# Reactive Scheduling of Computational Resources in Control Systems

Hodai Goldman

Ben-Gurion University of the Negev
Department of Computer Science

January 1, 2018

# Overview - *TODO: clean this slide*

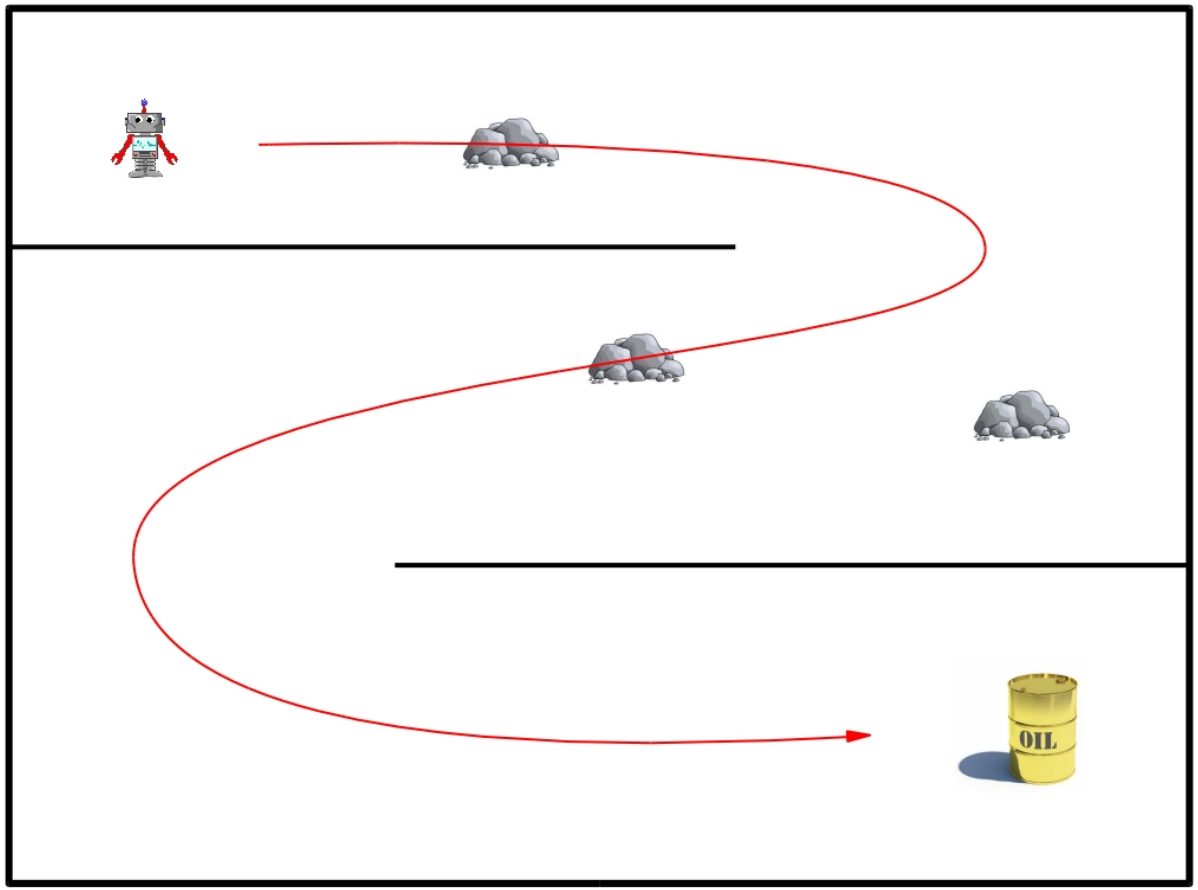
## Contributions

- Development of control and scheduling co-design framework
- **Reactive** scheduling (environment condition adaptation)
- **Independent**, **adaptive**, and **composable** interface (Based on automata theory)
- *What we do better?*
- Prepare the ground for automata-based **scheduling tool**
- Development of scheduling technique based on **Kalman filter**

## Achievements of this thesis

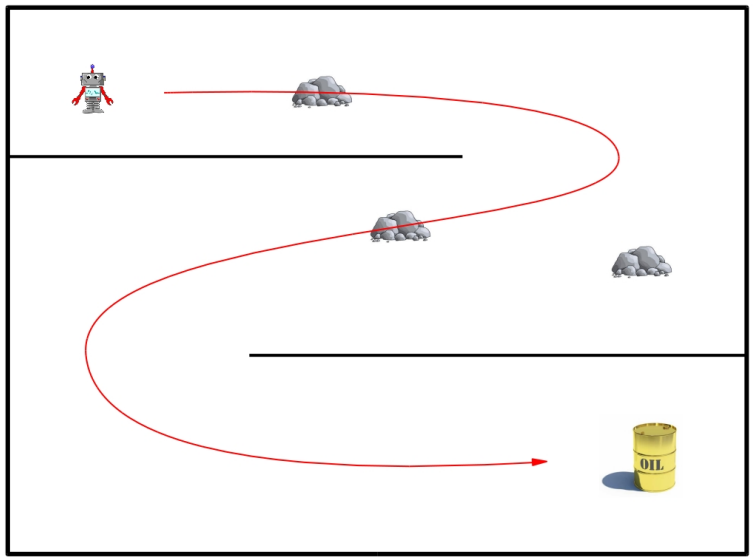
- Continue the work of **RTComposer**
- Proof of concept with **simulation**
- Proof of concept with **real-life case-study**
- **Bridge** the gaps between control and software engineering

# An control problem example




Robot navigation

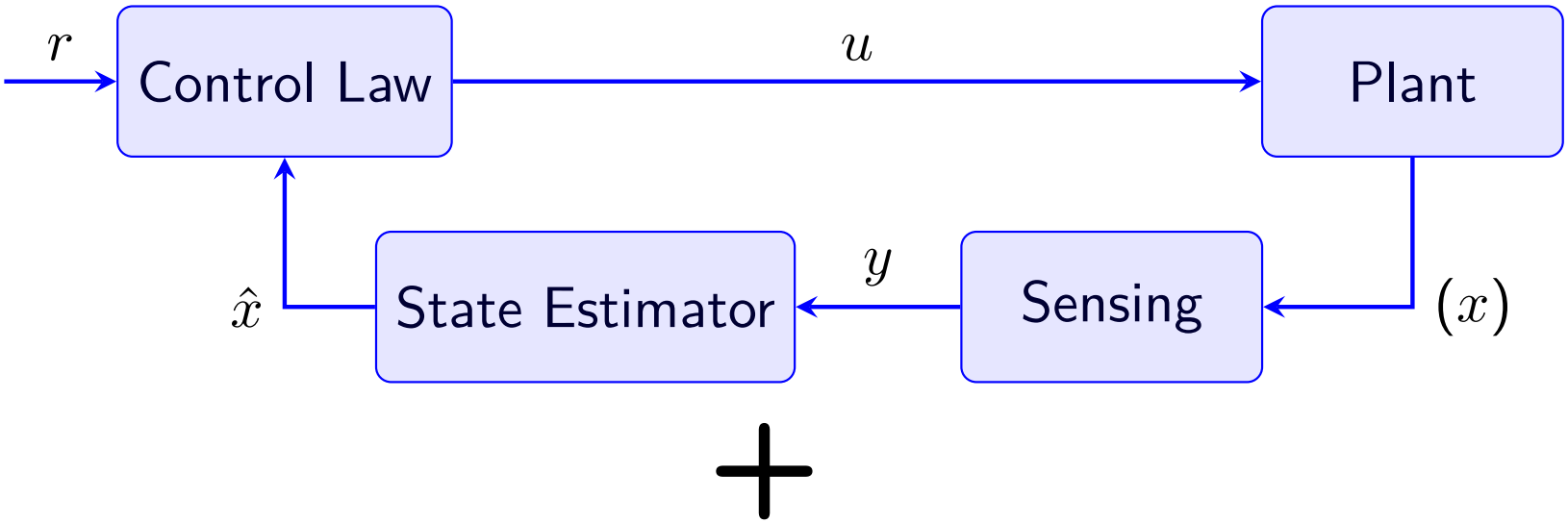
# An control problem example




## The Objectives

- The robot need to reach the target point **fast** and **safely**
- The robot have on-board camera for **obstacle-avoidance**
- The robot use GPS for general **navigating**

