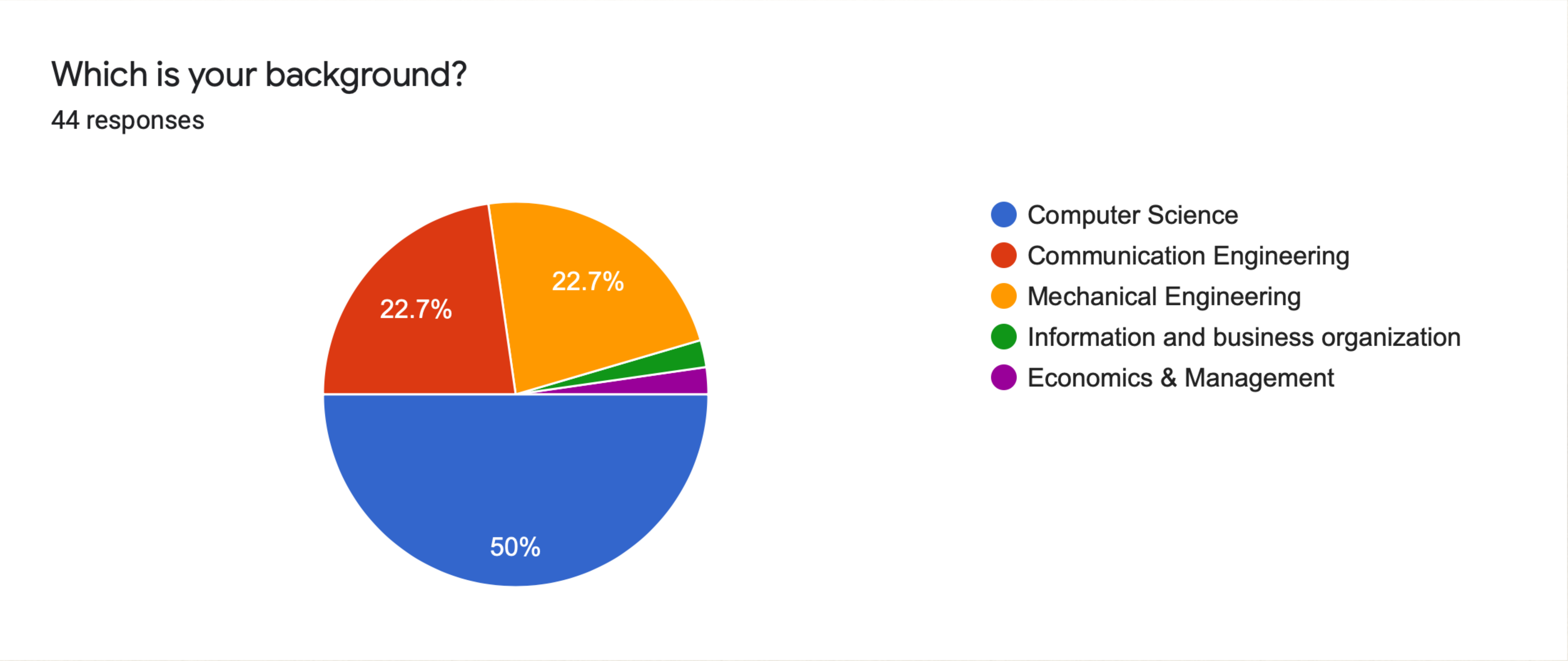
Computer Vision & Multimedia Analysis Course

