Balloon Popping Robair

Group 4

Juan Vazquez, Charles Marshall, Sannara Ek, Shakeel Ahmad Sheikh

Contact: charles.marshall@grenoble-inp.org

Our Project

Robair:

- Intelligently identifies balloon targets while avoiding other targets, including human legs.
- Moves smoothly to the balloons
- Pops all balloons in the arena

*We use an upside down laser for improved detection of balloons.

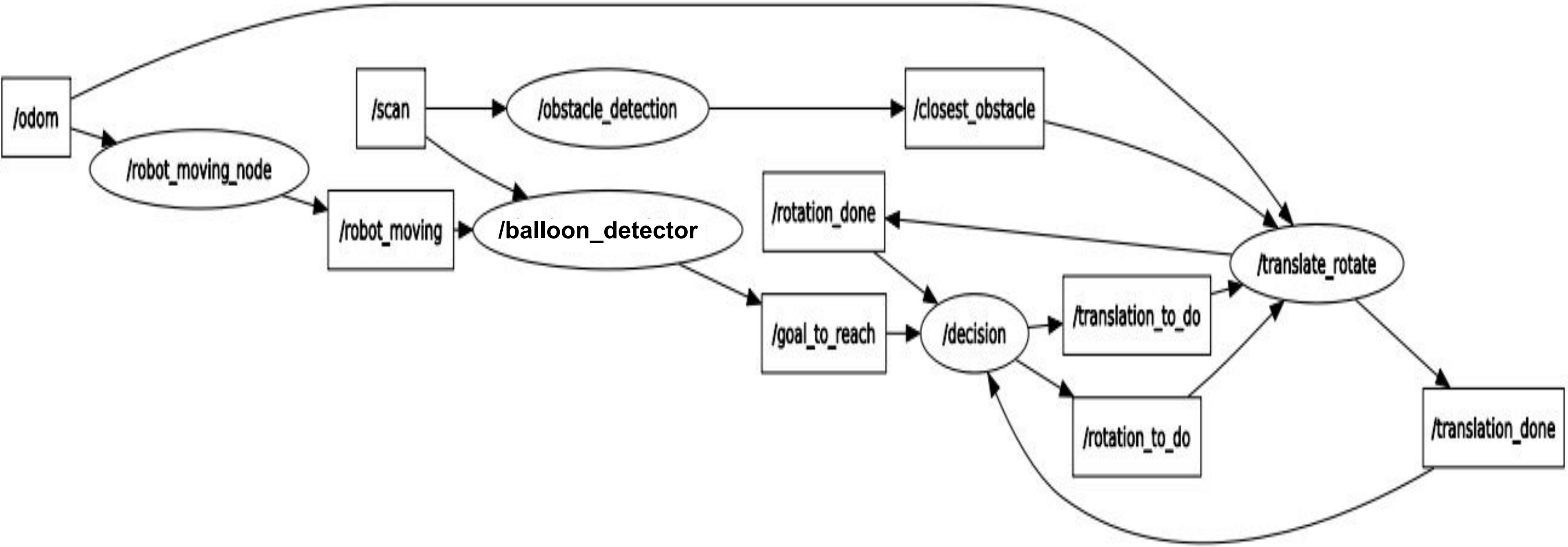
Developments

- **Decision** → Decision_Node.cpp
- **Vision** → Balloon_detector_node.cpp
- **Rotation** / **Translation** → Translate_Rotate_Node.cpp

