Topic 13

Evaluating Pose Likelihood

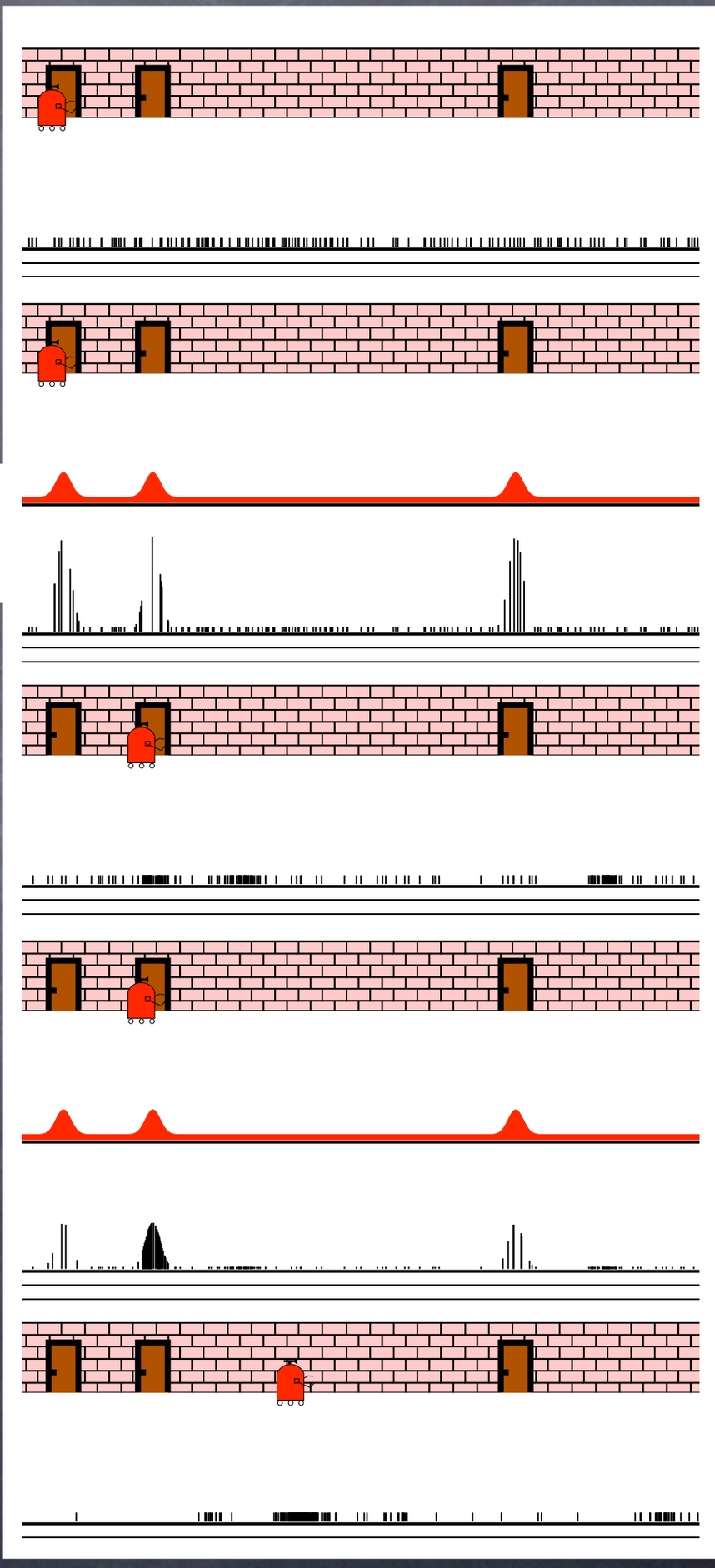
Particle filter recap

posterior likelihood dynamics prior

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \alpha\, p(\mathbf{Z}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})$$

"belief at t=k" "update" "predict" "belief at t=k-1"

- Bayes filter model to estimate true pose from all possible poses
- recursive Markovian inference over time with predict-update loop
- Particle filter algorithms predict-update with sample set of poses



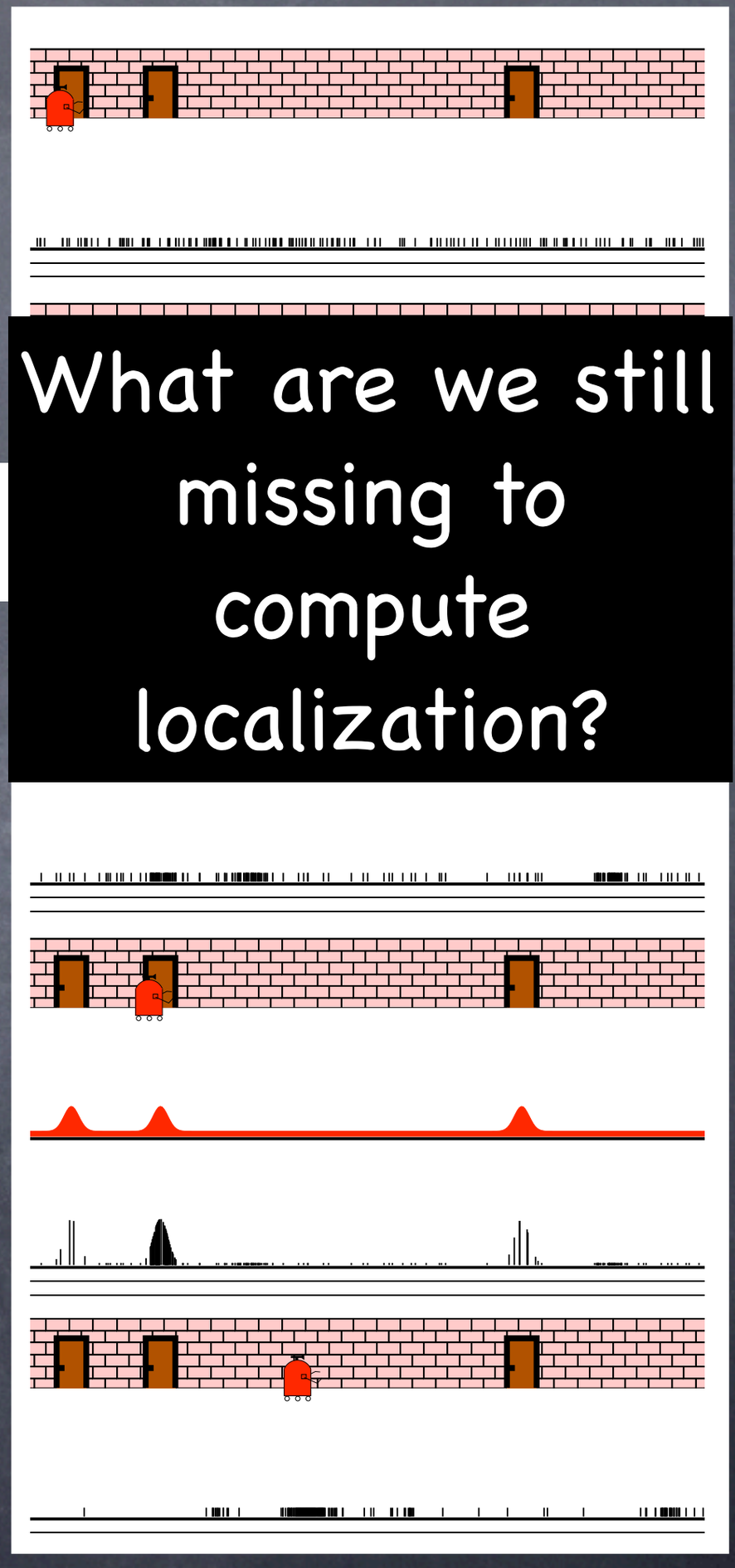
Particle filter recap

posterior likelihood dynamics prior

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \alpha\, p(\mathbf{Z}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})$$

"belief at t=k" "update" "predict" "belief at t=k-1"

- Bayes filter model to estimate true pose from all possible poses
- recursive Markovian inference over time with predict-update loop
- Particle filter algorithms predict-update with sample set of poses



What are we still missing to compute localization?

