Quadcopter Navigation through Obstacles using Potential Field

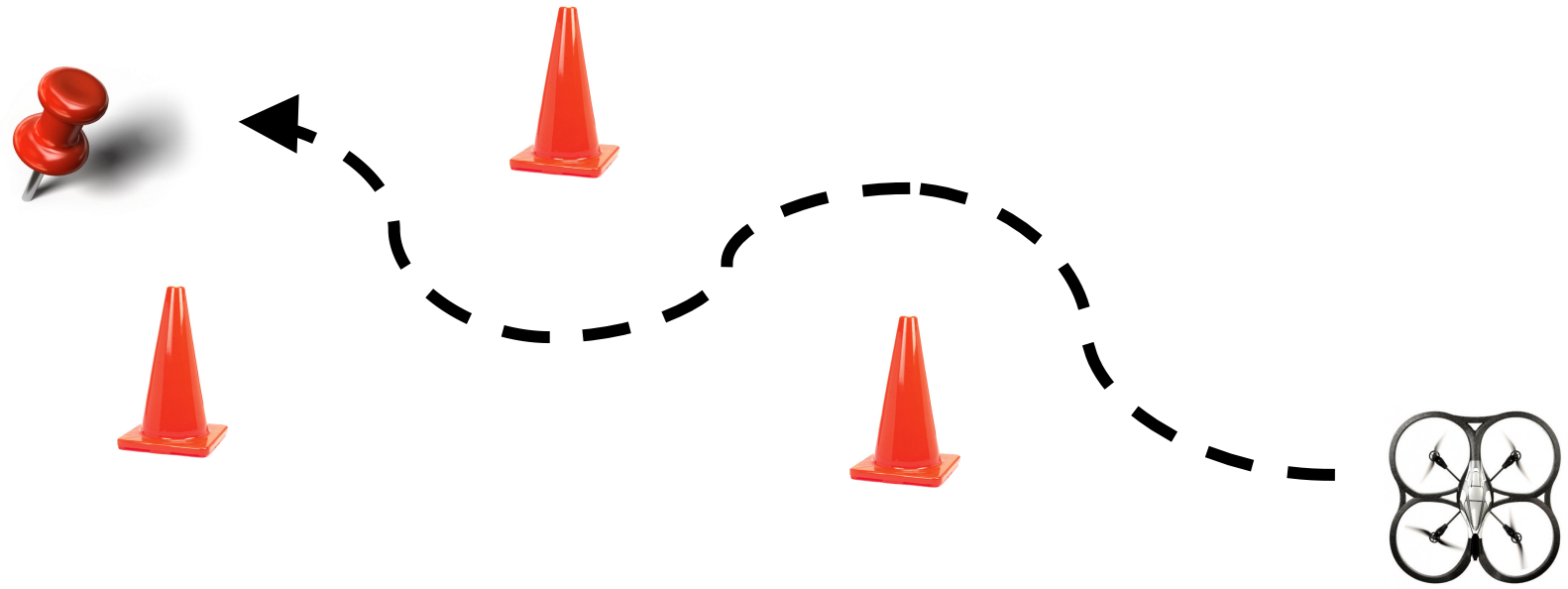
Visual Navigation for Flying Robots

Summer Semester 2013