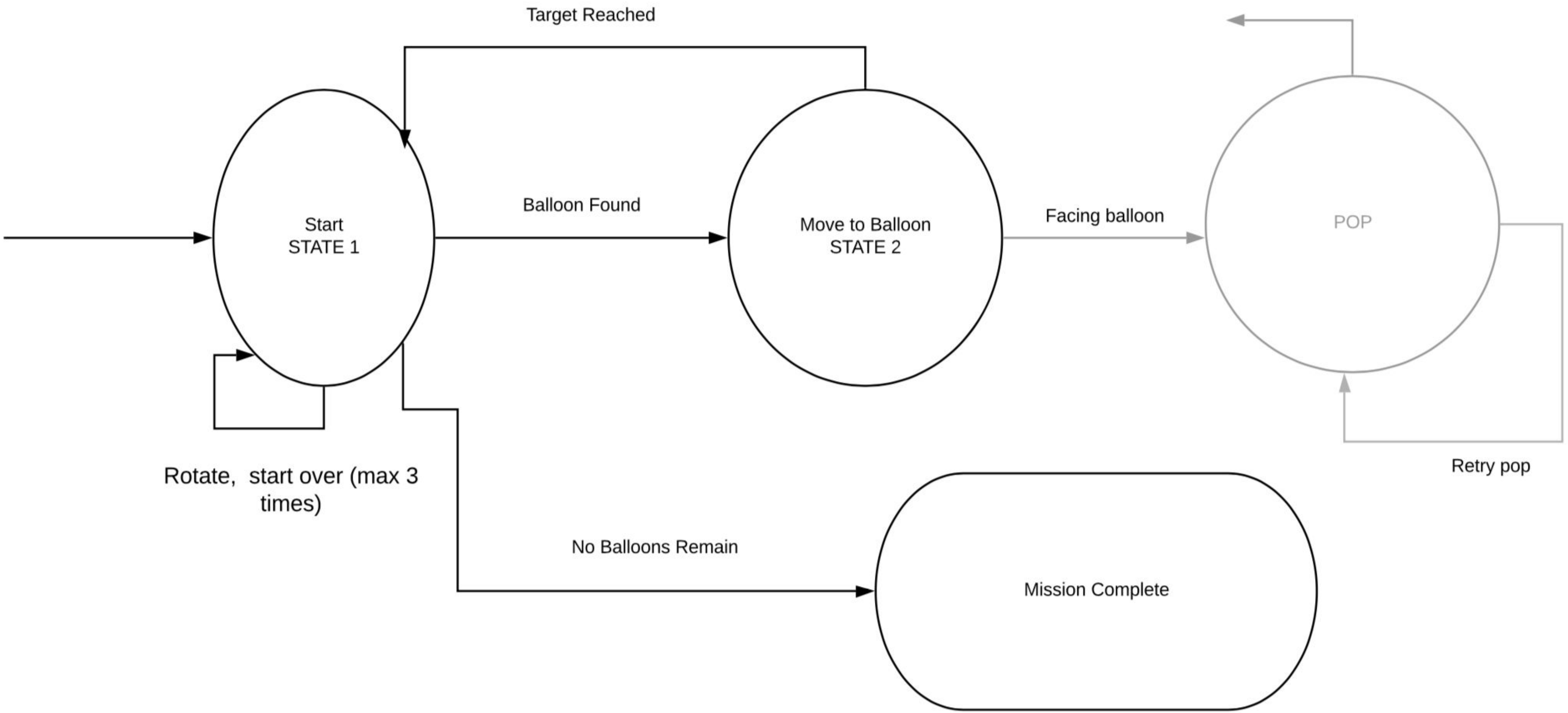
Additionally:

- Balloon Tracking → Balloon_Tracker_Node.cpp

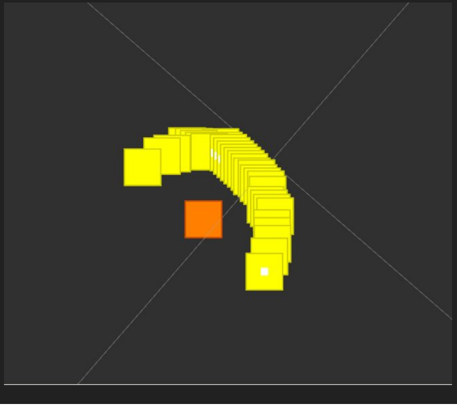
Architecture



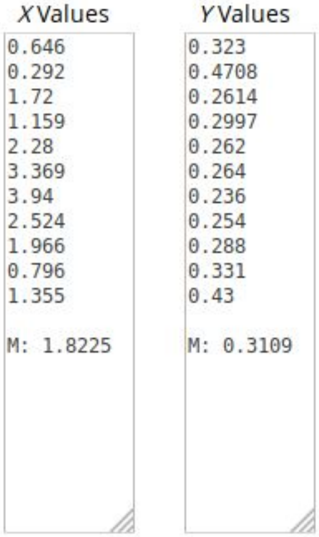
Decision


Target Reached
Start
STATE 1
Balloon Found
Move to Balloon
STATE 2
Facing balloon
POP
Rotate, start over (max 3 times)
Retry pop
No Balloons Remain
Mission Complete

