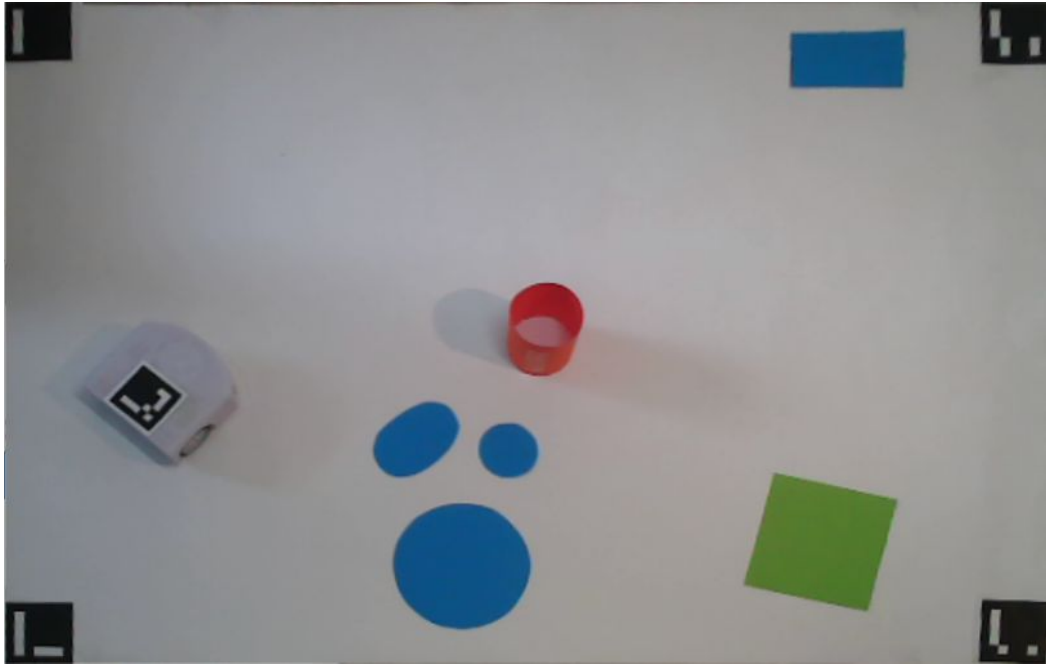
Basics of mobile robotics

Thymio project

Group 47 : Martin Daigne, Laetitia Fayad,
Arthur Lamour, Amandine Meunier

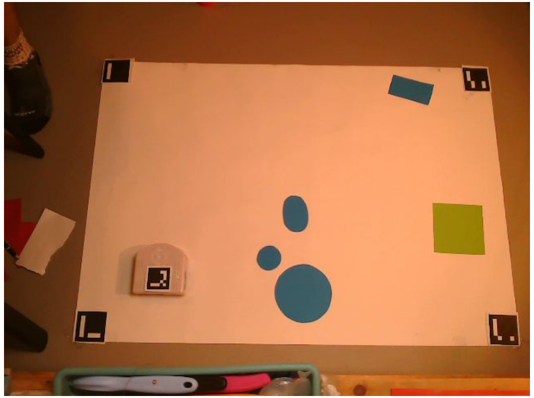
Our project: Arena



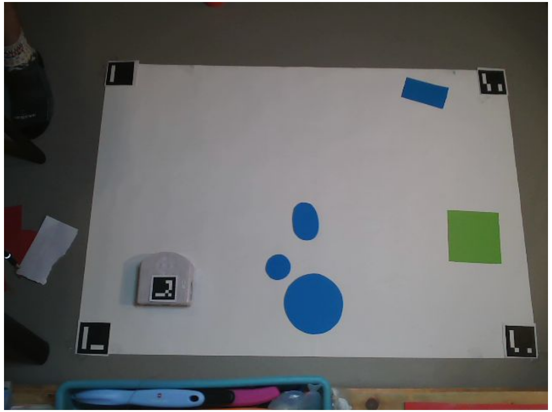
- A0 White sheet
- Goal : shape of different color than global obstacles
- Global obstacles: any colors, any shapes
- Local obstacles: small shapes, any color
- 5 ArUco markers : corners and thymio