K. Neczaj & D. Susanto

Idea

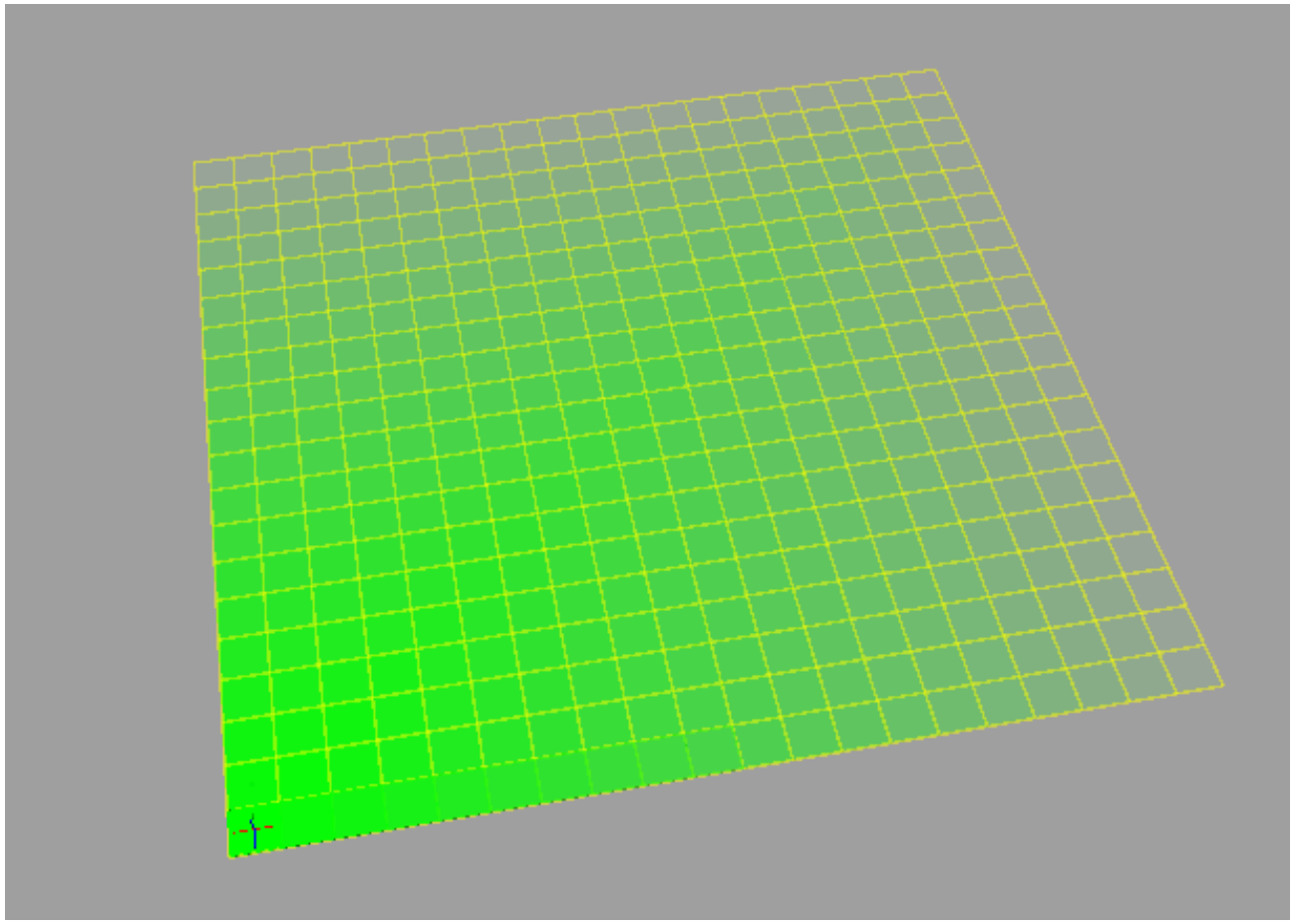
Navigate the quadcopter **autonomously** to the goal point by **avoiding obstacles** on the path?



