_x + \dfrac{\partial I}{\partial y}v_y + \dfrac{\partial I}{\partial t} = 0$ where v_x, v_y are the pixel velocities
  - In vector form, $I_x v_x + I y v_y = \Delta I^T \boldsymbol{v} = -I_t$
    - Physically we can interpret this as looking at the "flow of intensity" into and out of a pixel
- Due to the aperture problem, this is ill-posed, so we need some additional information in the neighbourhood to actually compute this
  - The *Lucas-Kanade method* assumes that a local patch has the same flow, so we compute the velocity for an entire patch
    - This can be solved using least squares, by constructing a matrix equation for the entire patch, assuming the same flow
    - $\begin{bmatrix} I_x(\boldsymbol{p}_1) & I_y(\boldsymbol{p}_1) \\ \vdots & \vdots \\ I_x(\boldsymbol{p}_N) & I_y(\boldsymbol{p}_N) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\boldsymbol{p}_1) \\ \vdots \\ I_t(\boldsymbol{p}_N) \end{bmatrix} \iff \boldsymbol{Ad} = \boldsymbol{b}$
    - $\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} \iff (\boldsymbol{A}^T\boldsymbol{A})\boldsymbol{d} = \boldsymbol{A}^T\boldsymbol{b}$
  - The aperture problem can make motion seem as if it's in a different direction if we only look at a small part of the image (think the *barber pole illusion*)
  - This means it's often a good idea to incorporate some global information
  - The *Horn-Schunck method* imposes a smoothness constraint over the whole image
    - Minimize $E = \iint [(I_x u + I_y v + I_t)^2 + \alpha^2(\|\Delta u\|^2 + \|\Delta v\|^2)]\, \mathrm{d}x\, \mathrm{d}y$
      - $\boldsymbol{v} = \begin{bmatrix} u(x, y) \\ v(x, y) \end{bmatrix}$ is the flow vector, as a function of pixel location
      - α is a regularization constant and $\Delta u, \Delta v$ are the flow vector gradients
      - Practically we can take its derivative and convert to a summation, and solve the problem using NLS
        - This would be a very large problem so we'd need to exploit the problem structure as with bundle adjustment
    - This imposes a smoothness constraint, which means adjacent pixels influence each other; large changes in the flow vector at adjacent points is penalized
    - Can help fill in homogeneous (featureless) regions, at the cost of blurring some boundaries

- Optical flow can give us information about the relative depth of objects, since objects further away move
  - This can be exploited for e.g. UAV navigation in urban canyons, where GPS is denied
  - Apparently bees use optical flows for navigation
- A simple navigation rule, if we have 2 cameras pointed towards 2 walls on either side, is to simply steer so that the optical flows on the two sides are equal, since if we're closer to a wall it'll have higher optical flow