# The Traditional Solution




## Constant time steps + periodic tasks

*time steps*

*figure*+

| Task | Period | Deadline |
|---|---|---|
| Check for obstacles | 10ms | 1.5ms |
| Check GPS position | 10ms | 0.5ms |
| Control Law | 2ms | 0ms |
| ... | | |

# The Main Software Design Problems

| Task | Period | Deadline |
|---|---|---|
| Check for obstacles | 10ms | 1.5ms |
| Check GPS position | 10ms | 0.5ms |
| Control Law | 2ms | 0ms |
| ... | | |

## The design problems from our point of view

- **All the tasks are highly coupled**: *any change or addition of some task require to consider all other tasks requirements*
- **Static and inefficient scheduling**: *the table is defined for the worst case talk about related work on this direction*
- **No consideration of the environmental conditions**: it is a cyber-physical system after all

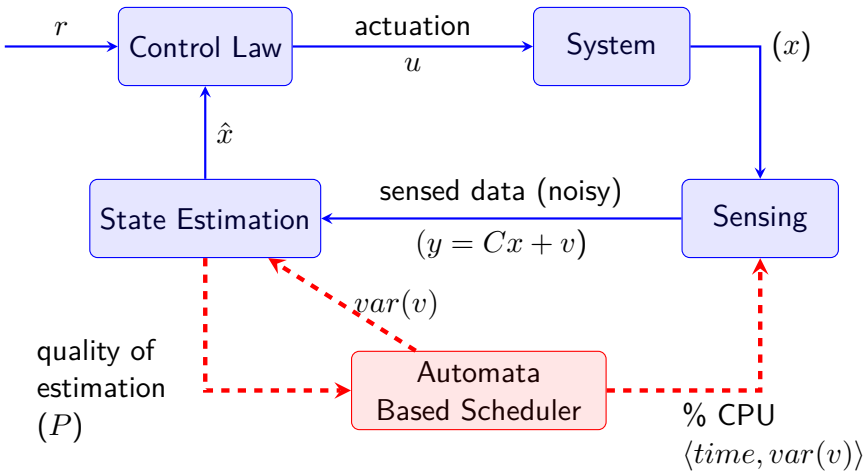
# The Goal

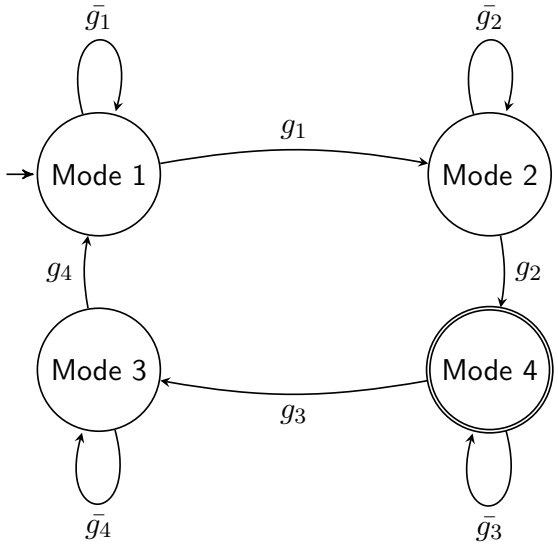
In this thesis we design an **reactive** scheduling framework for real-time systems

## Required features:

- **Independent** and **composable** requirements
- **Control objective based** requirement interface
- Environment **adoptive** scheduler