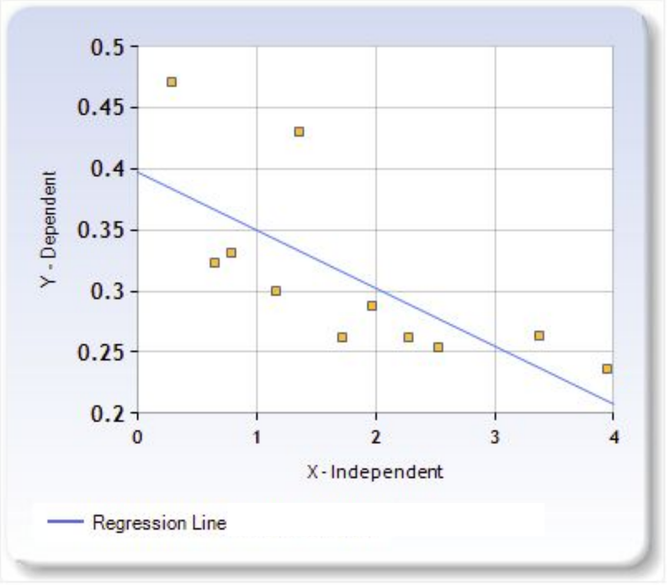
Detecting the size of a balloon



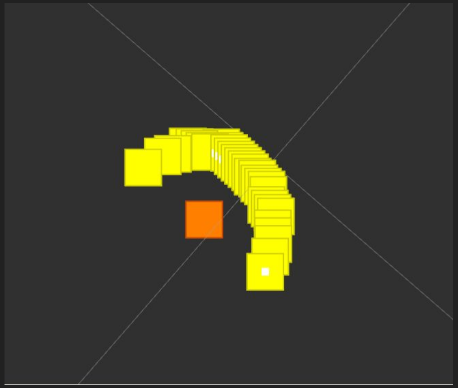
- A Particular Width
- Collected Data
- A Linear Regression Problem
  - BS = 0.05 * Distance - 0.28
- Outliers to adjust



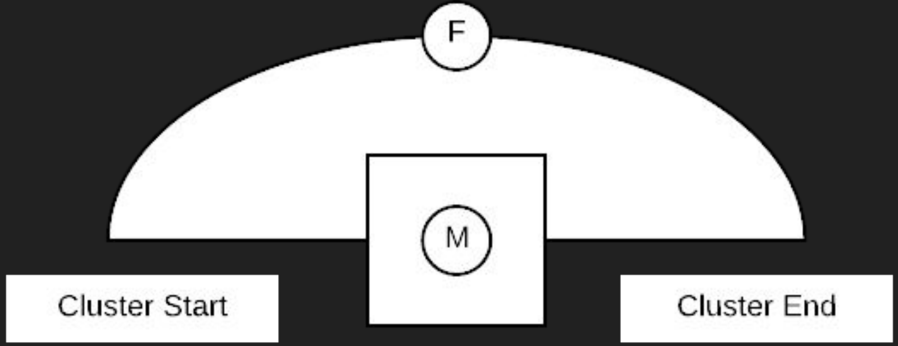
X Values	*Y* Values
0.646	0.323
0.292	0.4708
1.72	0.2614
1.159	0.2997
2.28	0.262
3.369	0.264
3.94	0.236
2.524	0.254
1.966	0.288
0.796	0.331
1.355	0.43
M: 1.8225	M: 0.3109



Detecting the curve of a balloon



- Calculate the middle(M) and the front point (F)
- Determine a threshold distance
- Calculate the distance between M and F
- Check if F is within the threshold

