Computer Vision & Multimedia Analysis Course

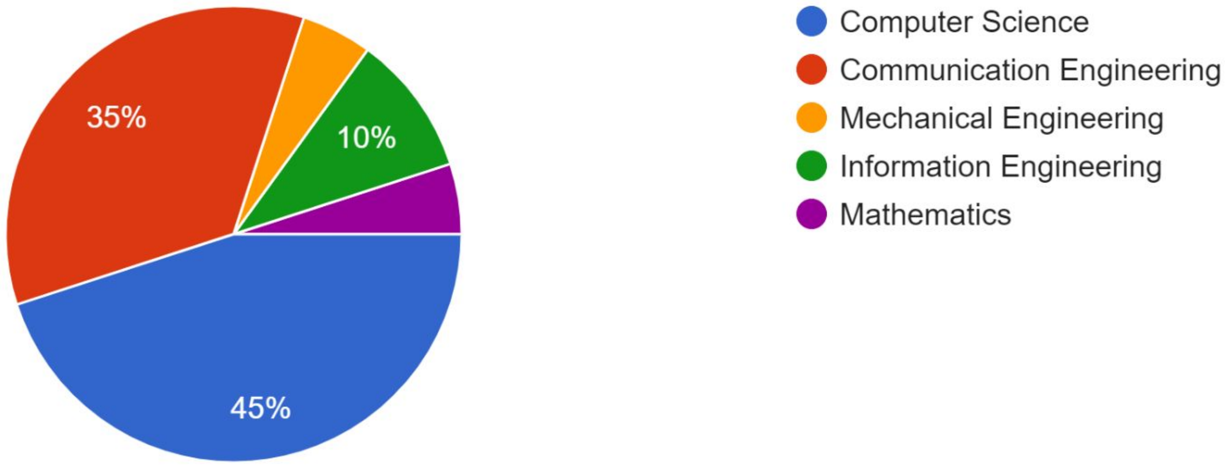
Lab 3: Tracking

Nicola Garau
nicola.garau@unitn.it

Feedback

Which is your background?

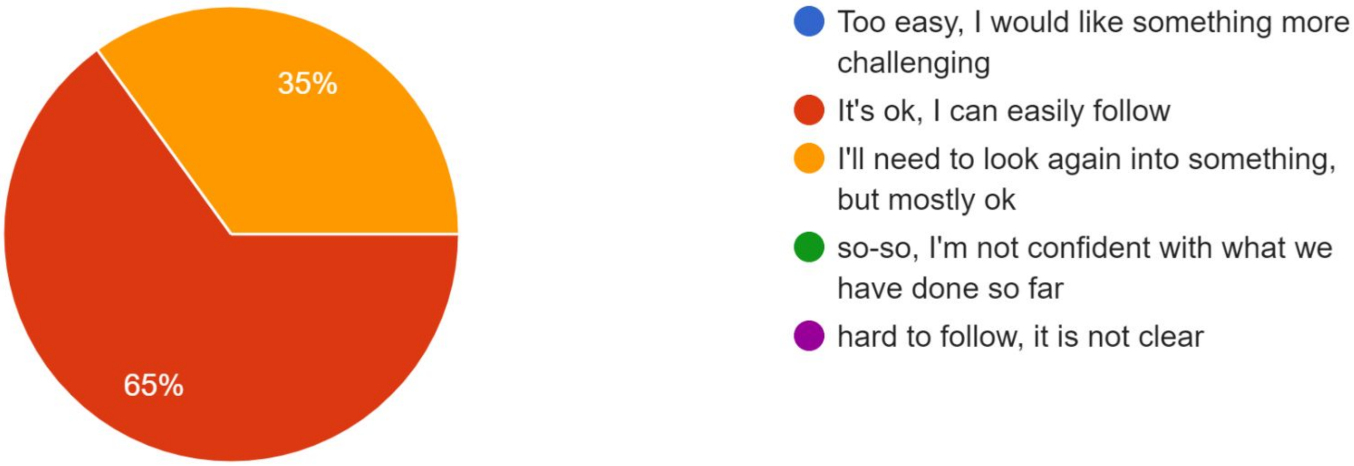
20 risposte



Feedback

The Lab so far:

20 risposte



What you asked for

- More OpenCV *"behind the scenes"* 🚧
- More individual tasks 📃

What's up today (and next time?)