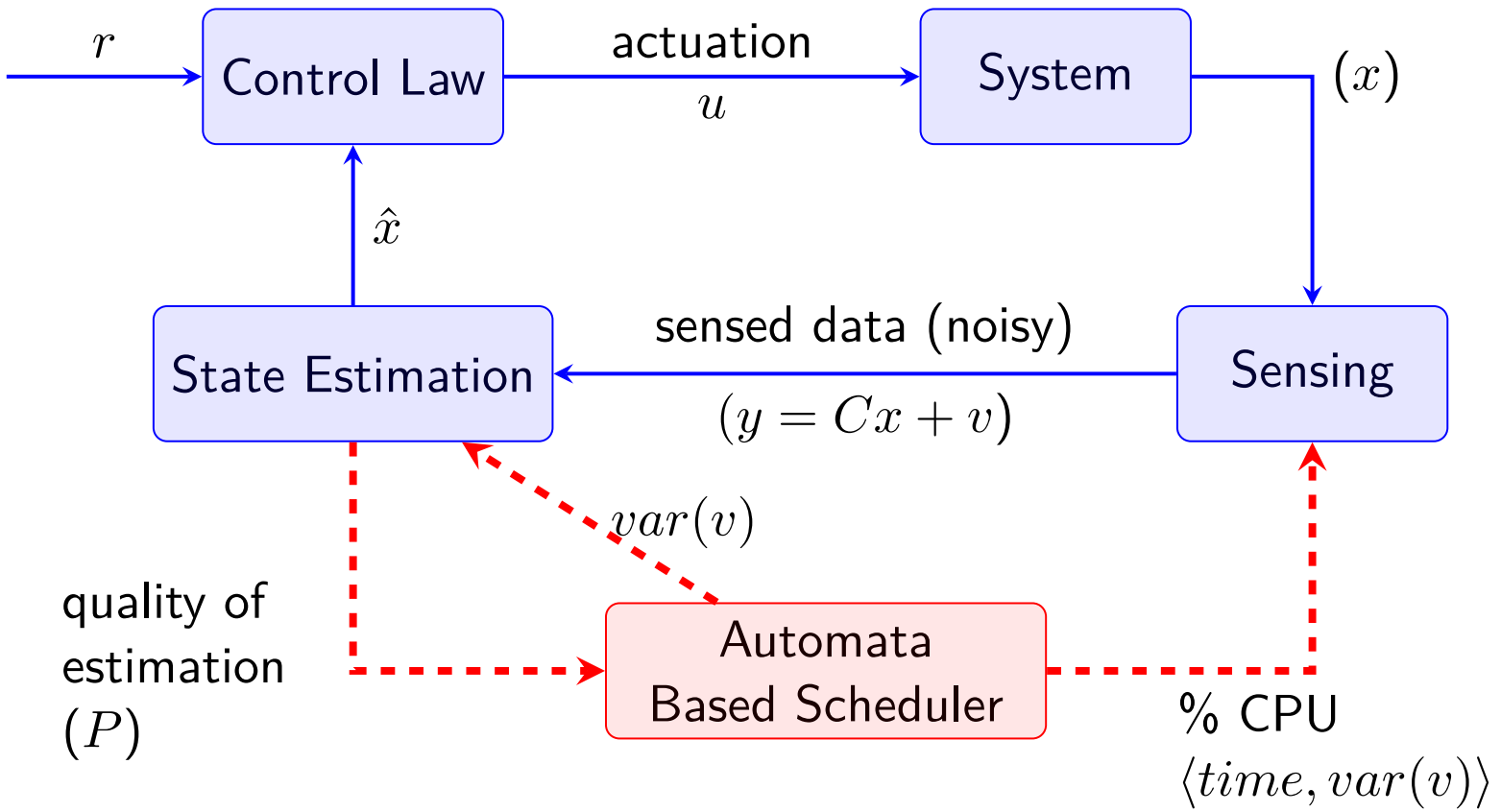
# The Proposed Architecture



# System Design
## The Proposed Architecture

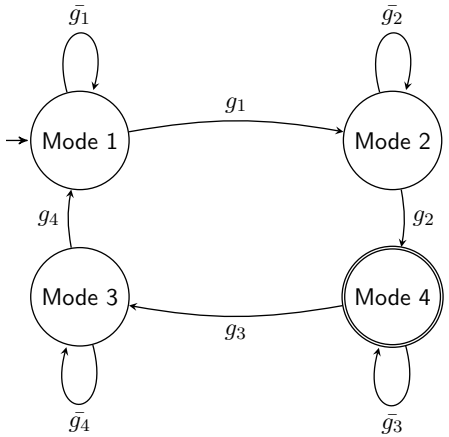
*Explain that the scheduler is involve in the control loops*



# Automata-Based Specification Interface

## The Proposed Architecture

*maybe add a word about RTcomposer and GameComposer*



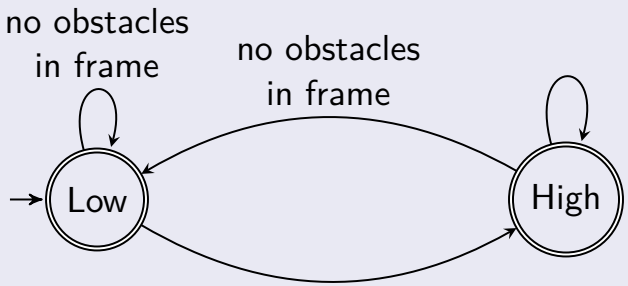

### Why Automata

- **Lite:** minimal resource consumption at run-time
- **Composable**: easy to compose independent components
- **Automata theory built in:** allows for tools such *GOAL*
- **Expressiveness**