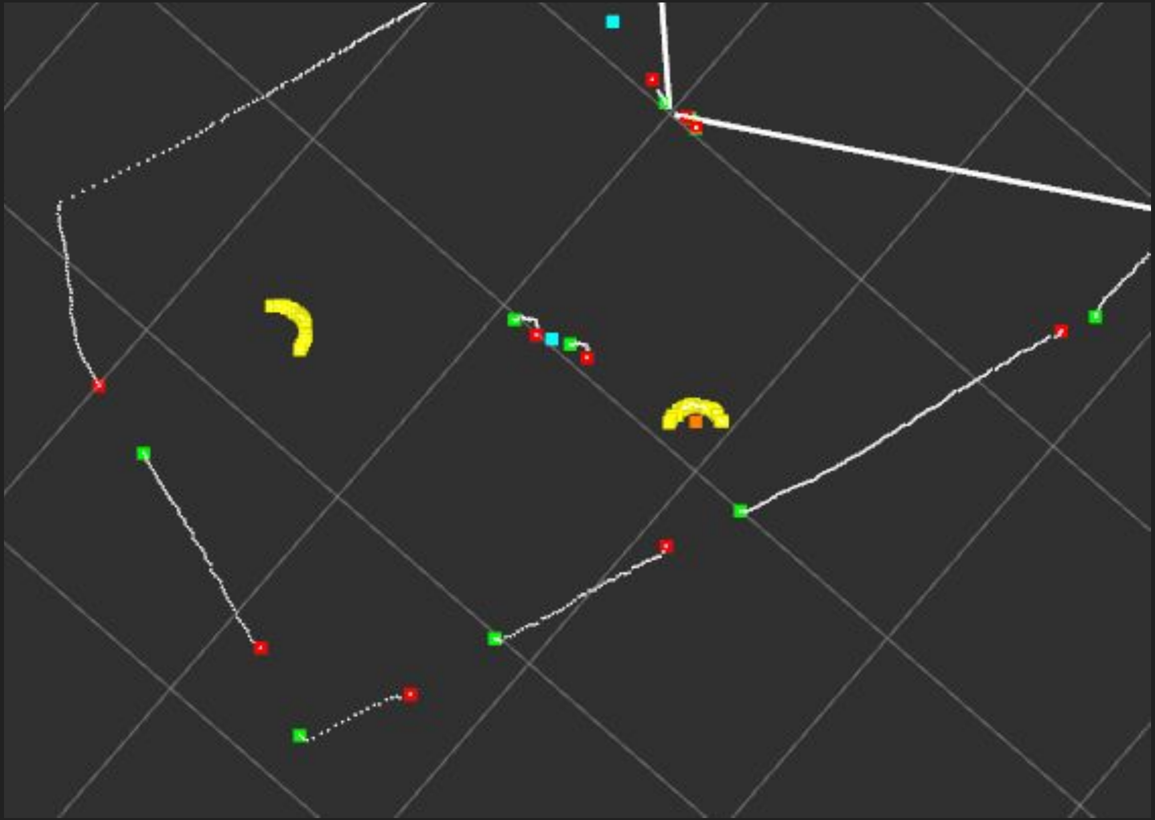


Avoiding Legs

- **Identify Human targets**
- **Filters out noises (Legs)**
- **Threshold for uncertainty**