- Good Features to Track + Lucas Kanade optical flow
- Meanshift/Camshift algorithm
- Kalman filter

Good Features to Track

- For each candidate point, compute:

$$Z = \begin{bmatrix} \sum_{W} J_x^2 & \sum_{W} J_x J_y \\ \sum_{W} J_y J_x & \sum_{W} J_y^2 \end{bmatrix}$$

- $\boldsymbol{J_x}$ and $\boldsymbol{J_y}$ are the gradients evaluated on the point in ***x*** and ***y*** direction within ***W*** (***nxn*** window)
- A good feature point is where the smallest **eigenvalue** of ***Z*** is larger than a specified threshold
- *In practice, it highlights corner points and textures*

Lucas-Kanade optical flow estimation

- Two-frame differential method for **optical flow estimation** developed by *Bruce D. Lucas* and *Takeo Kanade* (1981)
- Consider **u=[u$_x$, u$_y$]** in frame **I** and **v=[v$_x$, v$_y$]** in frame **J**
- The goal is to find **d** that satisfies **v=u+d** such as **I and J are similar** (translational model)
- Because of the **aperture problem**, similarity must be defined in 2D
- **d** is the vector that minimizes

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} \left(I(x,y) - J(x+d_x, y+d_y)\right)^2 .$$

- ω is the integration window

GFF+LK tracking

🎯 Detect and select good Features using GFF

↓

🛤️ Track detected feature using LK optical flow

Exercises

Part 1 🛤️

- Track features in the environment using

```
corners, status, err = cv2.calcOpticalFlowPyrLK(prev_frame, frame, prev_corners, None)
```