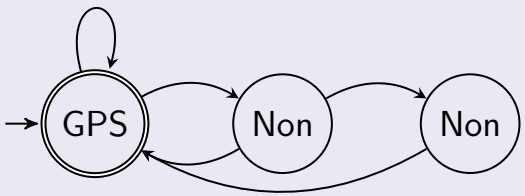
# Example of Guarded Automata

## The Proposed Architecture

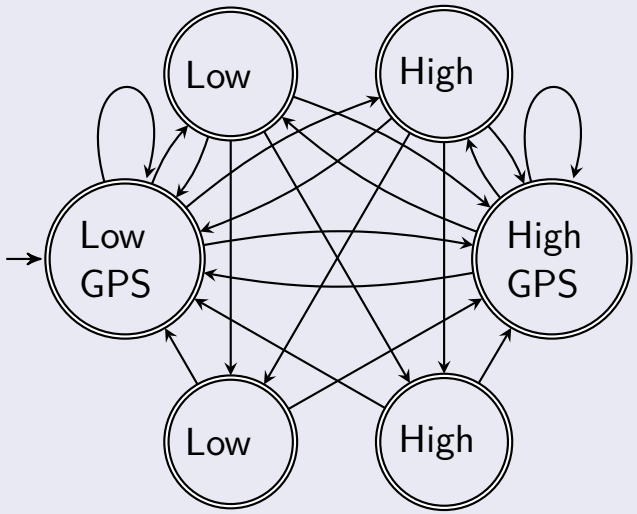
### Obstacle avoidance component




### GPS navigation component



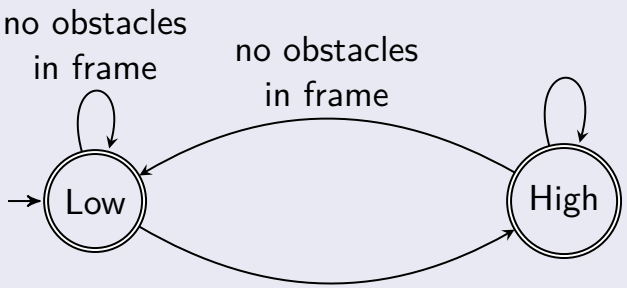

### Composed guarded automata



# Simplifying the Guarded Automata

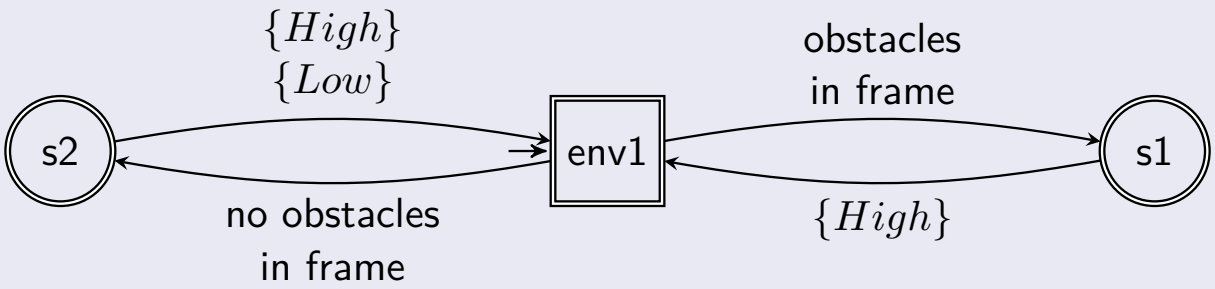
## The Proposed Architecture

### Mode-based guarded automata (for good intuition)



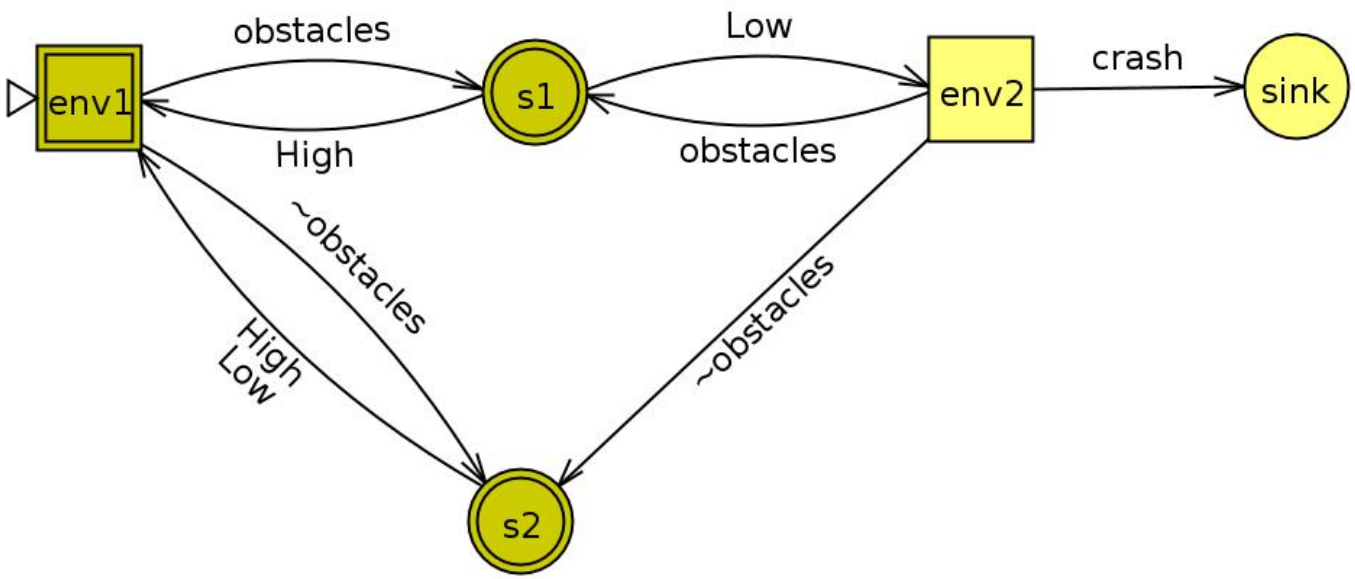

$\Downarrow$

### The automata in practice (best match $\omega$-word theory)

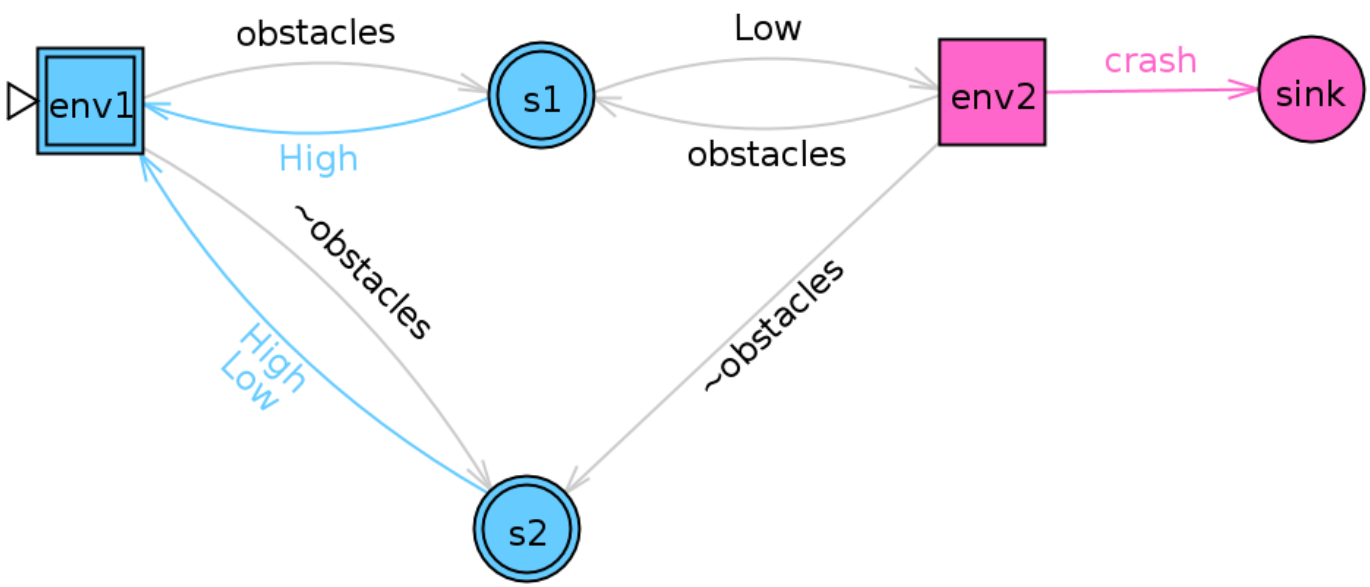



**Q: How to create the guarded automata?** By wining Büchi games

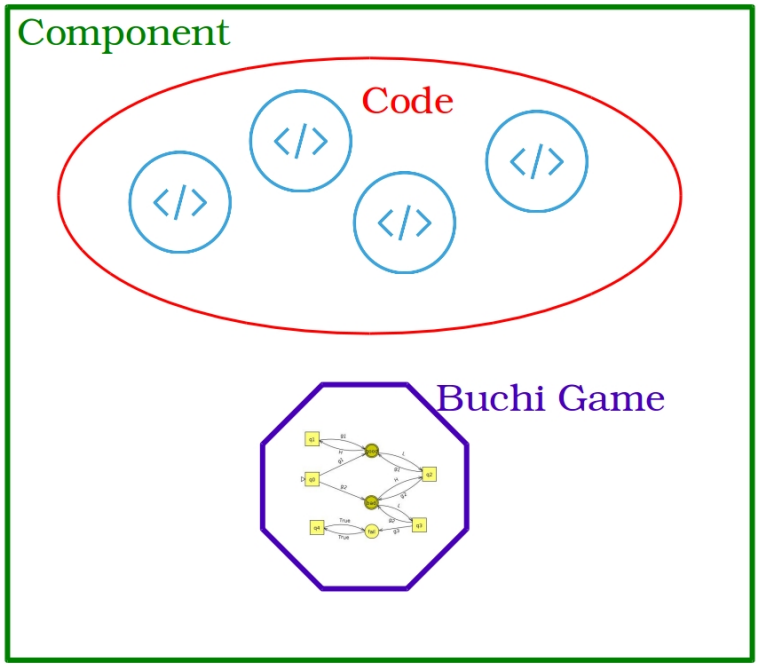
# Büchi game remainder



# Büchi game remainder



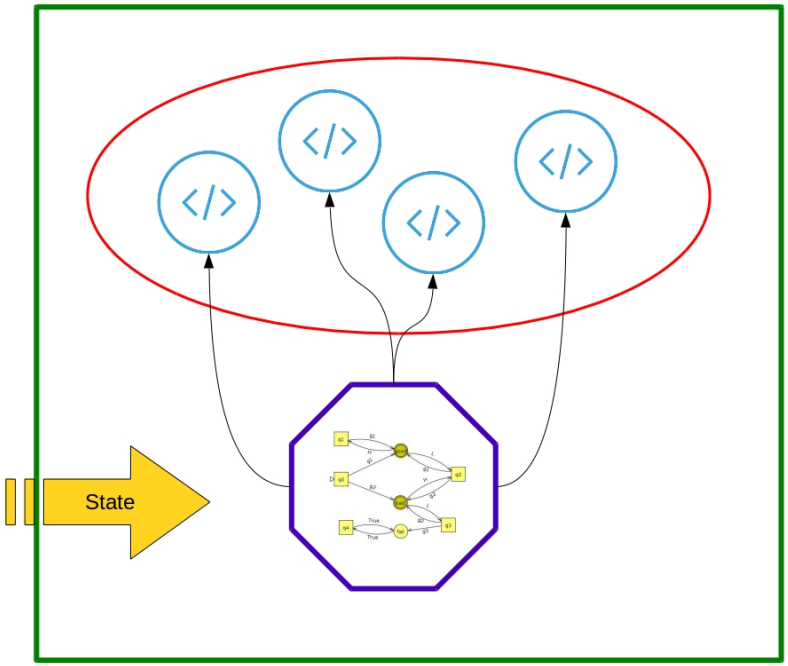
# A Component in the System




## Component Definition ($\langle T, G \rangle$)

- A set of subroutines (functions code)