Particle filter recap

posterior likelihood dynamics prior

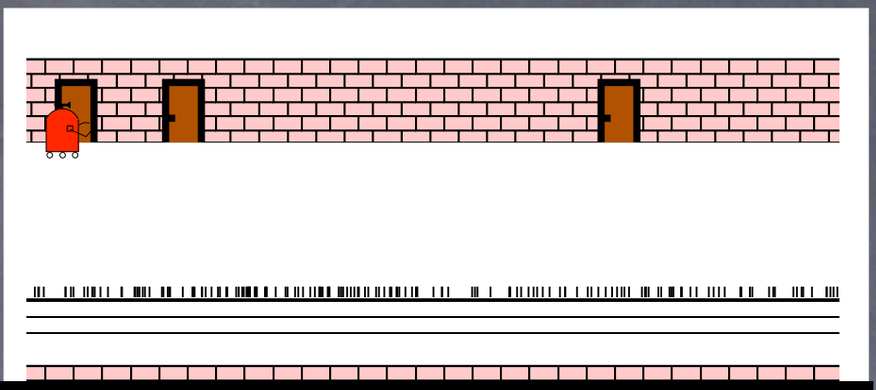
$$p(\mathbf{x}_k|\mathbf{Z}_k) = \alpha p(\mathbf{Z}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})$$

"belief at t=k" "update" "predict" "belief at t=k-1"

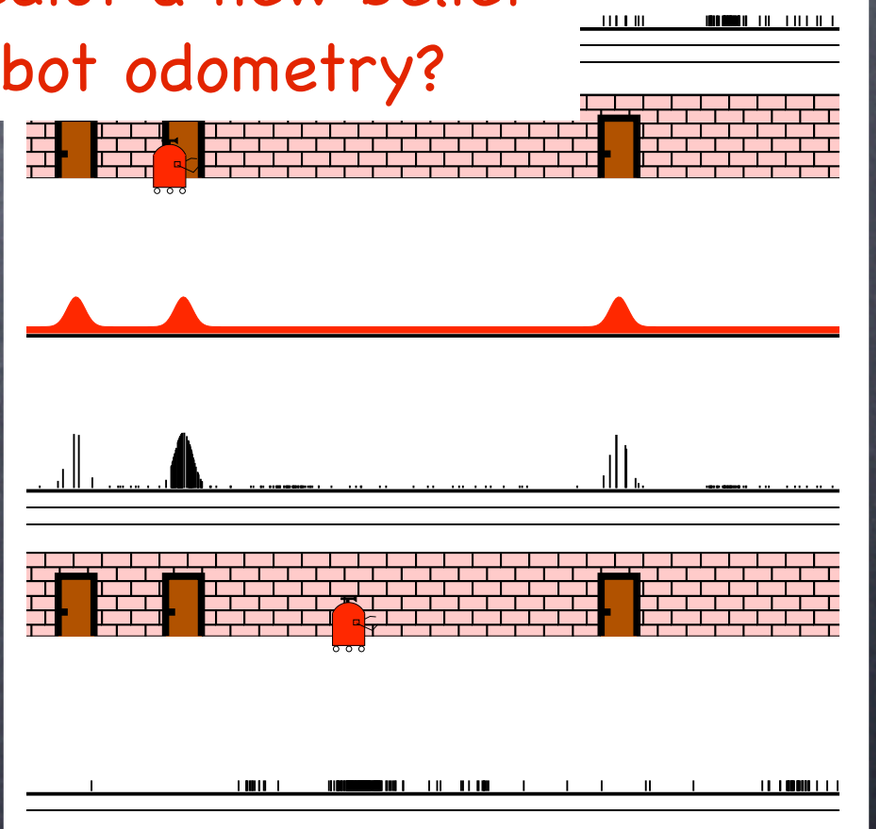
How to evaluate the likelihood of a pose given robot observations?

How to predict a new belief given robot odometry?

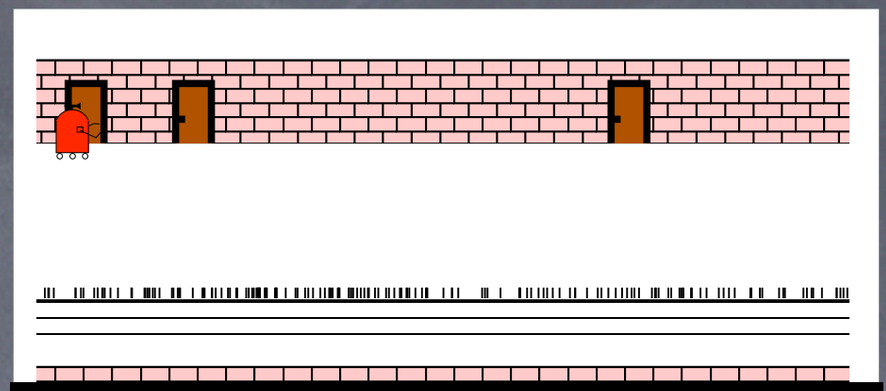
- recursive Markovian inference over time with predict-update loop
- Particle filter algorithms predict-update with sample set of poses



What are we still missing to compute localization?



Particle filter recap



What are we still missing to compute localization?

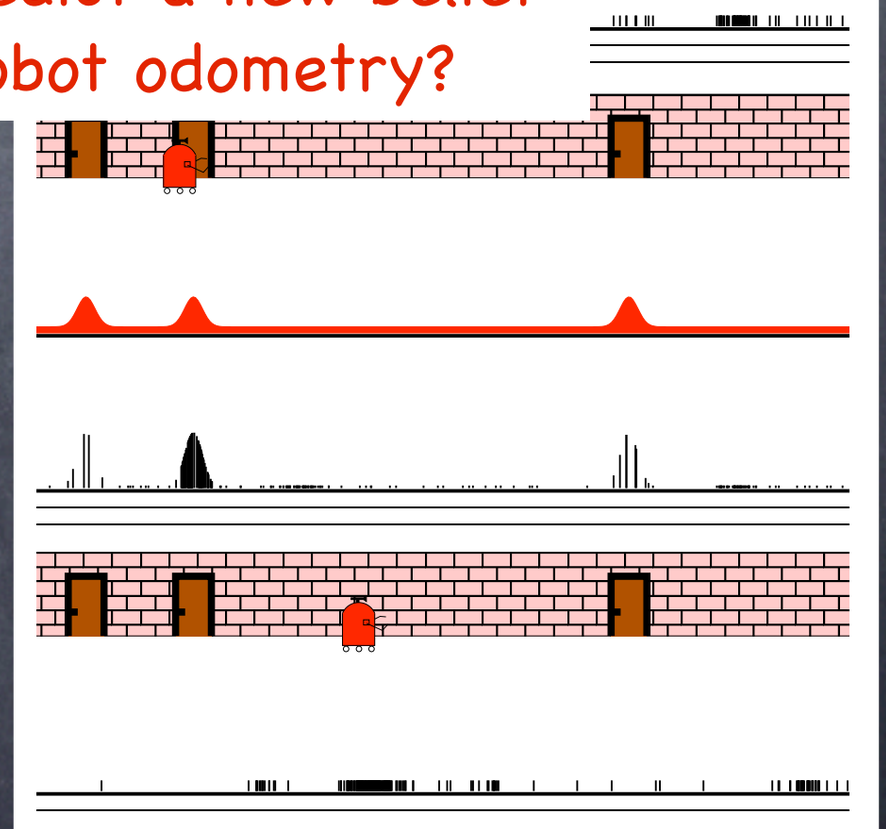
posterior likelihood dynamics prior

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \alpha p(\mathbf{Z}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})$$