Part 2 (optional) ✏️

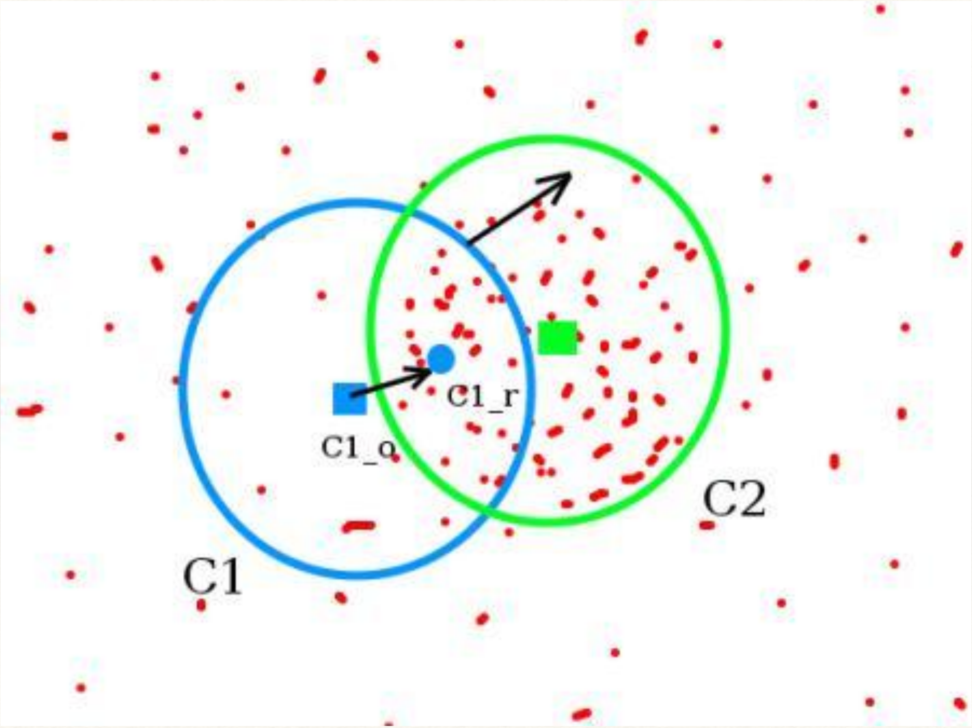
- Draw trajectory of tracked points

Exercises

Part 3 🤔

- How to avoid losing features after some time?
  - Re-detect features using GFF

Meanshift algorithm



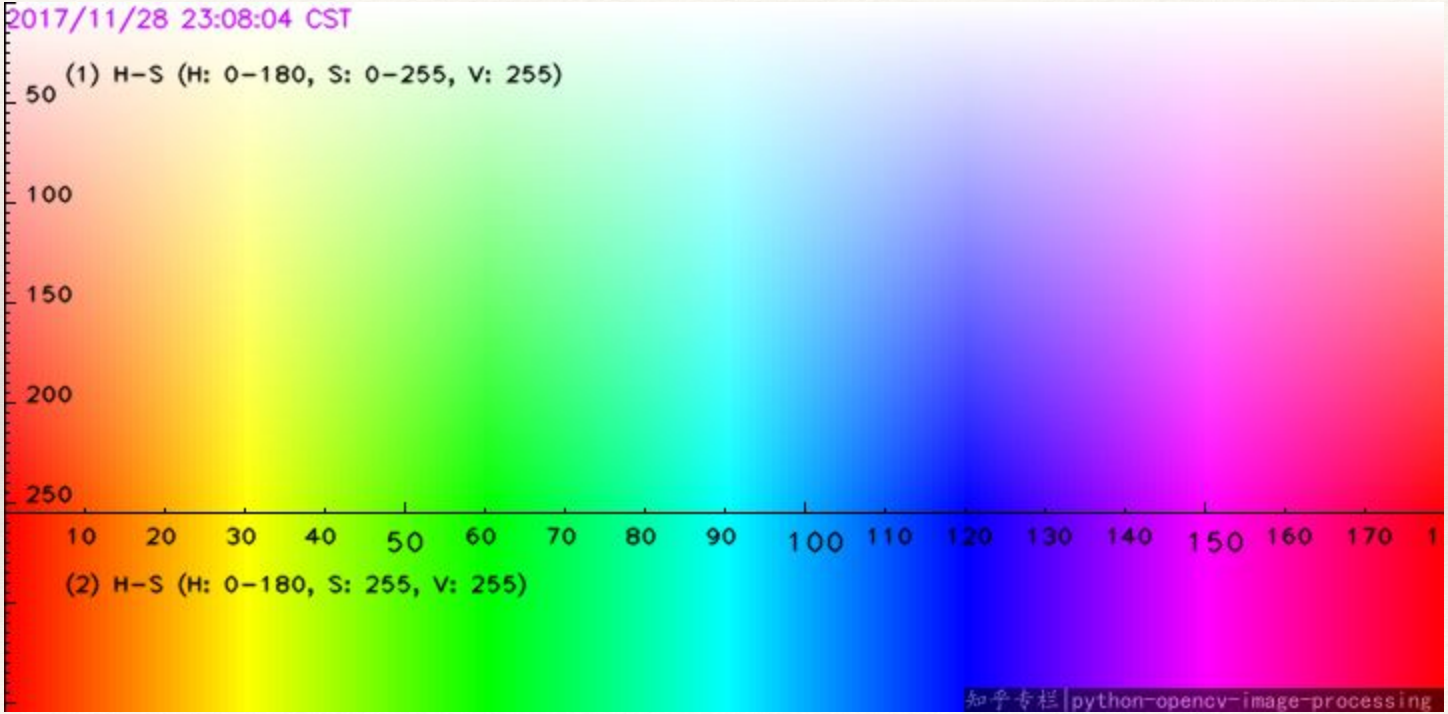
Meanshift algorithm

roi

Target

Meanshift algorithm

Meanshift algorithm



Meanshift algorithm

- RGB to HSV image conversion
- Manually select Region Of Interest (ROI)
- Calculate histogram of ROI
- Back projection of the histogram
- Tracking

Camshift algorithm

- Finds an object center using MeanShift
- Adjusts the window size and finds the optimal rotation.