- A Generalize Büchi Game

# A Component in the System

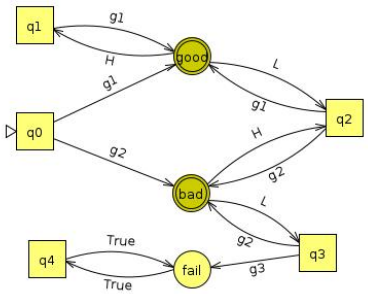



## The Büchi game ($G = \langle A, \langle P_{sched}, P_{env} \rangle \rangle$)

- Is played in turns by the **environment** and the **scheduler**
- Represent the **interaction** between the scheduler and the environment reaction

# Scheduling Büchi Game

## A Component in the System



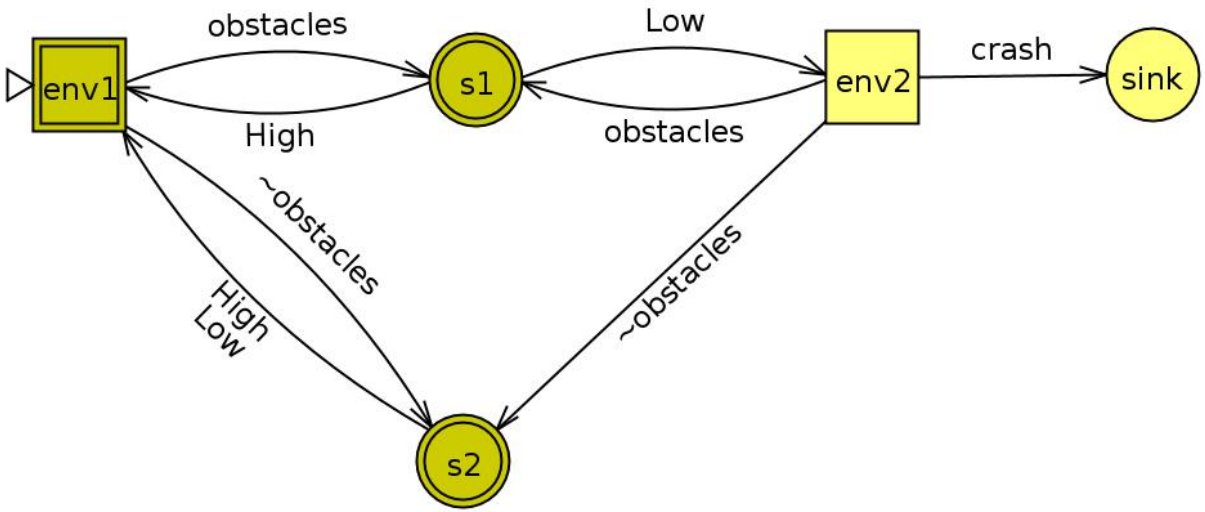

### Scheduling Büchi Game

- **Alternating turns**
- Scheduler alphabet is $\Sigma_{schd} = 2^T$
- Environment alphabet is $\Sigma_{env} = \mathbb{R}^n$ (*scheduler feedback variables*)
- There is an Edge for any **possible** environmental outcome
- The **scheduler feedback variables** can be any environment-depended value
- Environment player plays first

# Example - Büchi Game

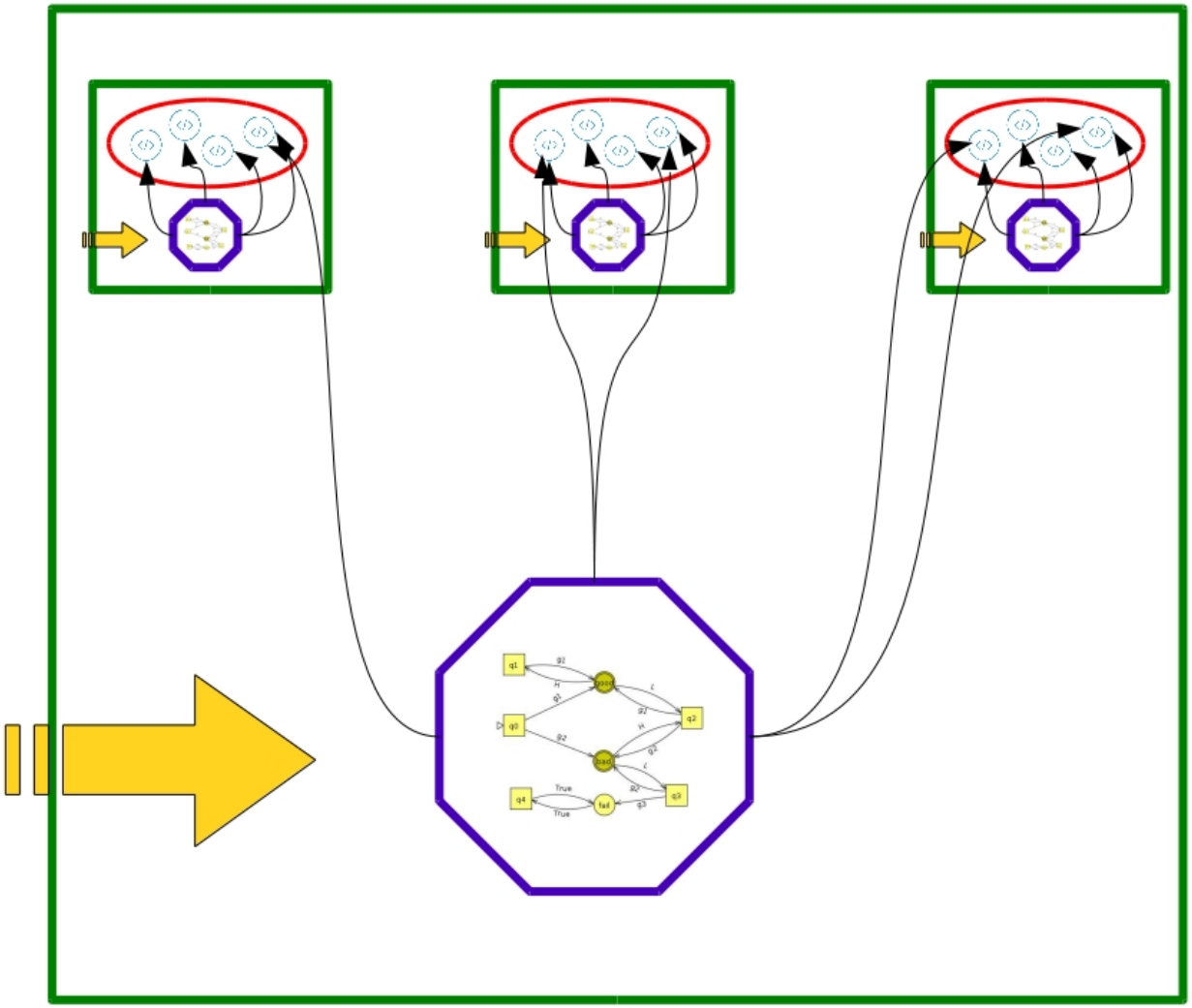
## A Component in the System

The Büchi Game of the obstacles avoidance component:

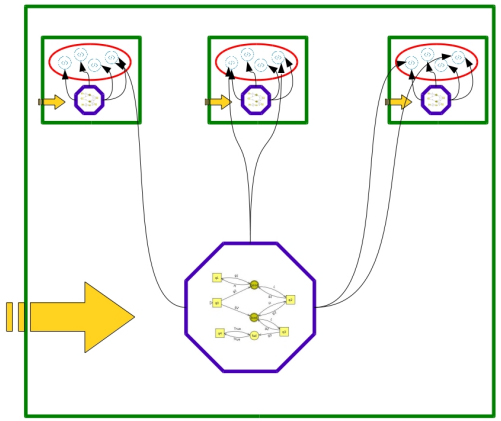



- The objectives of the component is to avoid obstacles
- The scheduler **win** $\Leftrightarrow$ the corresponding word $\omega \in \mathcal{L}(A) \Leftrightarrow$ the component achieved his **objectives**

# Component Composition



# Component Composition



## Requirements

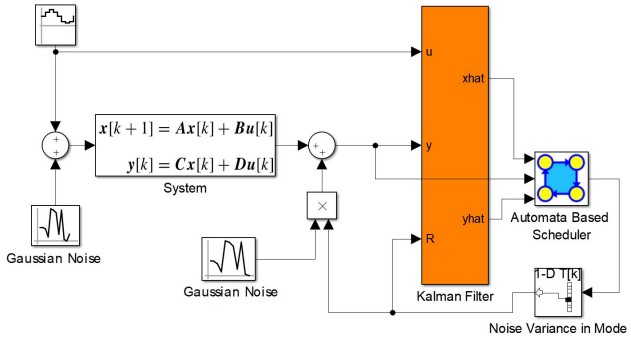
- A game ($G = \langle A, \langle P_s, P_e \rangle \rangle$) correspond to all the components
- The game of Component is $G_i = \langle A_i, \langle P_s^i, P_e^i \rangle \rangle$
- $\omega \in \mathcal{L}(A) \Leftrightarrow \forall i : \omega(i) \in \mathcal{L}(A_i)$

*TODO: how to present the composition details?*

*TODO: show the resource component*

*TODO: show the scheduler work: 1. find wining strategy 2. simultaneously walk through the strategy automata*

# Integration with Kalman



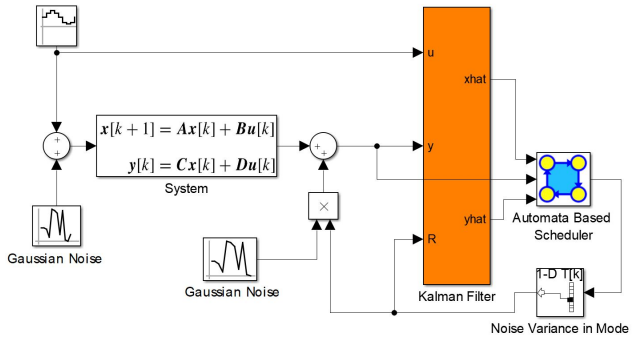

*Kalman filter figure*

## Resource utilization with Kalman filter

- Novel technique for on-line trade-off between estimation quality and resource consumption
- Evaluate the overall errors using Kalman filter
- Schedule sensing-tasks based on the estimation quality

# Integration with Kalman

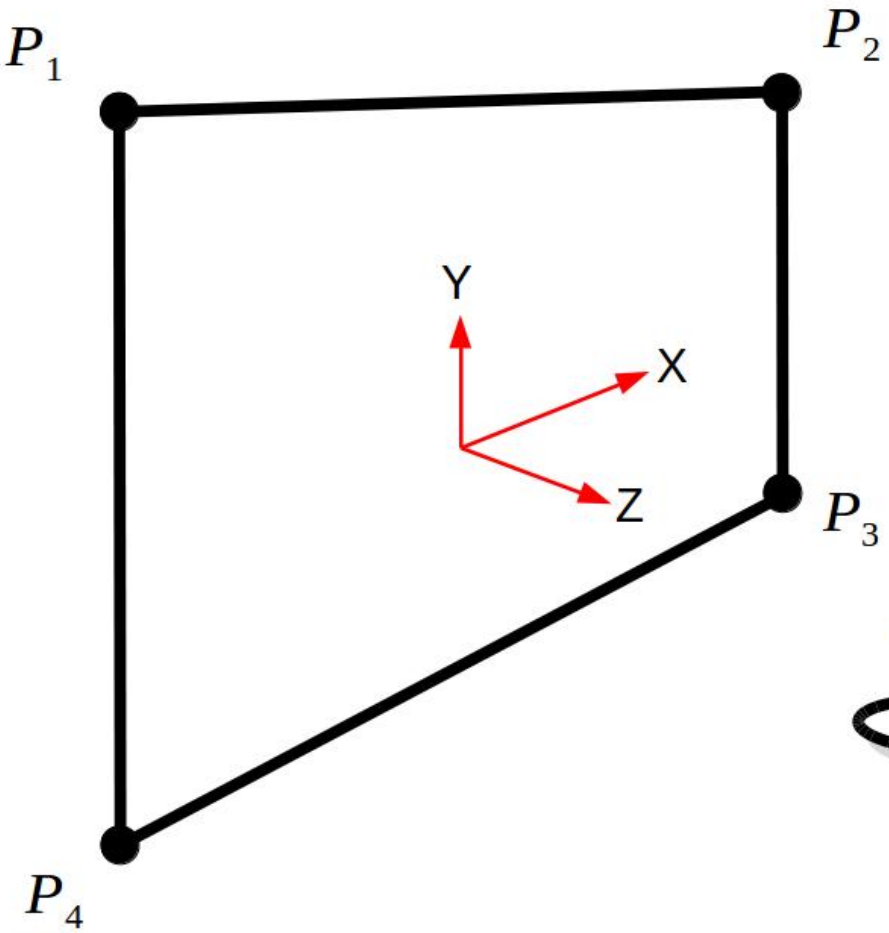
*Explain the concept of estimate the errors*