"belief at t=k" "update" "predict" "belief at t=k-1"

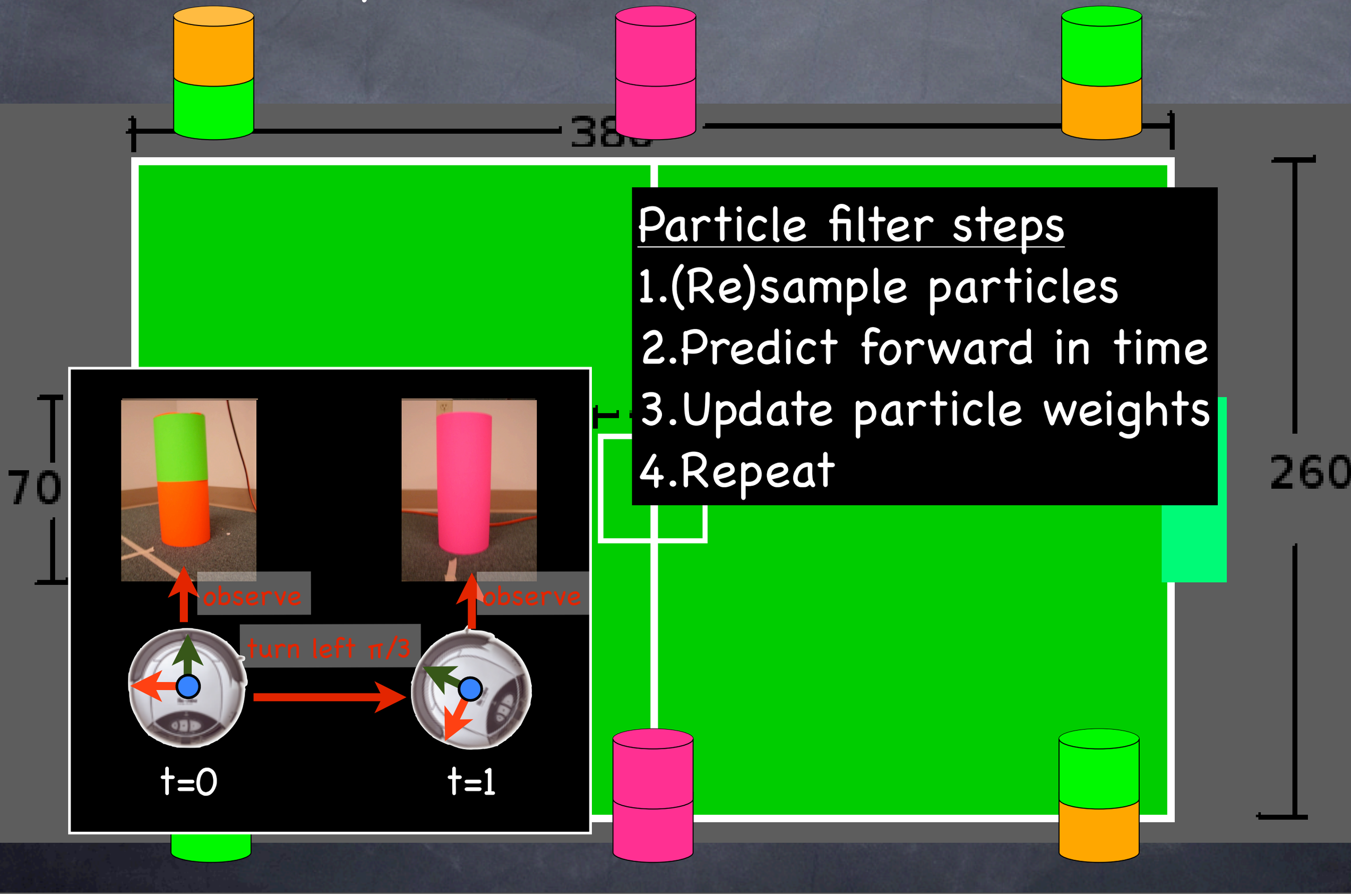
How to evaluate the likelihood of a pose given robot observations?

How to predict a new belief given robot odometry?



- recursive Markovian inference over time with predict-update loop
- Particle filter algorithms predict-update with sample set of poses

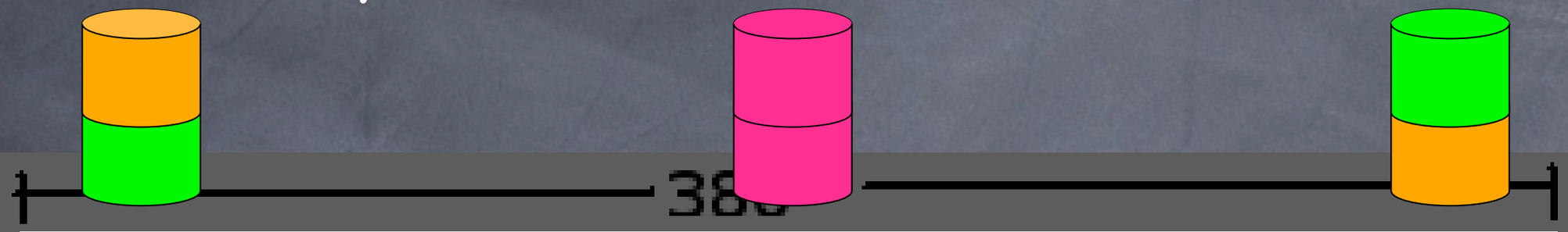
suppose the robot starts by seeing this landmark,
then turns left, and then sees another landmark