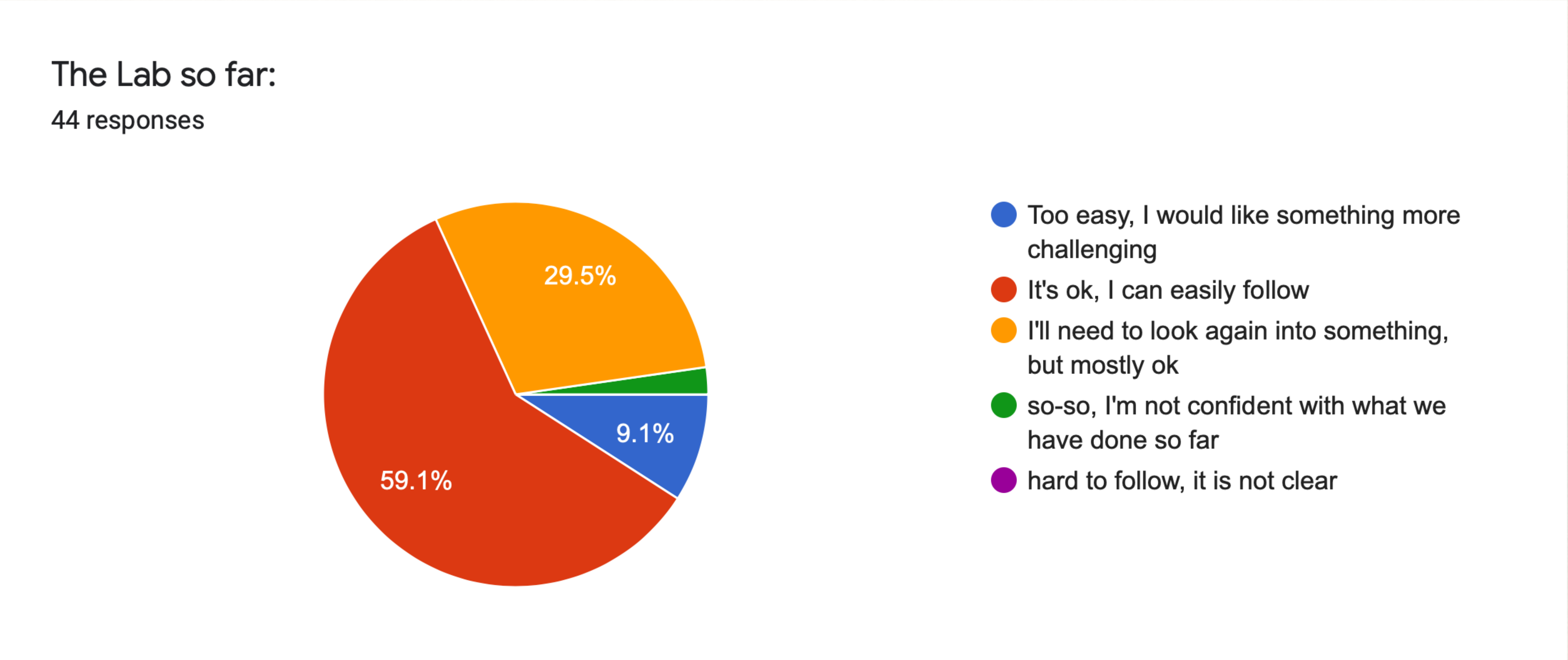
Lab 3: Tracking

Niccolò Bisagno
niccolo.bisagno@unitn.it

Feedback


Which is your background?
44 responses
22.7%
22.7%
50%
Computer Science
Communication Engineering
Mechanical Engineering
Information and business organization
Economics & Management

Feedback


The Lab so far:
44 responses
29.5%
9.1%
59.1%
Too easy, I would like something more challenging
It's ok, I can easily follow
I'll need to look again into something, but mostly ok
so-so, I'm not confident with what we have done so far
hard to follow, it is not clear

Feedback

- Material before lectures
- Real-world applications examples
- More challenging suggestions/material

What's up today (and Monday)

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

Good Features to Track

- For each candidate point, compute:

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

- J_x and 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 $\mathrm{u}=[\mathrm{u}_x, \mathrm{u}_y]$ in frame I and $\mathrm{v}=[\mathrm{v}_x, \mathrm{v}_y]$ in frame J
- The goal is to find **d** that satisfies $\mathbf{v=u+d}$ 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

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

- ω is the integration window

GFF+LK tracking

Use GFF to detect and select good Features

↓

Track detected feature using LK optical flow

Exercise

Part 1

- Track features in the environment using
- corners, status, err = cv2.calcOpticalFlowPyrLK(prev_frame, frame, prev_corners, None)

Part 2 (optional)