Computer Vision – Initialization

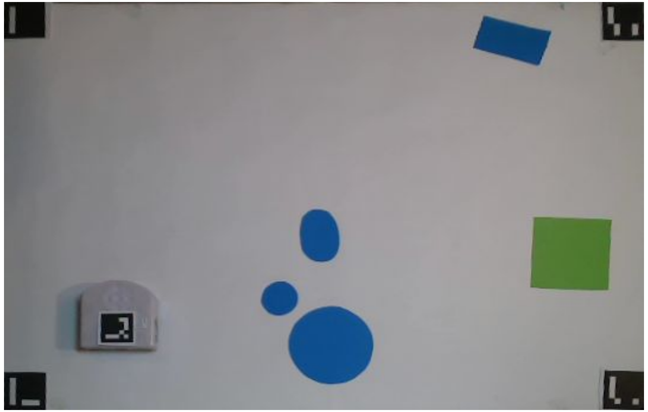
1) First image



2) After Warm Up

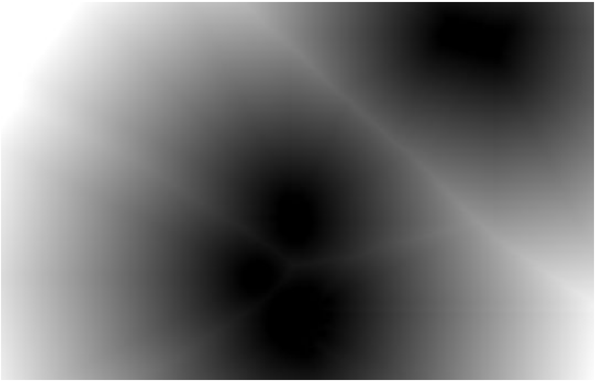


3) After perspective correction



Goal Detection and Obstacles Expansion

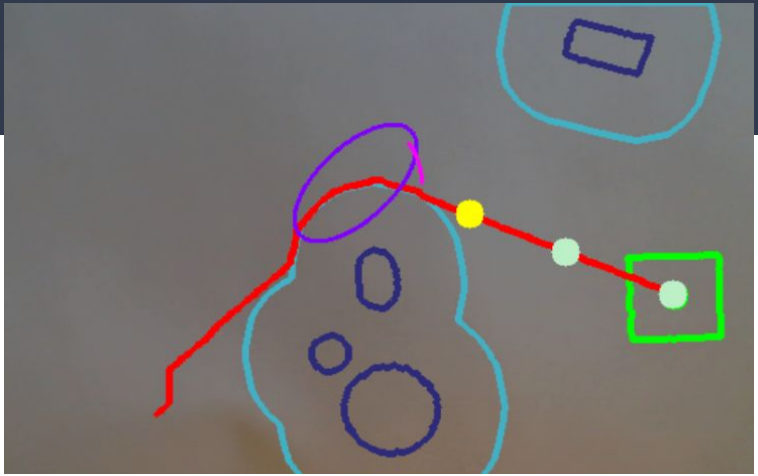
- Global obstacles and goal found with color thresholds
- Small blobs removed
- Contours extracted and expanded by the thymio radius for the obstacles



Distance map to obstacles for expansion

Final image for plotting and A*

Kalman Filter



Motion Equations: (Runge Kutta 2 for better precision)

$$\begin{pmatrix} x(t+\Delta t) = x(t) + v \cdot \cos(\theta + \frac{\omega \cdot \Delta t}{2})\Delta t \\ y(t+\Delta t) = y(t) + v \cdot \sin(\theta + \frac{\omega \cdot \Delta t}{2})\Delta t \\ \theta(t+\Delta t) = \theta(t) + \omega \cdot \Delta t \end{pmatrix}$$

$$\omega = \frac{v_L - v_R}{L} \qquad v = \frac{v_L + v_R}{2}$$

1. Prediction: $x_t = g(u_t, x_{t-1})$ and $\Sigma_t = G_t \Sigma_{t-1} G_t^\top + Q_t$
2. Update: $K_t = \Sigma_t H_t^\top \left(H_t \Sigma_t H_t^\top + R_t\right)^{-1}$, $x_t = x_t + K_t\left(z_t - h(x_t)\right)$ and $\Sigma_t = (I - K_t H_t)\Sigma_t$

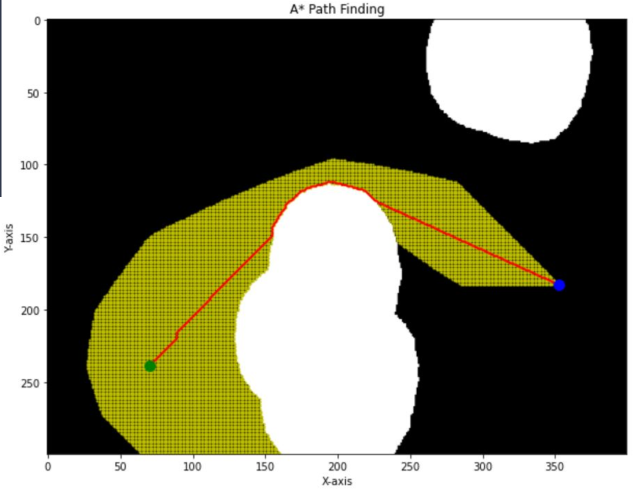
With: $Q = Gu, t \cdot Q_{control} \cdot Gu, t^\top$