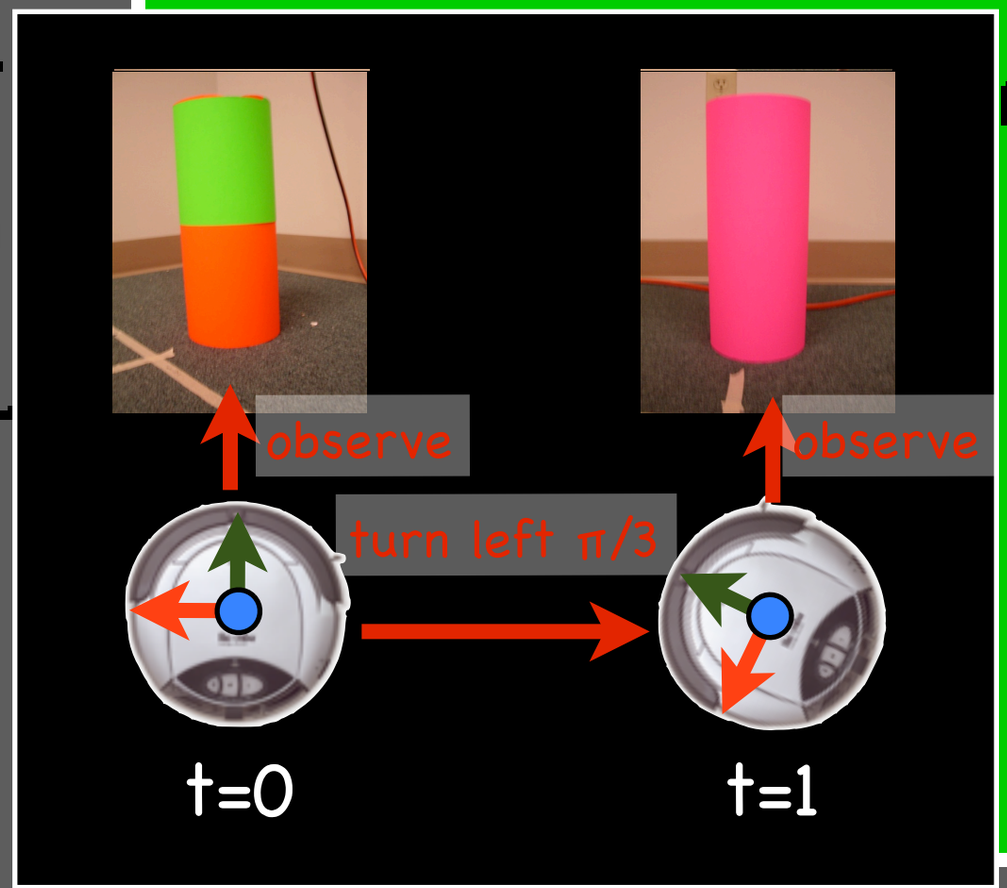
Particle filter steps

1. (Re)sample particles
2. Predict forward in time
3. Update particle weights
4. Repeat

suppose the robot starts by seeing this landmark, then turns left, and then sees another landmark