Approach

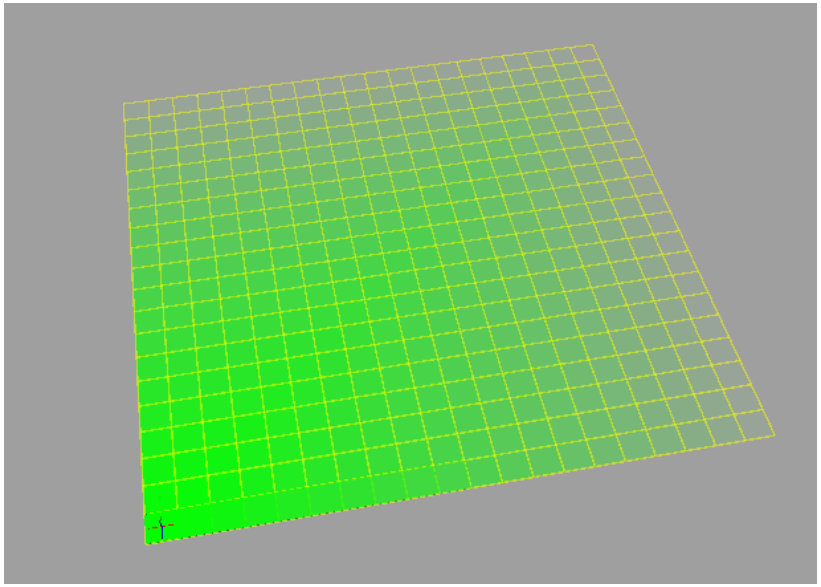
- Discretization of the environment with **grid**.
- Apply **potential field** to the environment grid.
- Detect obstacles with **markers**.
- **Convolve** the obstacles with **Gaussian** kernel.
- Control correction using **PID controller**.
- Position correction using **Kalman filter**.