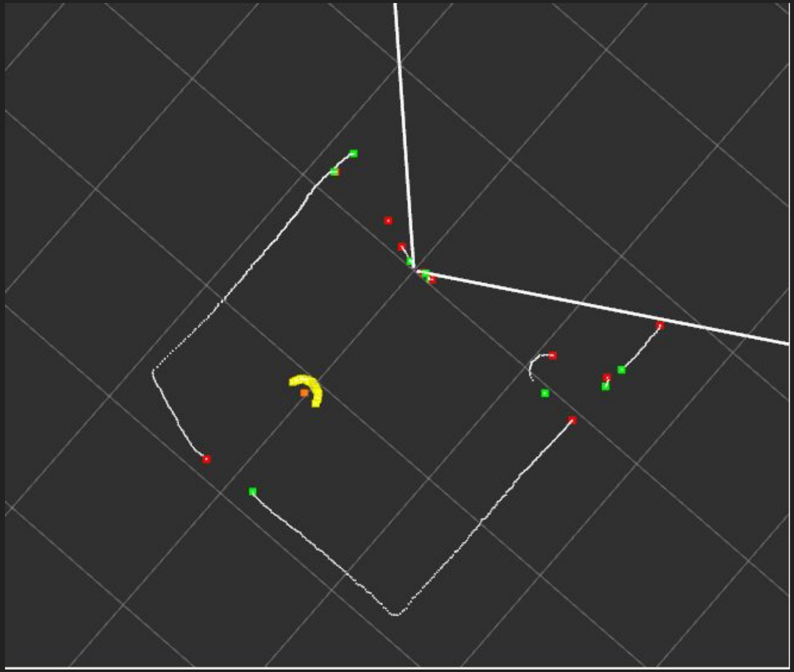
Balloon Selections

- Closest Balloon



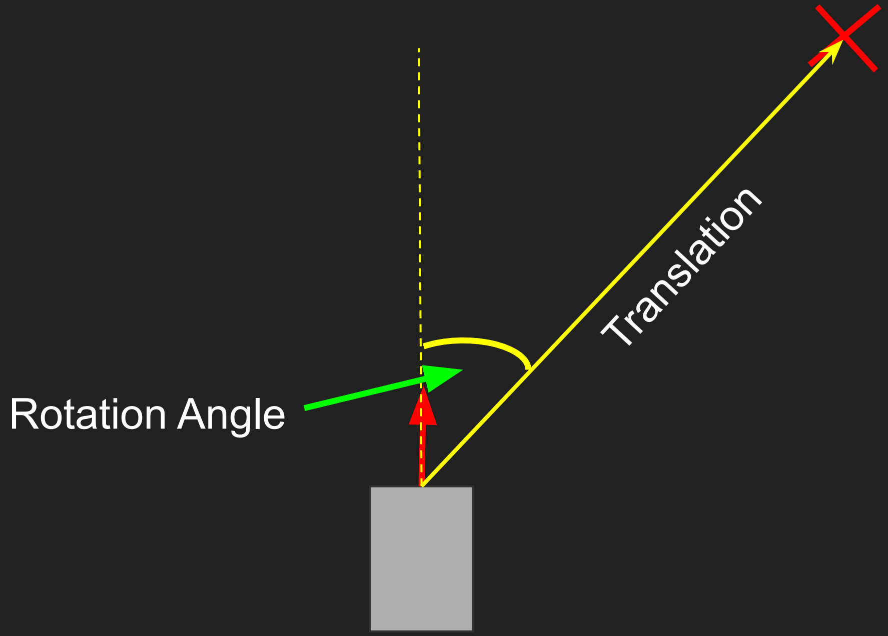
Constraints with robair and balloons

- Using a upside down laser
- (some of the field of view is lost)
- Balloon Color Problem
- Possibility of using two lasers



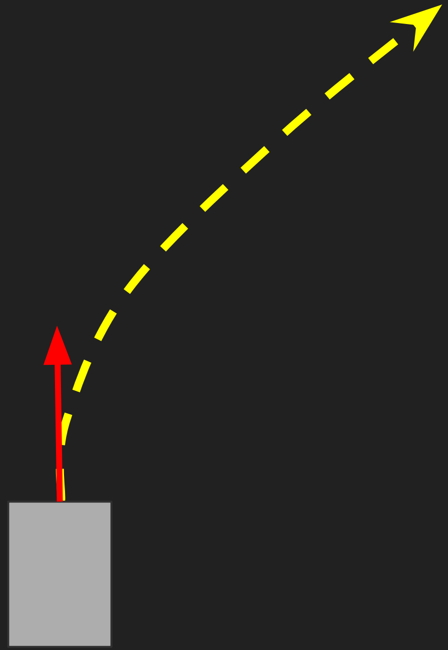
Rotation / Translation

- Decision node:
  - Send rotation_to_do
  - Send translation_to_do
- Rotation/Translation:
  - Starts when both values have been received