Initially, the robot has no observations



Particle filter steps

1.(Re)sample particles

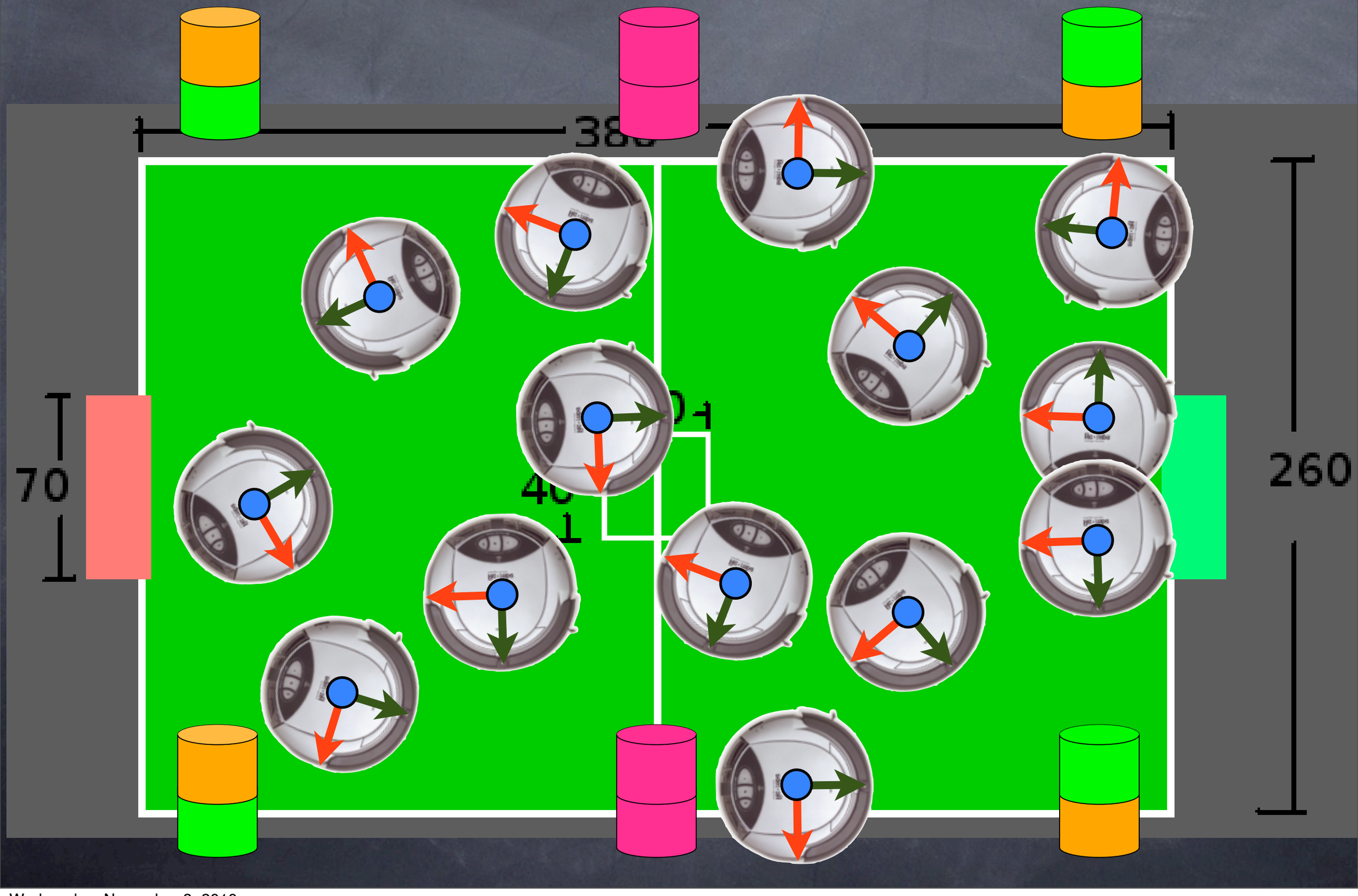
2.Predict forward in time

3.Update particle weights

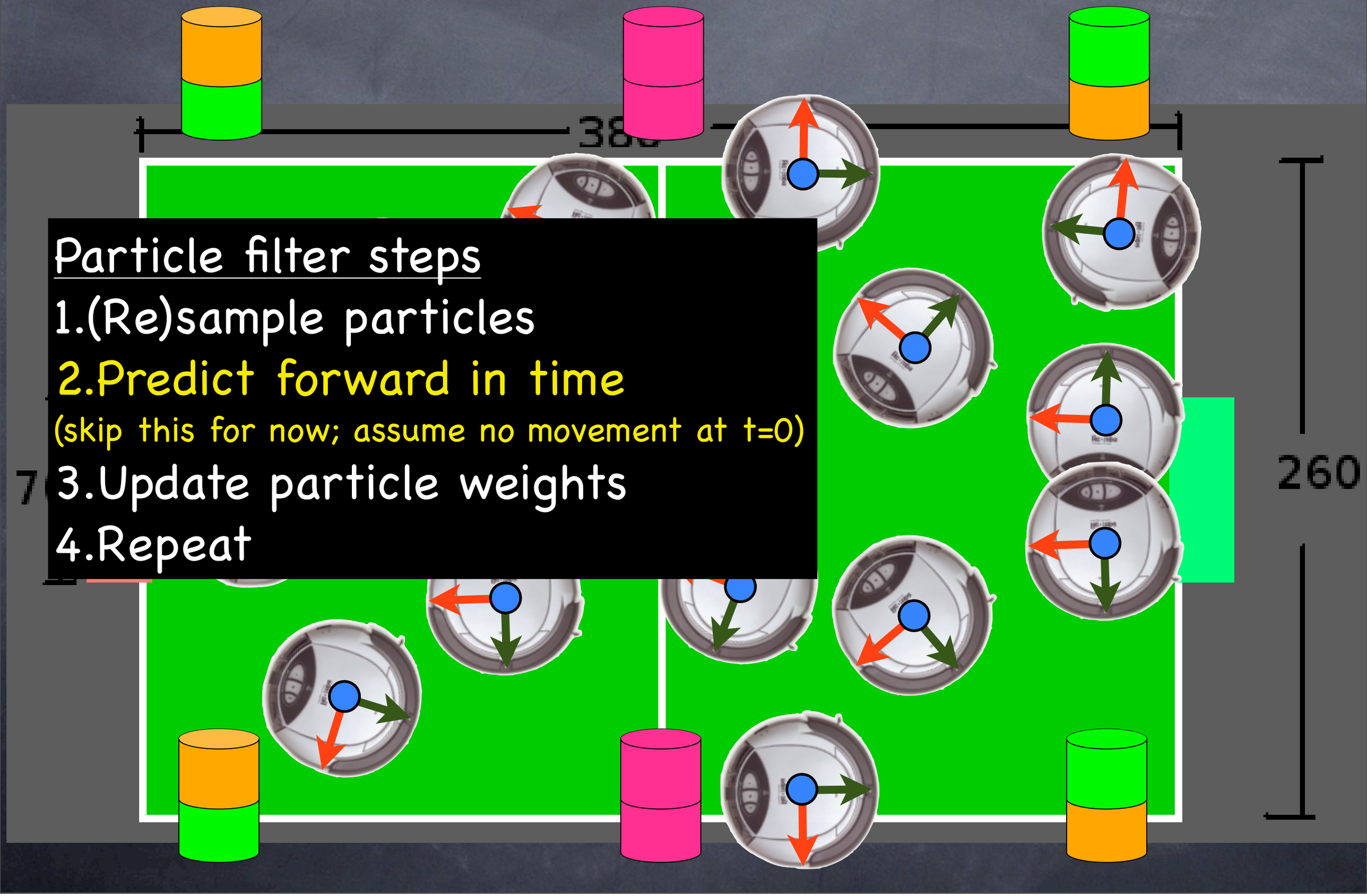
4.Repeat

What should happen during initial sampling?

poses sampled from a uniform distribution across pitch



poses sampled from a uniform distribution across pitch



Particle filter steps

1.(Re)sample particles

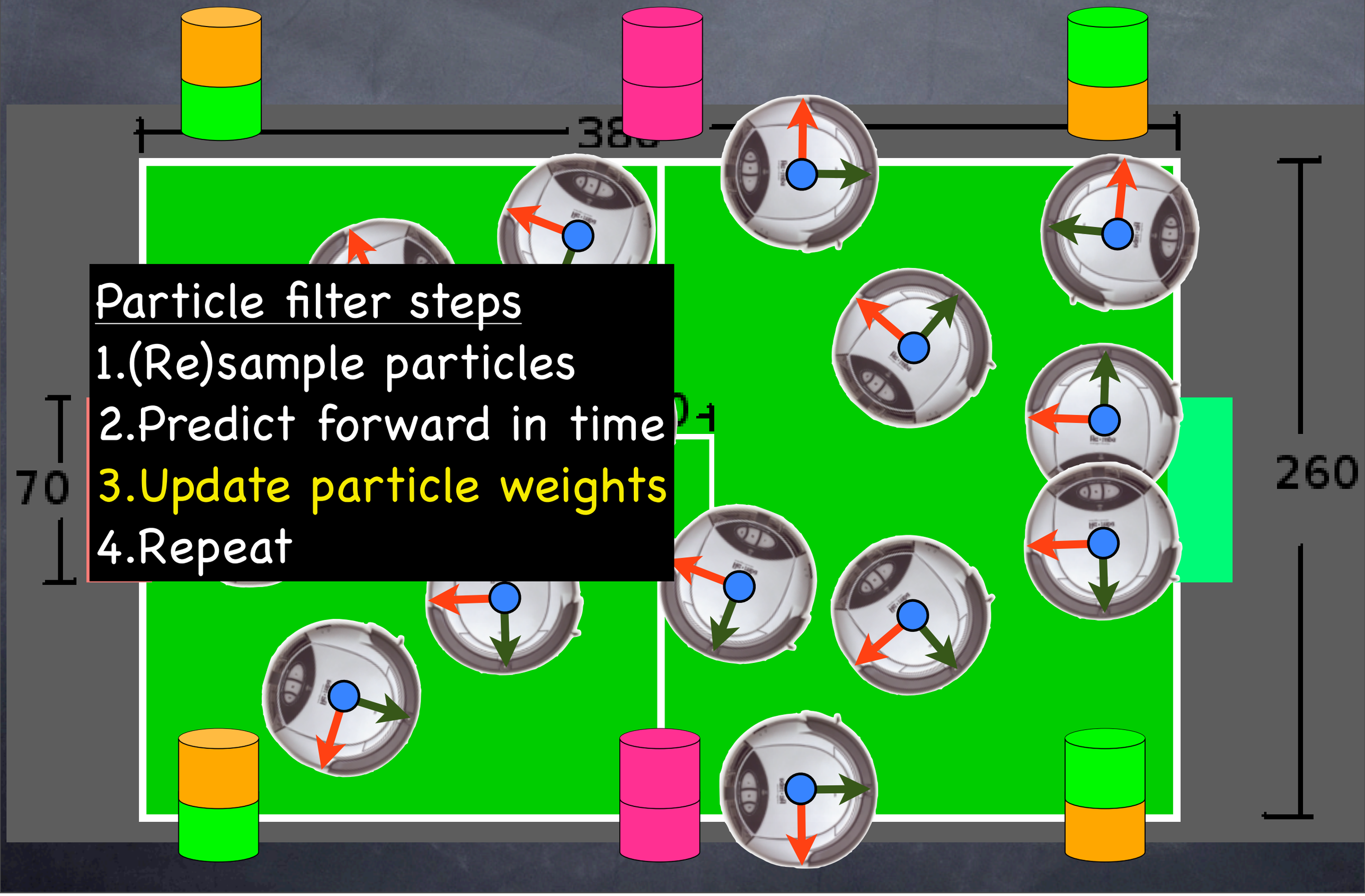
2.Predict forward in time

(skip this for now; assume no movement at t=0)