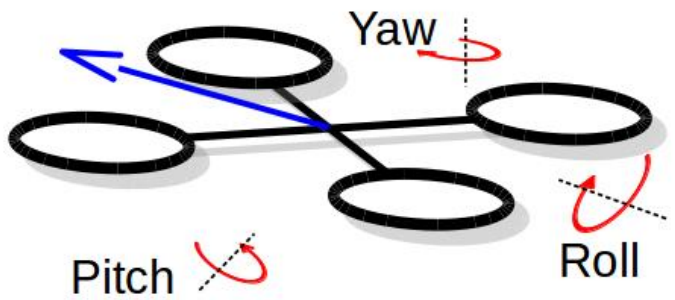


*TODO*

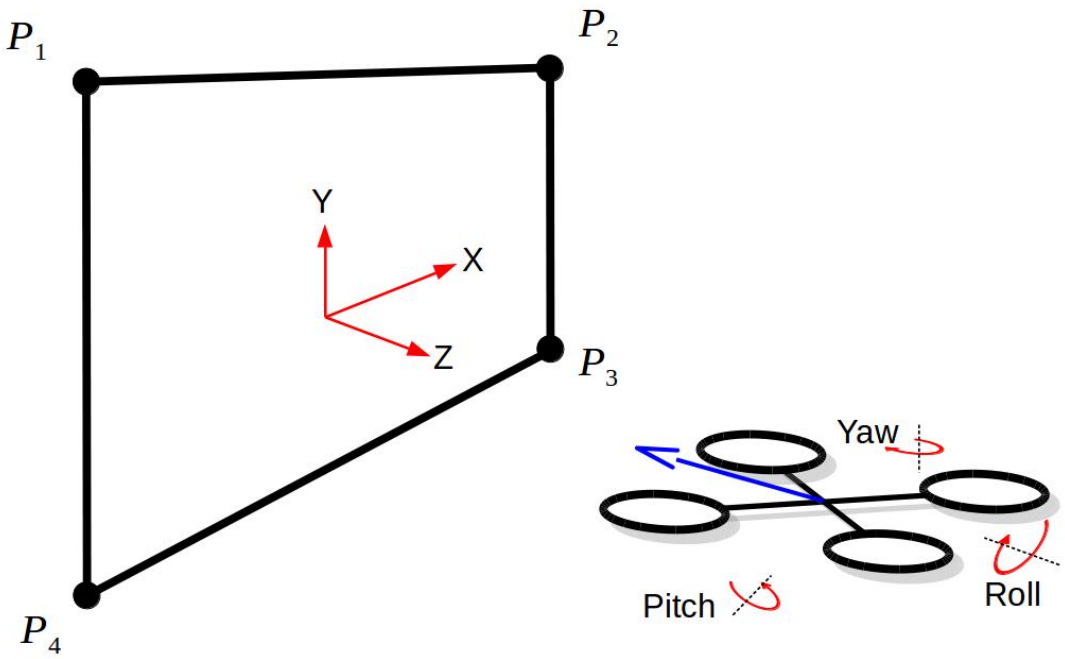
# Mission Definition

*Explain the window motivation*



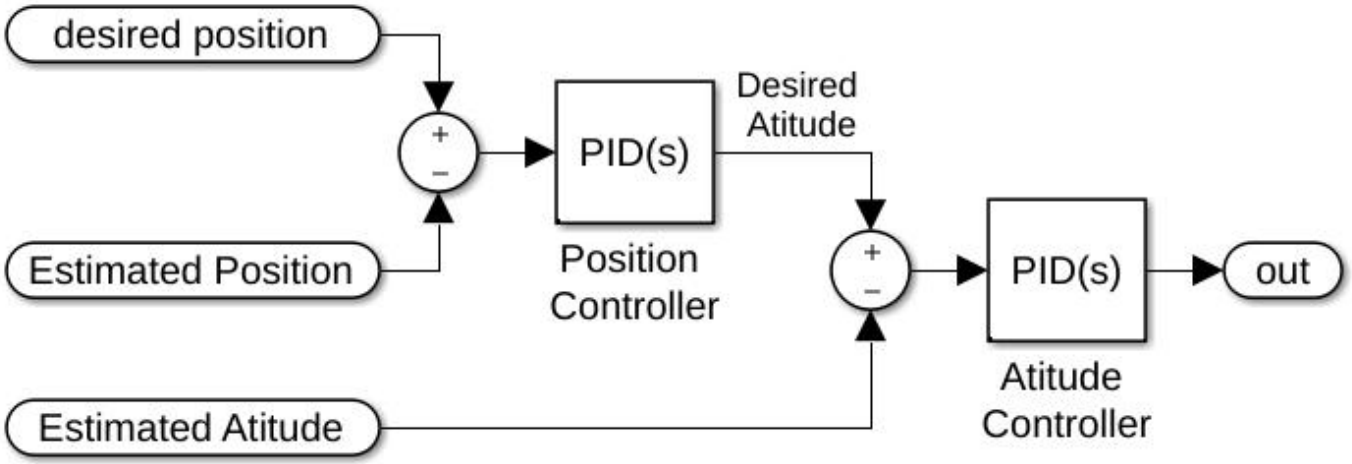
# Concrete Control Objectives




## Control & Scheduling Objectives

- Minimize the $x$-deviation
- Minimize the CPU usage of image processing task

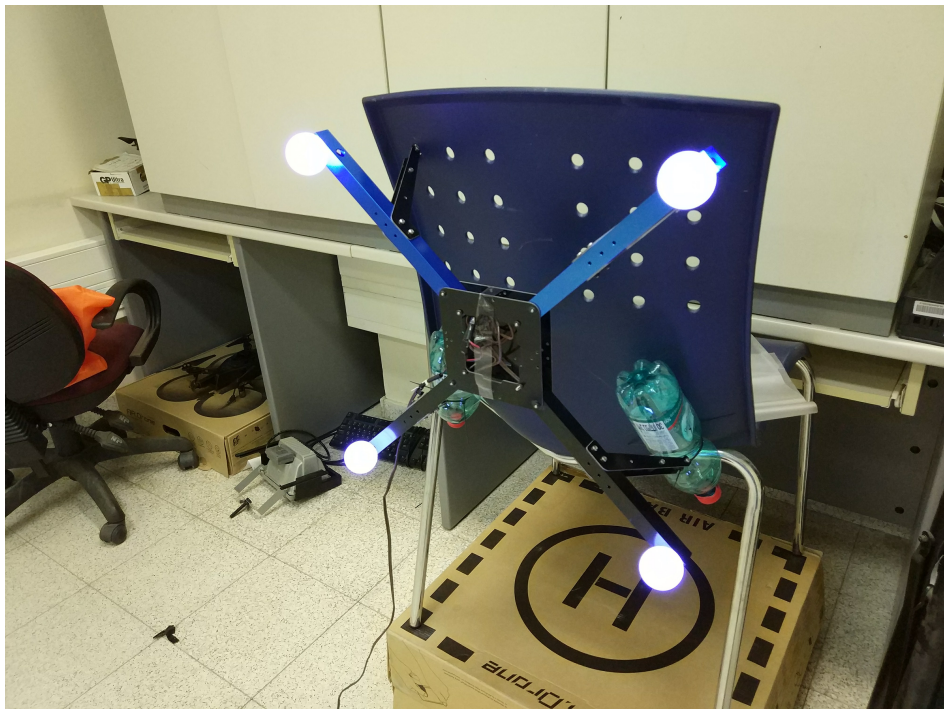
# Traditional Controller Design




## Attitude and position controller

- **vision** component estimate the $x$-position
- Position controller output a desired roll angle
- Attitude controller is a traditional attitude controller