Q= diag(var(left motor speed),var(right motor speed))

R=diag(var(x_cam),var(y_cam),var(theta_cam))

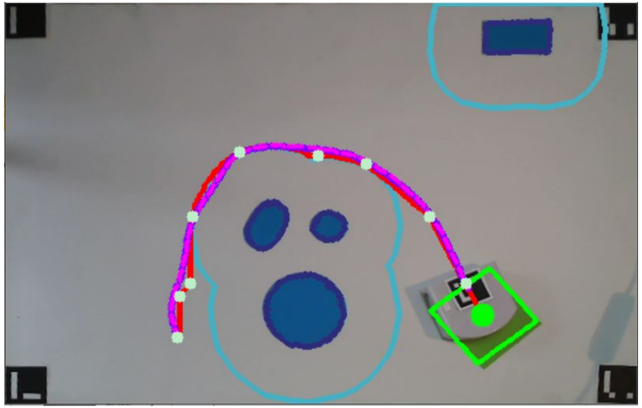
Both covariance measured, not even tuned!

Path Planning



A* path search:

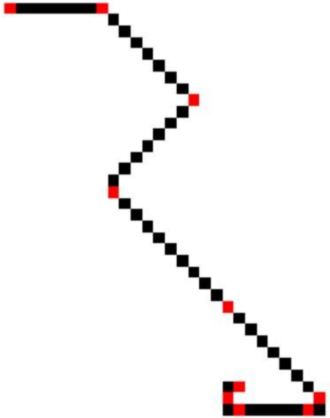
- Complete, will always give a solution
- image is a grid so straightforward implementation
- processing time/precision trade off easily tuned with grid size
- 8 neighbors, L2 norms
- Problem: not optimal because robot is not constraint to 45 degree angles



Key points:

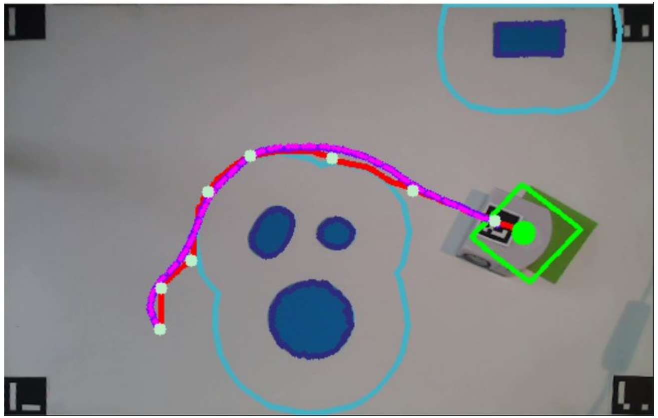
+ Easier movement
- Shortcuts paths

- Rotation
- Max interval

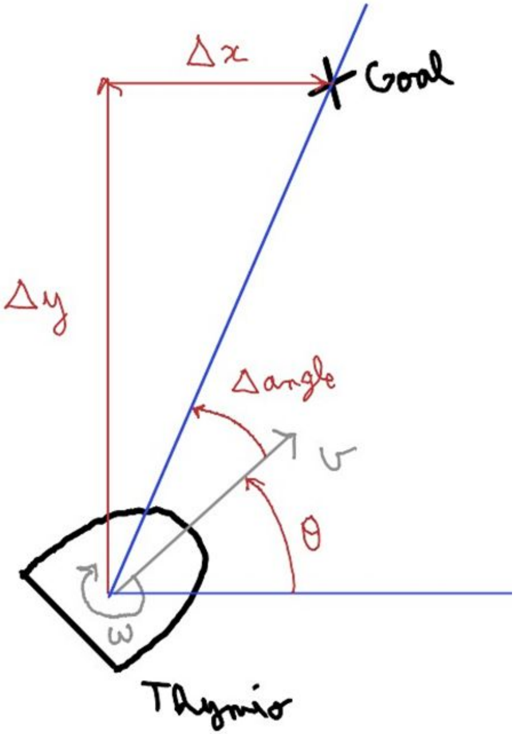


Motion control : Following the path

- Controller : proportional to Δangle



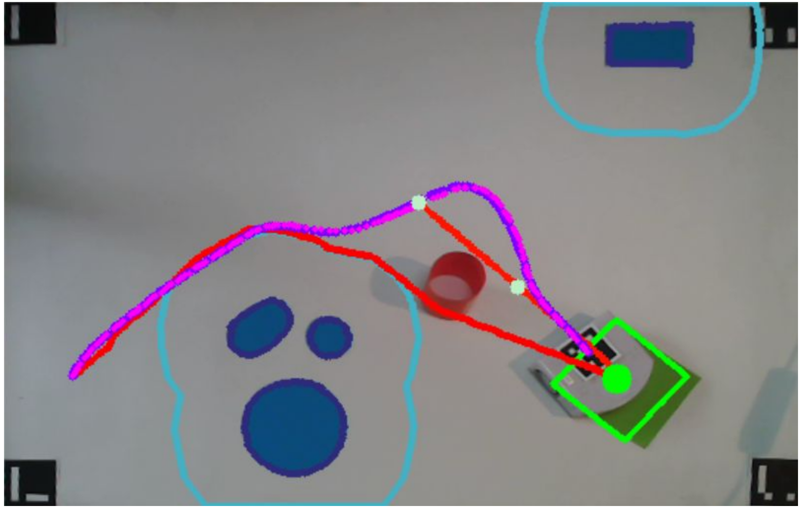
```
v = SPEED                           # Translational velocity PWM
omega = k_alpha*(delta_angle) # Rotational velocity [rad/s]

# Calculate motor speed
v_ml = (v+omega*L_AXIS) #PWM
v_mr = (v-omega*L_AXIS) #PWM
```



Motion control : Local avoidance

- Braitenberg logic : speed corrections are added proportionally to proximity sensors values

```
def avoid_obstacle(prox_values): # Prox values go from left to right
    braitenberg = [-2/300, -10/300, 25/300, 11/300, 3/300] # Tuned parameters
    v_mr, v_ml = SPEED, SPEED
    for i in range (len(prox_values)) :
        v_ml -= braitenberg[i] * prox_values[i]
        v_mr += braitenberg[i] * prox_values[i]
    return v_ml, v_mr
```