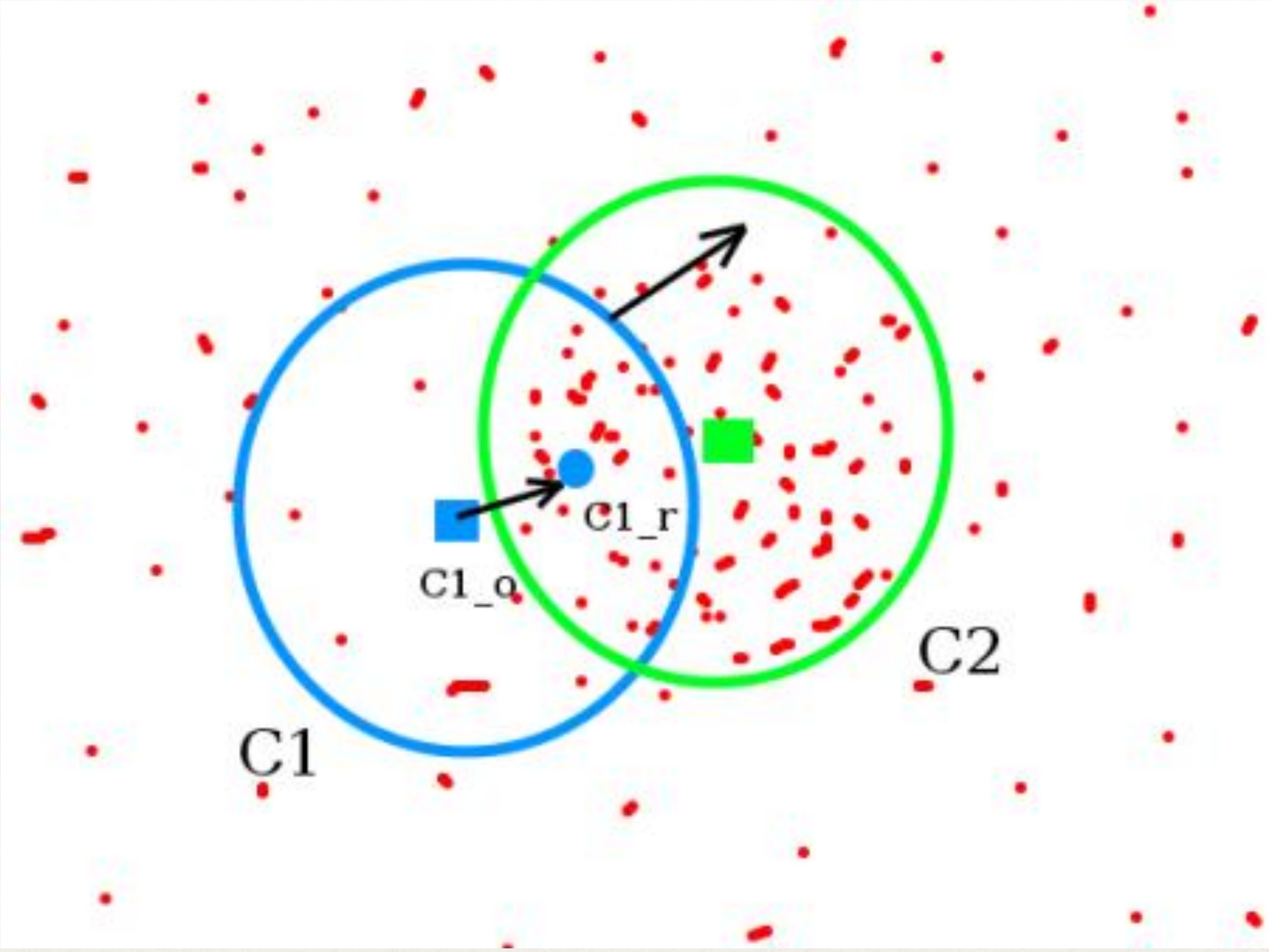
- Draw trajectory of tracked points

Exercise

Part 3

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

Meanshift algorithm



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)
pts = np.int0(pts)
img2 = cv2.polylines(frame,[pts],True, 255,2)
```

- Bonus: display backprojection and plot histograms

Kalman filter

- Inside the Virtual Machine (or in your programming environment)
- Go to this link and download the file
- https://github.com/mmlab-cv/CVLaboratories/Lab3/kalma_start.py

Kalman filter

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

$$z_k = H_k x_k + v_k$$

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

Kalman filter

$$\hat{x}_k^- = A_k \hat{x}_{k-1}$$
$$P_k^- = A_k P_{k-1} A_k^T + Q_{k-1}$$

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$$
$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - H_k \hat{x}_k^-)$$
$$P_k = (I - K_k H_k) P_k^-$$

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

- $X = [x, \quad y, \quad v\_x, \quad v\_y]^t$
- $x_{t+1} = x_t + v\_x_t$ ——-> $[1, \quad 0, \quad 1, \quad 0 \,]$
- $y_{t+1} = y_t + v\_y_t$
- $v\_x_{t+1} = v\_x_t$ ——-> $[0, \quad 0, \quad 1, \quad 0 \,]$
- $v\_y_{t+1} = v\_^t{}_t$

Exercise

- Insert acceleration in the transition matrix of the Kalman filter

$$x_t = x_0 + v_x * t$$

$$\Downarrow$$

$$x_t = x_0 + v_x * t + \frac{1}{2} a_x * t^2$$

Transition matrix

- $X = [x, y, v\_x, v\_y, a\_x, a\_y]^t$
- $x_{t+1} = x_t + v\_x_t + 0.5\, a\_x_t$
- $y_{t+1} = y_t + v\_y_t + 0.5\, a\_y_t$
- $v\_x_{t+1} = v\_x_t + a\_x_t$
- $v\_y_{t+1} = v\_t_t + a\_t_t$
- $a\_x_{t+1} = a\_x_t$
- $a\_y_{t+1} = a\_y_t$