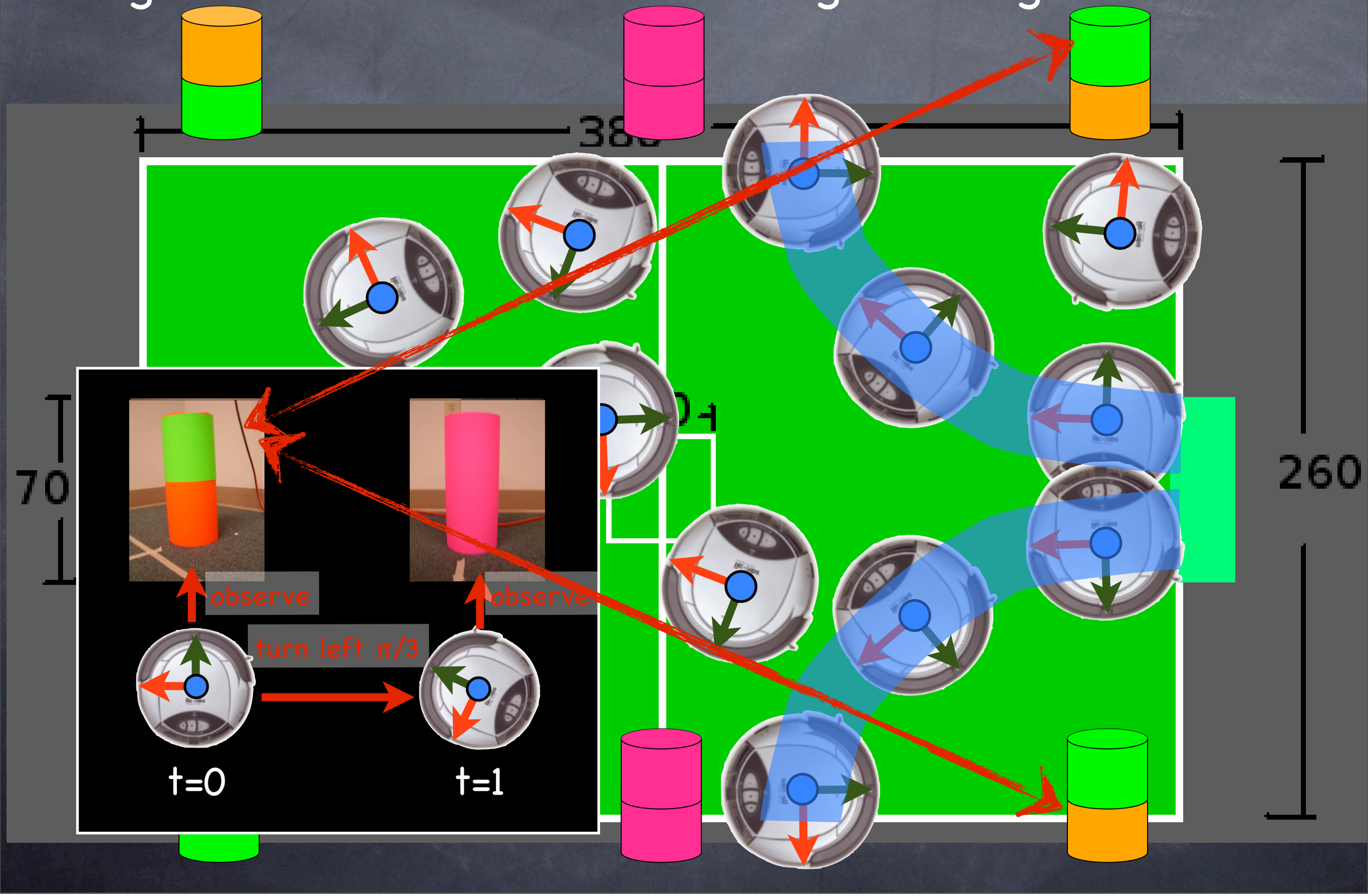
3.Update particle weights

4.Repeat

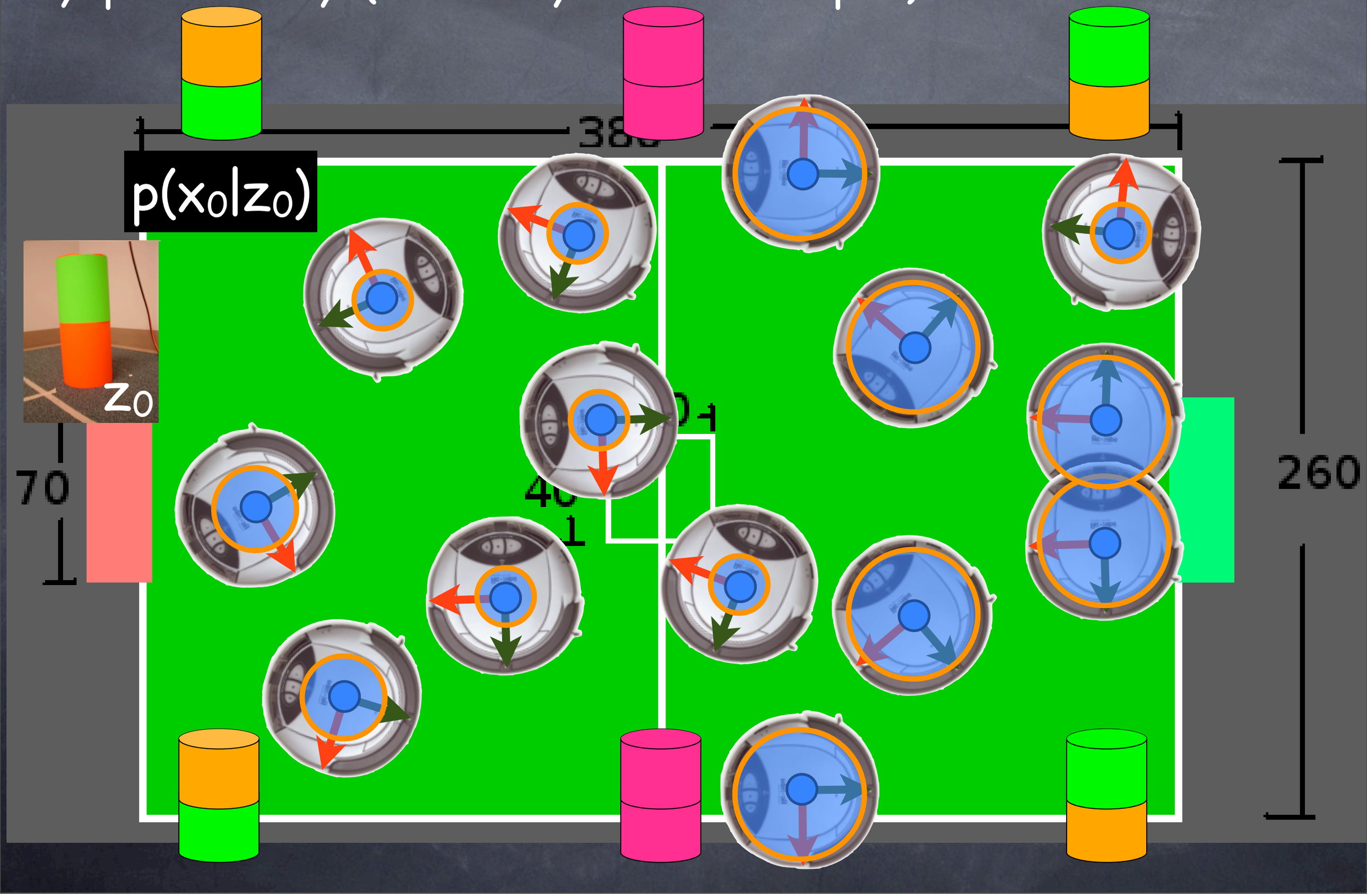
poses sampled from a uniform distribution across pitch



remember from previous example: poses seeing green/ orange landmark should receive higher weights



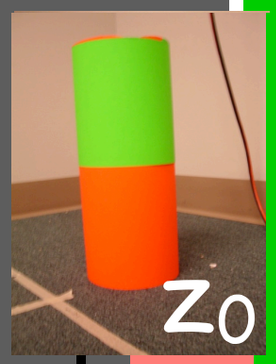
note: belief for particle filter are pose samples weighted by probability (noted by size of ellipse)



note: belief for particle filter are pose samples weighted by probability (noted by size of ellipse)

$p(x_0|z_0)$

How to evaluate the likelihood of each pose sample?