# Vision Component



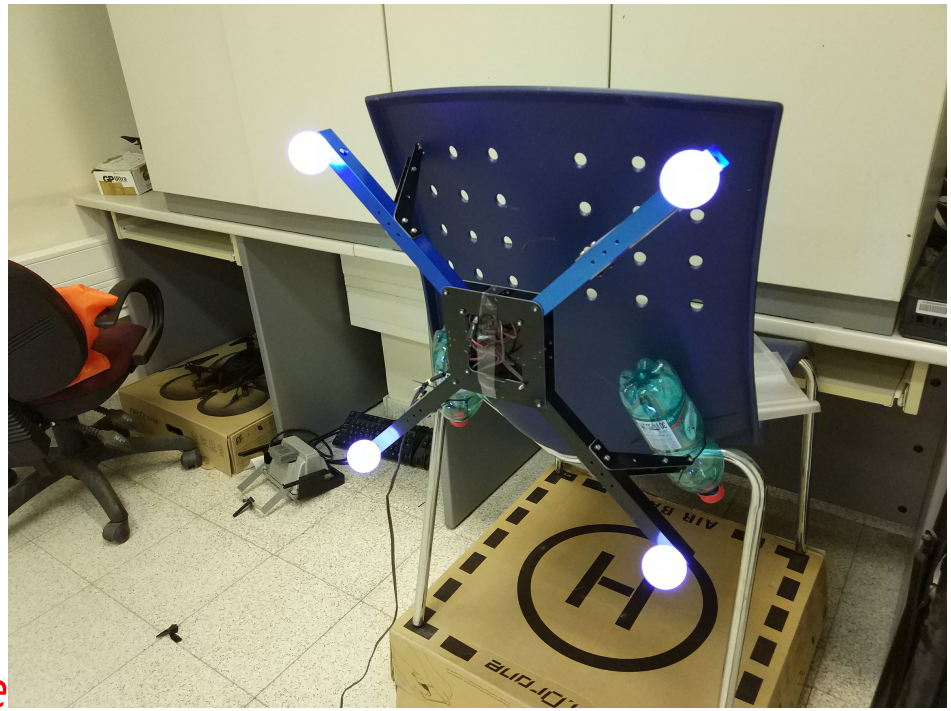
## Image Processing Algorithm

1. Find the window corners (brute force search)
2. Calculate the drone position

# Calculate the Drone Position

## Vertical Difference

*take a picture*

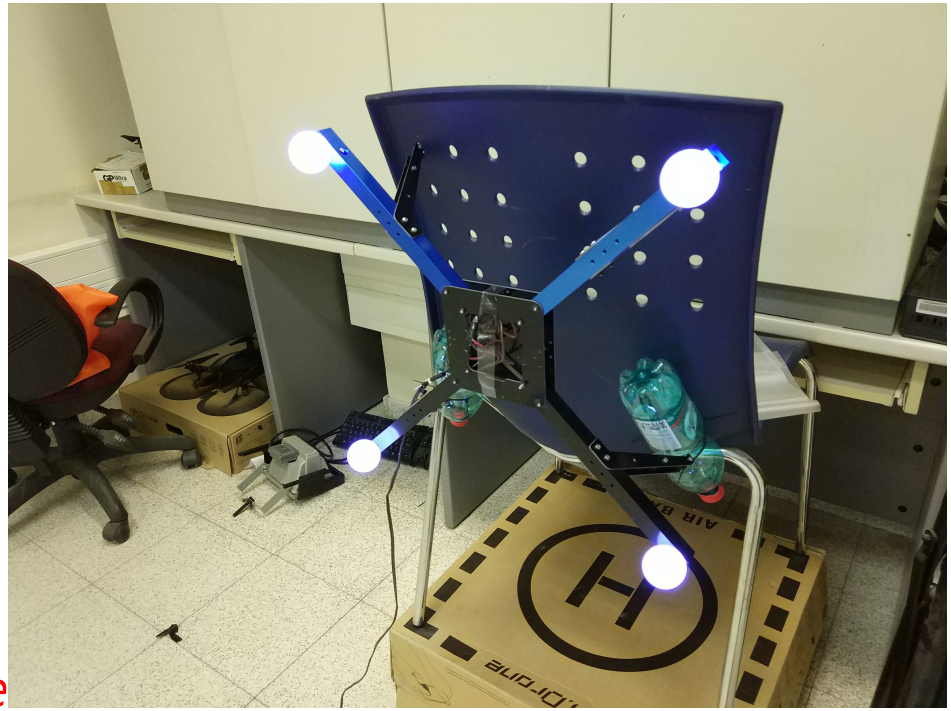


$$V_d = \frac{((y_1 - y_4) - (y_2 - y_3))}{((y_1 - y_4) + (y_2 - y_3))}$$

# Calculate the Drone Position

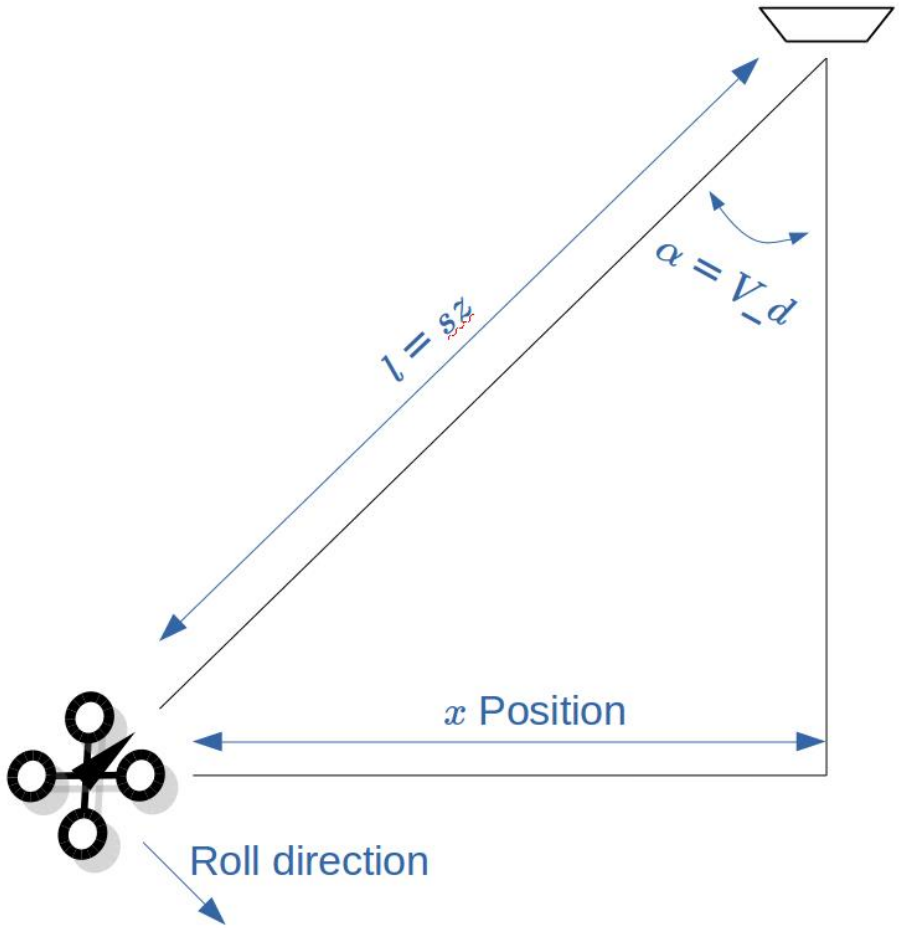
## Center of Mass

*take a picture*



$$S_x = \frac{x_1+x_2+x_3+x_4}{4}$$

# Calculate the Drone Position




*take a picture*

$$x = l \cdot \sin(V_d) \approx l \cdot V_d$$

*1. why not Kalman2. how we use complementary filter3. the linearize model in $x$ / roll axis4. update state (equations) the automata and their results*

*instead of with Related Work review of similar papers: A table with few papers*

# Thanks