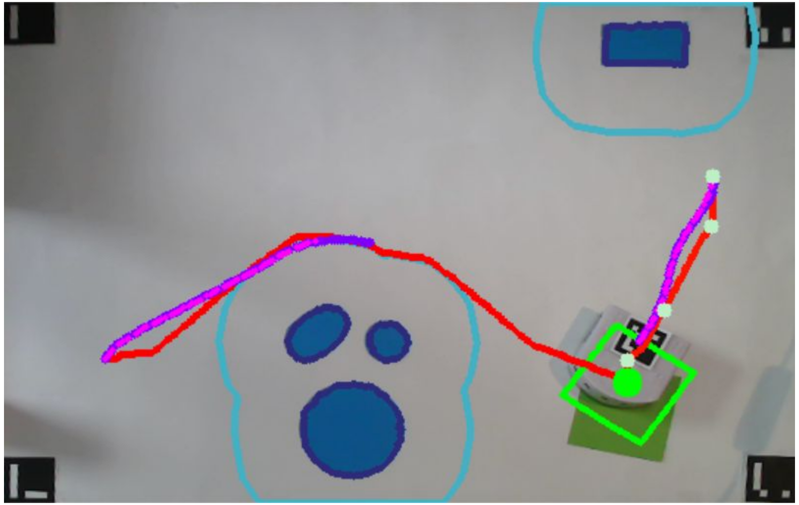


Kidnapping : using proximity ground sensors

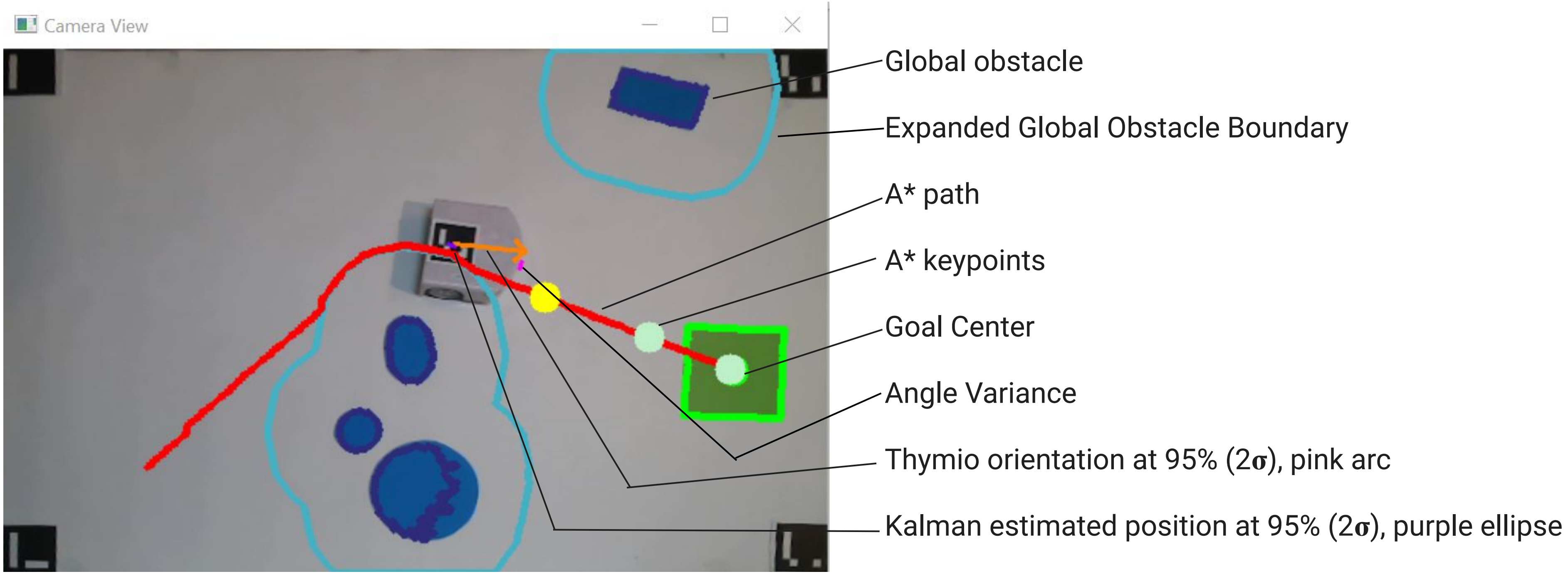
- If sensors value under a certain threshold -> not on the white ground anymore

  -> we stop the motors and stop the localisation

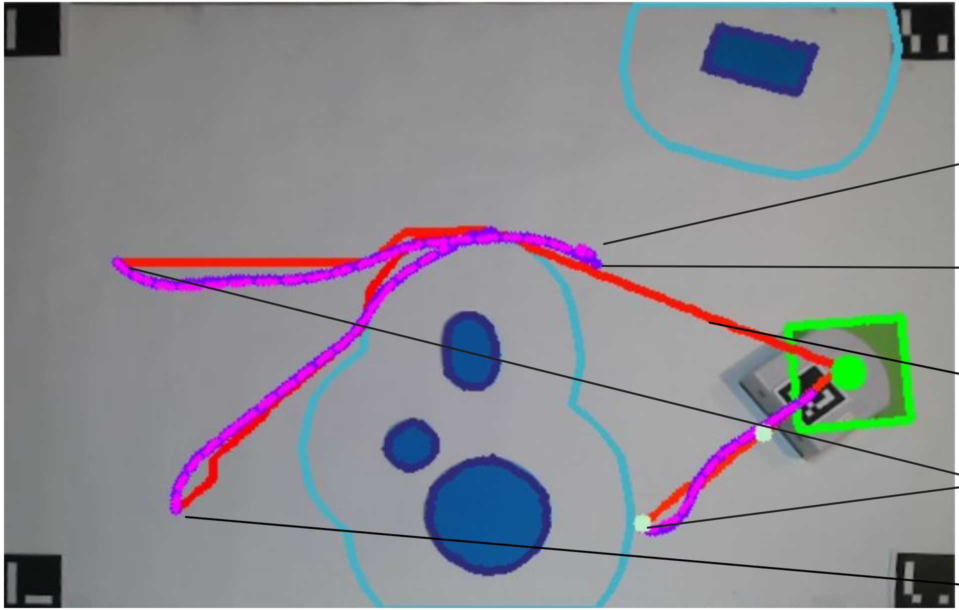
  Once back on the ground : update position and find new path



Path Tracking



Path History (with kidnapping)



Camera measured position, pink

Kalman estimated position, purple

A* paths, red

Kidnappings

Start position

Computer Vision – ArUco Markers

- The 4 corners of the arena are 4 Aruco markers with different ids. And there is one on the Thymio
- Aruco because robust to varying lighting conditions and precise
- With the corner’s markers size in pixel and the real size in mm we get the factor pix/mm
- Thymio angle is found with the top and bottom edge