z_0

70

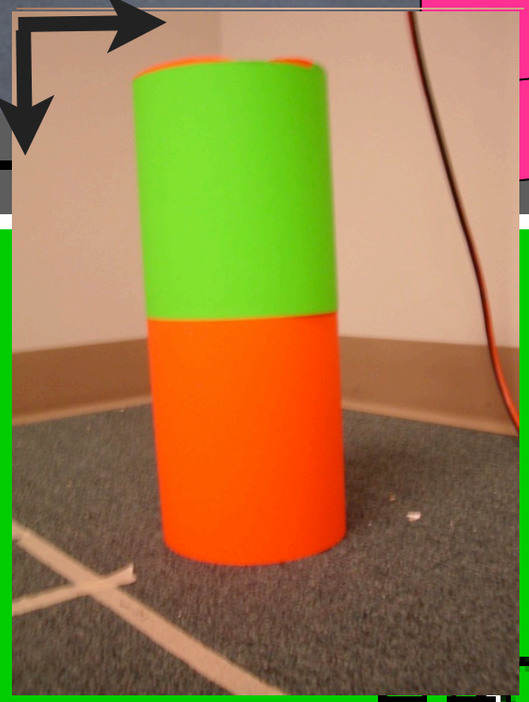
40

1

260

consider again the robot's coordinate spaces

image space: what our camera sees



view space: image with coords in center of view

pose space: represents 3D space centered at robot

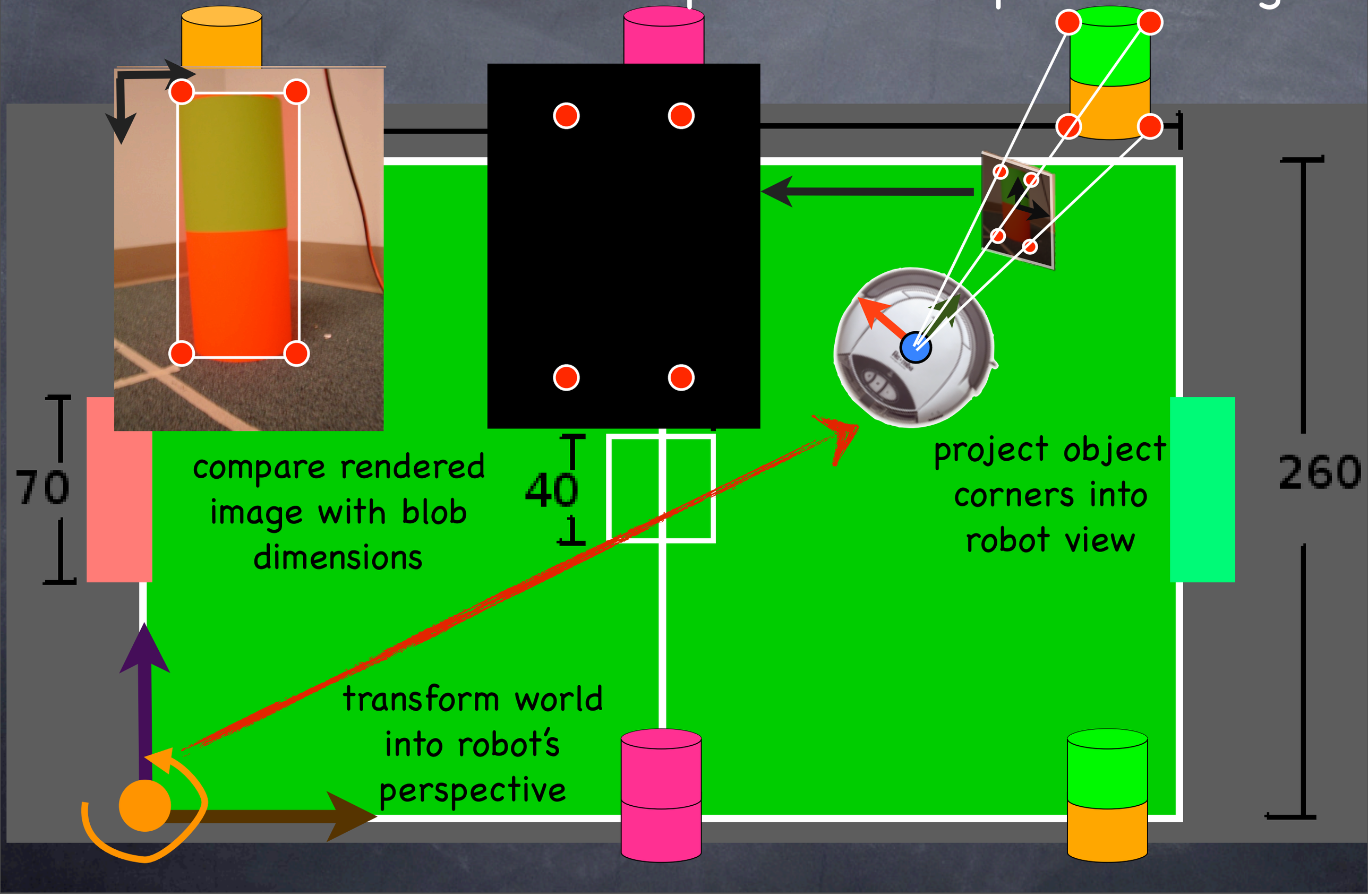
70

40

260

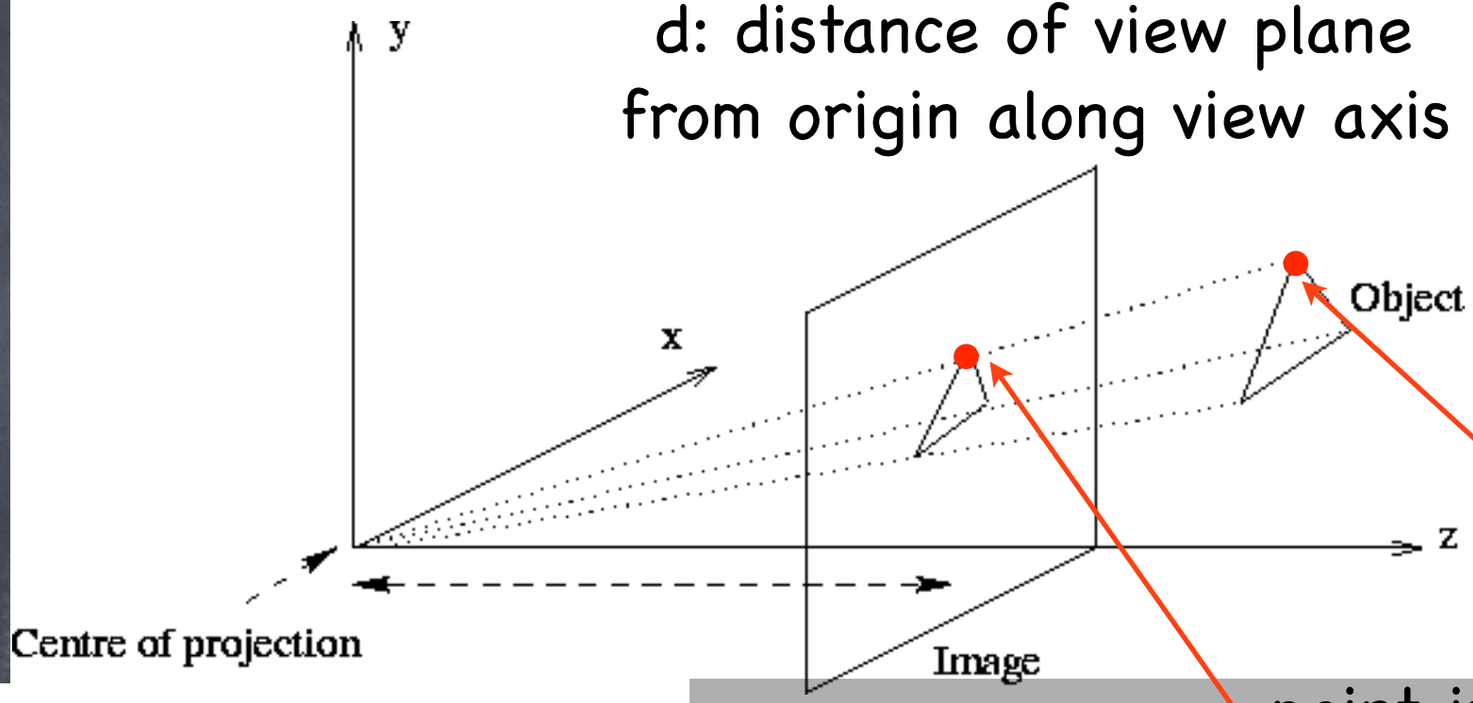
field space: 3D world space fixed to field