Potential Field



Potential Field

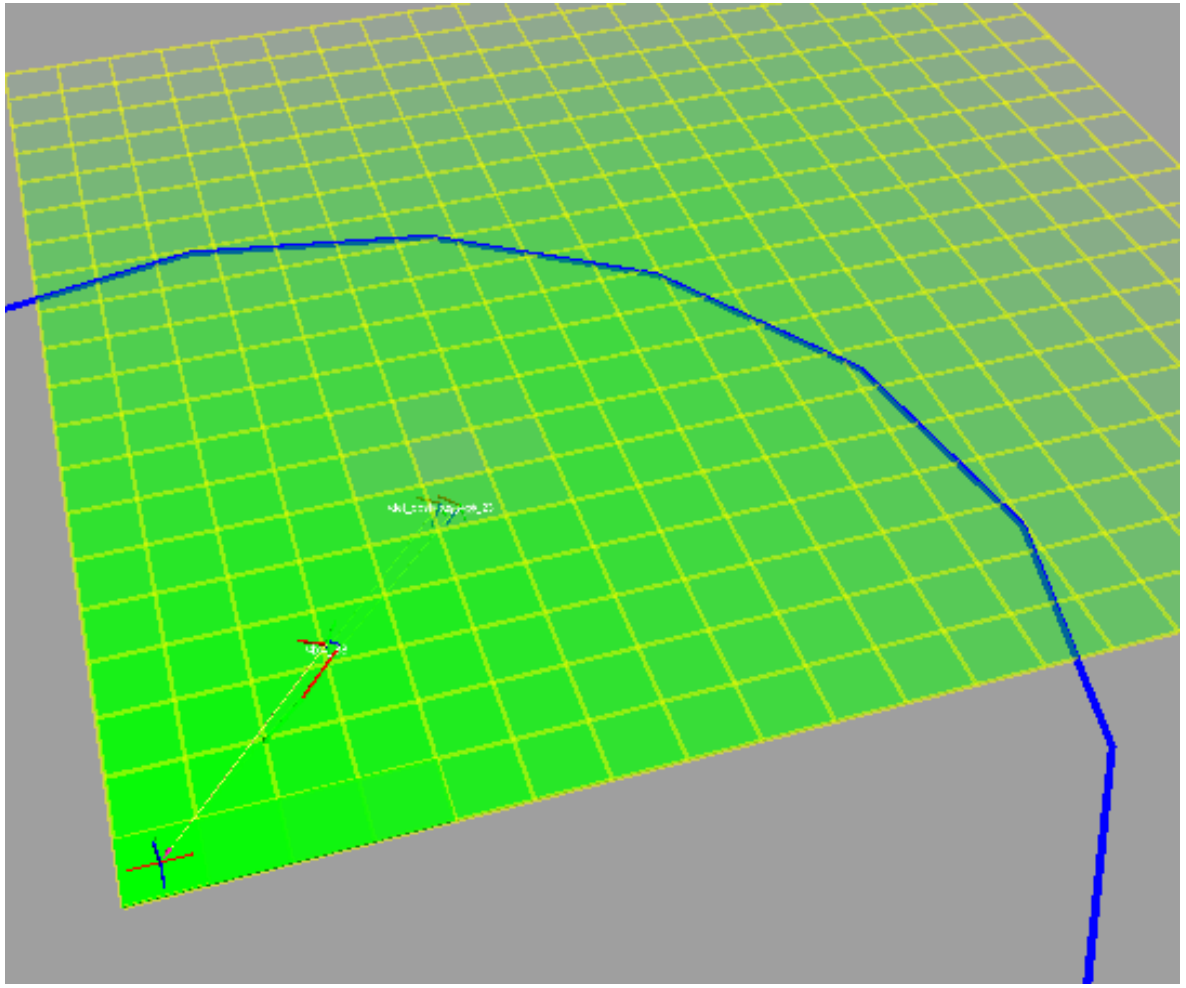
```
POTENTIAL FIELD (20)
0     5     10    15    20    25    30    35    40    45    50    55    60    65    70    75    80    85    90    95
5     5     10    15    20    25    30    35    40    45    50    55    60    65    70    75    80    85    90    95
10    10    10    15    20    25    30    35    40    45    50    55    60    65    70    75    80    85    90    95
15    15    15    15    20    25    30    35    40    45    50    55    60    65    70    75    80    85    90    95
20    20    20    20    20    25    30    35    40    45    50    55    60    65    70    75    80    85    90    95
25    25    25    25    25    25    30    35    40    45    50    55    60    65    70    75    80    85    90    95
30    30    30    30    30    30    30    35    40    45    50    55    60    65    70    75    80    85    90    95
35    35    35    35    35    35    35    35    40    45    50    55    60    65    70    75    80    85    90    95
40    40    40    40    40    40    40    40    40    45    50    55    60    65    70    75    80    85    90    95
45    45    45    45    45    45    45    45    45    45    50    55    60    65    70    75    80    85    90    95
50    50    50    50    50    50    50    50    50    50    50    55    60    65    70    75    80    85    90    95
55    55    55    55    55    55    55    55    55    55    55    55    60    65    70    75    80    85    90    95
60    60    60    60    60    60    60    60    60    60    60    60    60    65    70    75    80    85    90    95
65    65    65    65    65    65    65    65    65    65    65    65    65    65    70    75    80    85    90    95
70    70    70    70    70    70    70    70    70    70    70    70    70    70    70    75    80    85    90    95
75    75    75    75    75    75    75    75    75    75    75    75    75    75    75    75    80    85    90    95
80    80    80    80    80    80    80    80    80    80    80    80    80    80    80    80    80    85    90    95
85    85    85    85    85    85    85    85    85    85    85    85    85    85    85    85    85    85    90    95
90    90    90    90    90    90    90    90    90    90    90    90    90    90    90    90    90    90    90    95
95    95    95    95    95    95    95    95    95    95    95    95    95    95    95    95    95    95    95    100

(18,18)
(17,17)
(16,16)
(15,15)
(14,14)
(13,13)
(12,12)
(11,11)
(10,10)
(9,9)
(8,8)
(7,7)
(6,6)
(5,5)
(4,4)
(3,3)
(2,2)
(1,1)
(0,0)
```