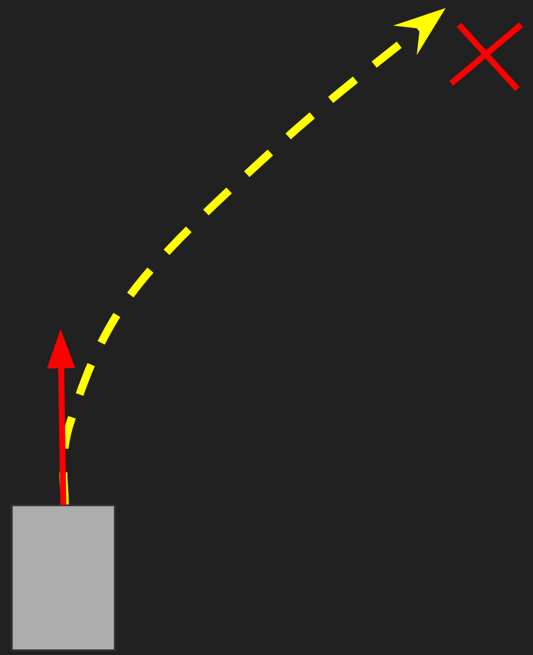
Translate_rotate node

- Waits for both rotation_to_do and translation_to_do
- Using PID, calculates rotation_speed and translation_speed
  - $W = 1 - \frac{rotation\_speed}{THETA\_MAX}$
  - Final translation speed
    - Translation_speed * w

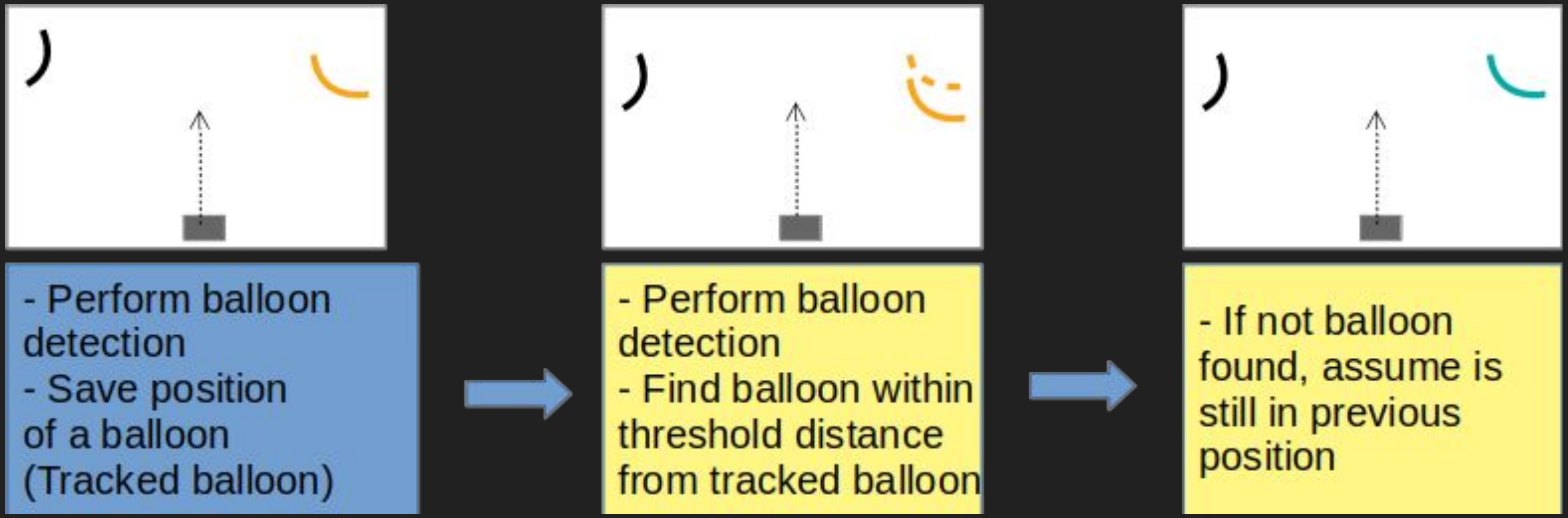


Translate and rotate: getting to objective

- We may miss the objective
    - Rely on the decision node logic
        - If objective is visible, go for it.
        - If not, rotate 90° until an objective is found



Extra feature: Balloon tracking*



- Perform balloon detection
- Save position of a balloon (Tracked balloon)

- Perform balloon detection
- Find balloon within threshold distance from tracked balloon

- If not balloon found, assume is still in previous position

*Not fully integrated in the system

Final Remarks

- Our work
- Our limitations
- Our ideas moving forward

DEMO

(3 Parts)