Exercise

- Implement camShift algorithm instead of MeanShift
- Check documentation on the website
- Display the window using the poly lines function

```
pts = cv2.boxPoints(ret).astype(int)
img2 = cv2.polylines(frame, [pts], True, 255, 2)
```

- Bonus: display backprojection and plot histograms

Kalman filter

$$x_k = A_k x_{k-1} + w_{k-1}$$

$$z_k = H_k x_k + v_k$$

- x_k ➡️ **current state**
- x_{k-1} ➡️ previous state
- A_k ➡️ state transition matrix
- w_k ➡️ process noise
- z_k ➡️ **actual measurement**
- H_k ➡️ measurement matrix
- v_k ➡️ measurement noise

Kalman filter: predict and correct

$$\hat{x}_k^- = A_k \hat{x}_{k-1}$$

$$P_k^- = A_k P_{k-1} A_k^T + Q_{k-1}$$

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H_k \hat{x}_k^-)$$

$$P_k = (I - K_k H_k) P_k^-$$

Kalman filter applied on mouse motion

- Motion equation: $P_t = P_0 + V * t$

$$x_k = A_k x_{k-1} + w_{k-1}$$

$$z_k = H_k x_k + v_k$$

- x_k is the current state ➡️ a vector with the position and velocity
- A_k is the state transition matrix ➡️ matrix that describe the system, in our case the motion equation
- H_k is the measurement matrix ➡️ determined by the current measured position of the mouse
- z_k is the actual measurement ➡️ used to compute the "posteriori"

Transition matrix A_t

$$X_{t+1} = A_t X_t$$

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ vx_{t+1} \\ vy_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ vx_t \\ vy_t \end{bmatrix}$$

Transition matrix

X = [x, y, v_x, v_y]^t

- $x_{t+1} = x_t + v\_x_t$ ➡️ `[1,0,1,0]`
- $y_{t+1} = y_t + v\_y_t$ ➡️ `[0,1,0,1]`
- $v\_x_{t+1} = v\_x_t$ ➡️ `[0,0,1,0]`
- $v\_y_{t+1} = v\_y_t$ ➡️ `[0,0,0,1]`

$$X_{t+1} = A_t X_t$$

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ vx_{t+1} \\ vy_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ vx_t \\ vy_t \end{bmatrix}$$

Exercise

Insert **acceleration** in the transition matrix of the Kalman filter

$$X_t = X_0 + V_x * t$$

$$\Downarrow$$

$$X_t = X_0 + V_x * t + \tfrac{1}{2}\, a_x * t^2$$

Transition matrix

$X = [x, y, v\_x, v\_y, a\_x, a\_y]^t$

- $x_{t+1} = x_t + v\_x_t + 0.5\, a\_x_t$
- $y_{t+1} = y_t + v\_y_t + 0.5\, a\_y_t$
- $v\_x_{t+1} = v\_x_t + a\_x_t$
- $v\_y_{t+1} = v\_y_t + a\_y_t$
- $a\_x_{t+1} = a\_x_t$
- $a\_y_{t+1} = a\_y_t$

$$X_{t+1} = A_t X_t$$

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ vx_{t+1} \\ vy_{t+1} \\ ax_{t+1} \\ ay_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ vx_t \\ vy_t \\ ax_t \\ ay_t \end{bmatrix}$$