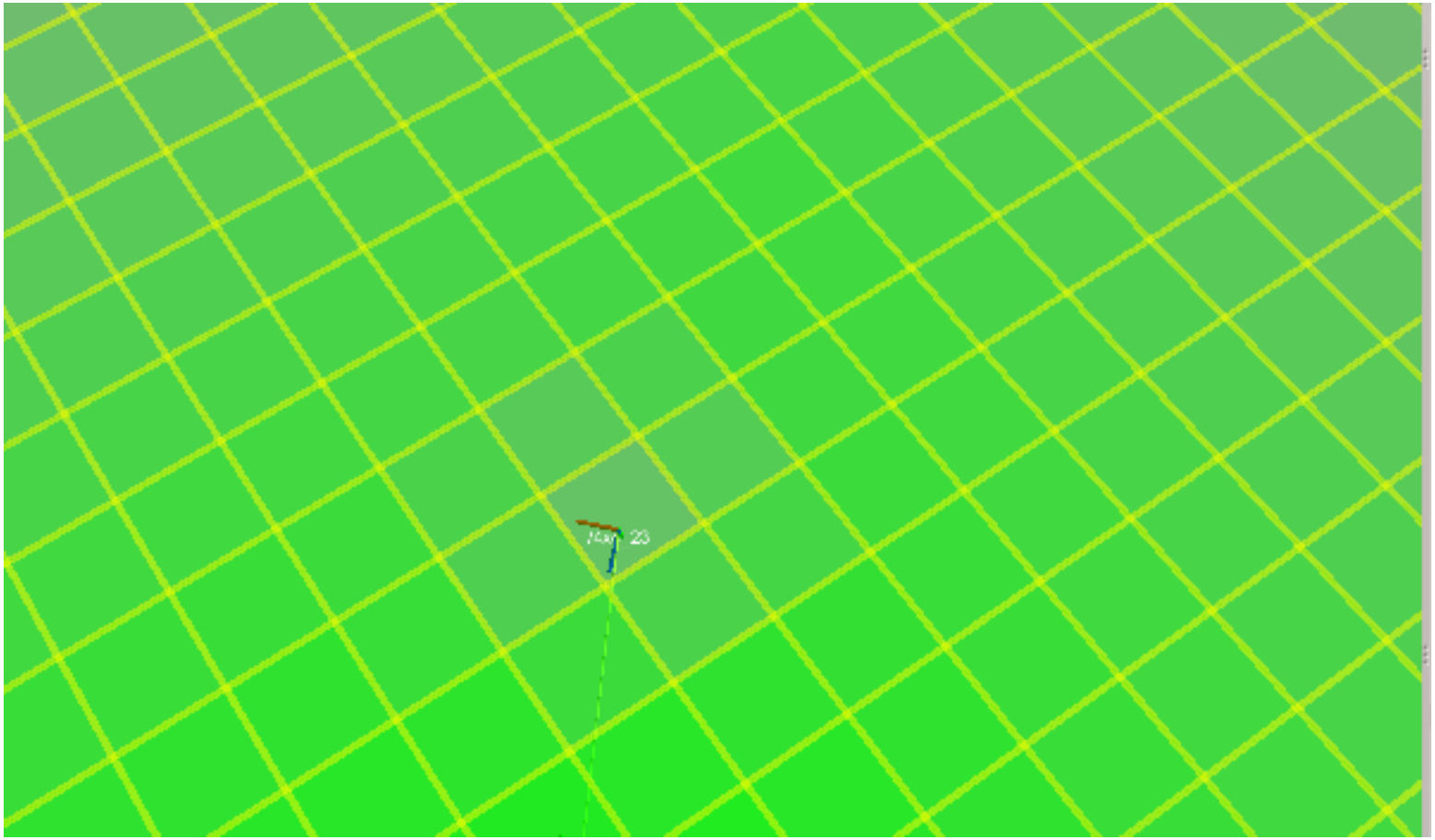
Approach

- ~~Discretization of the environment with **grid**.~~
- ~~Apply **potential field** to the environment grid.~~
- Detect obstacles with **markers**.
- **Convolve** the obstacles with **Gaussian** kernel.
- Control correction using **PID controller**.
- Position correction using **Kalman filter**.

Obstacles Detection



Convolve Obstacle's Potential



Approach

- ~~Discretization of the environment with **grid**.~~
- ~~Apply **potential field** to the environment grid.~~
- ~~Detect obstacles with **markers**.~~
- ~~**Convolve** the obstacles with **Gaussian** kernel.~~
- Control correction using **PID controller**.
- Position correction using **Kalman filter**.

PID Controller

Approach

- ~~Discretization of the environment with **grid**.~~
- ~~Apply **potential field** to the environment grid.~~
- ~~Detect obstacles with **markers**.~~
- ~~**Convolve** the obstacles with **Gaussian** kernel.~~
- Control correction using **PID controller**.
- Position correction using **Kalman filter**.