Option 1: To evaluate a pose, we could graphically render what should be seen at the pose and compare to image



Basic Perspective Projection

d: distance of view plane
from origin along view axis



assumes camera focal point at origin looking along z-axis

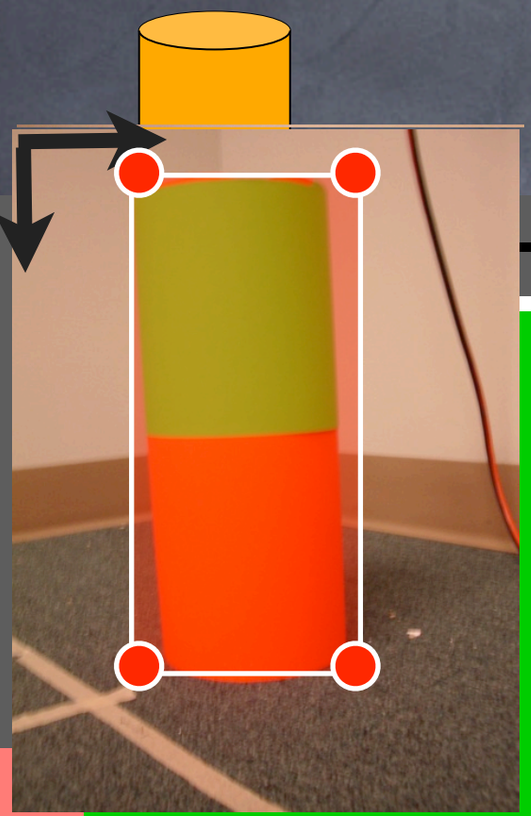
points must be transformed from world to camera space

point in image point in camera

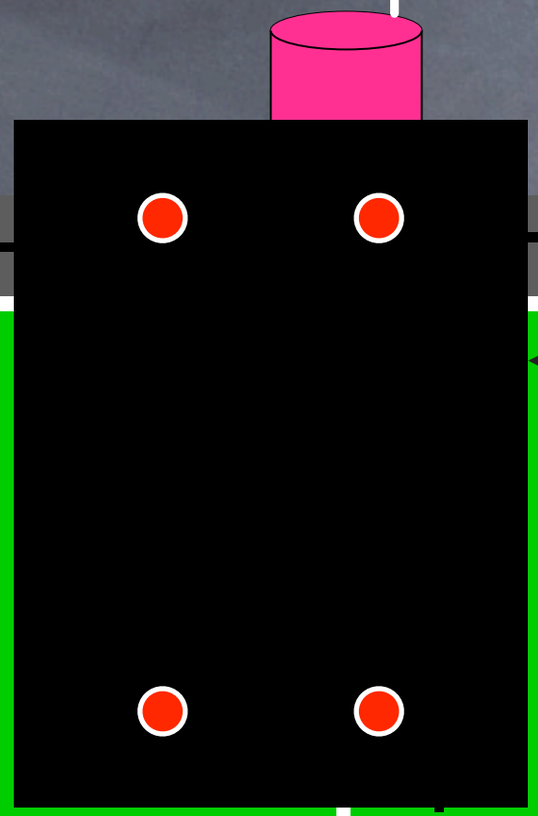
$$\begin{bmatrix} x_s \\ y_s \\ d \end{bmatrix} \equiv \begin{bmatrix} x_v \\ y_v \\ z_v \\ z_v/d \end{bmatrix} \qquad \mathbf{P}_s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \mathbf{P}_v$$

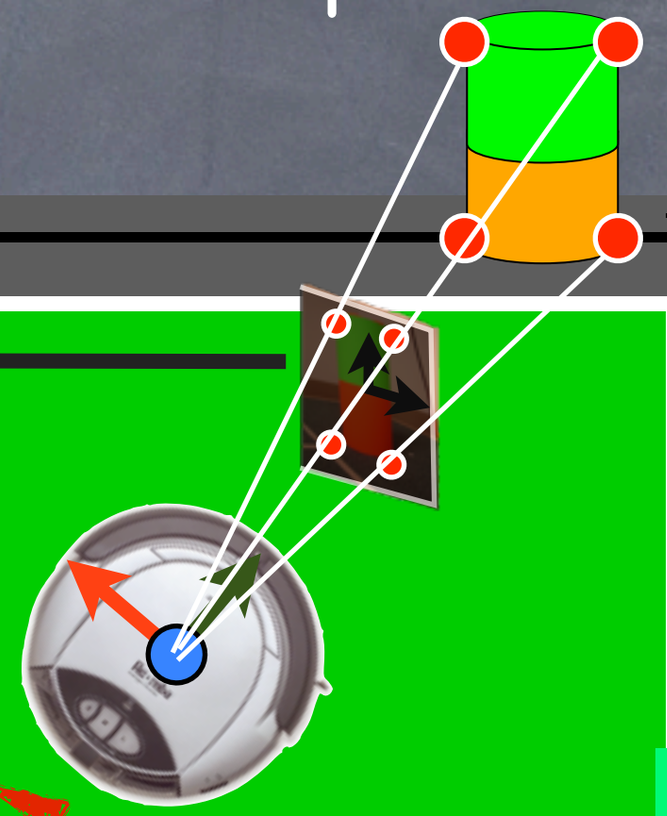
perspective divide projection matrix

Option 1: To evaluate a pose, we could graphically render what should be seen at the pose and compare to image



compare rendered image with blob





project object corners into

70

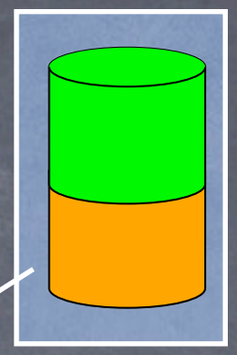
40

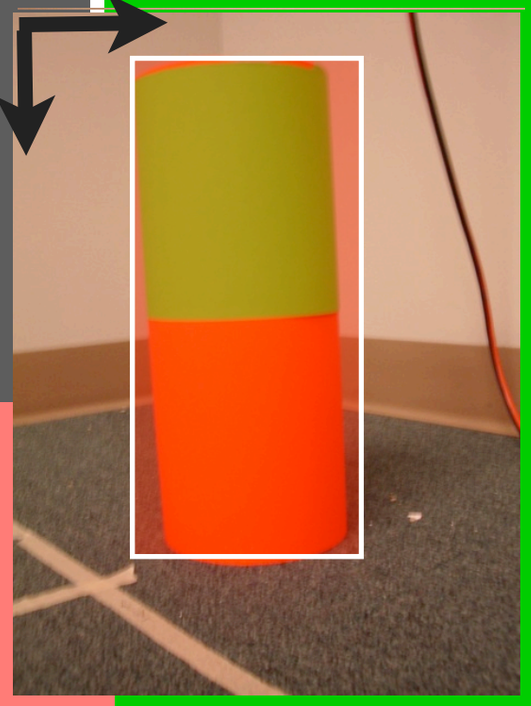
260

General perspective requires accurate camera model

intrinsic calibration parameters:
aspect ratio, field of view, barrel distortion

Option 2: To evaluate a pose, we can compare the range and bearing of objects in robot coordinates



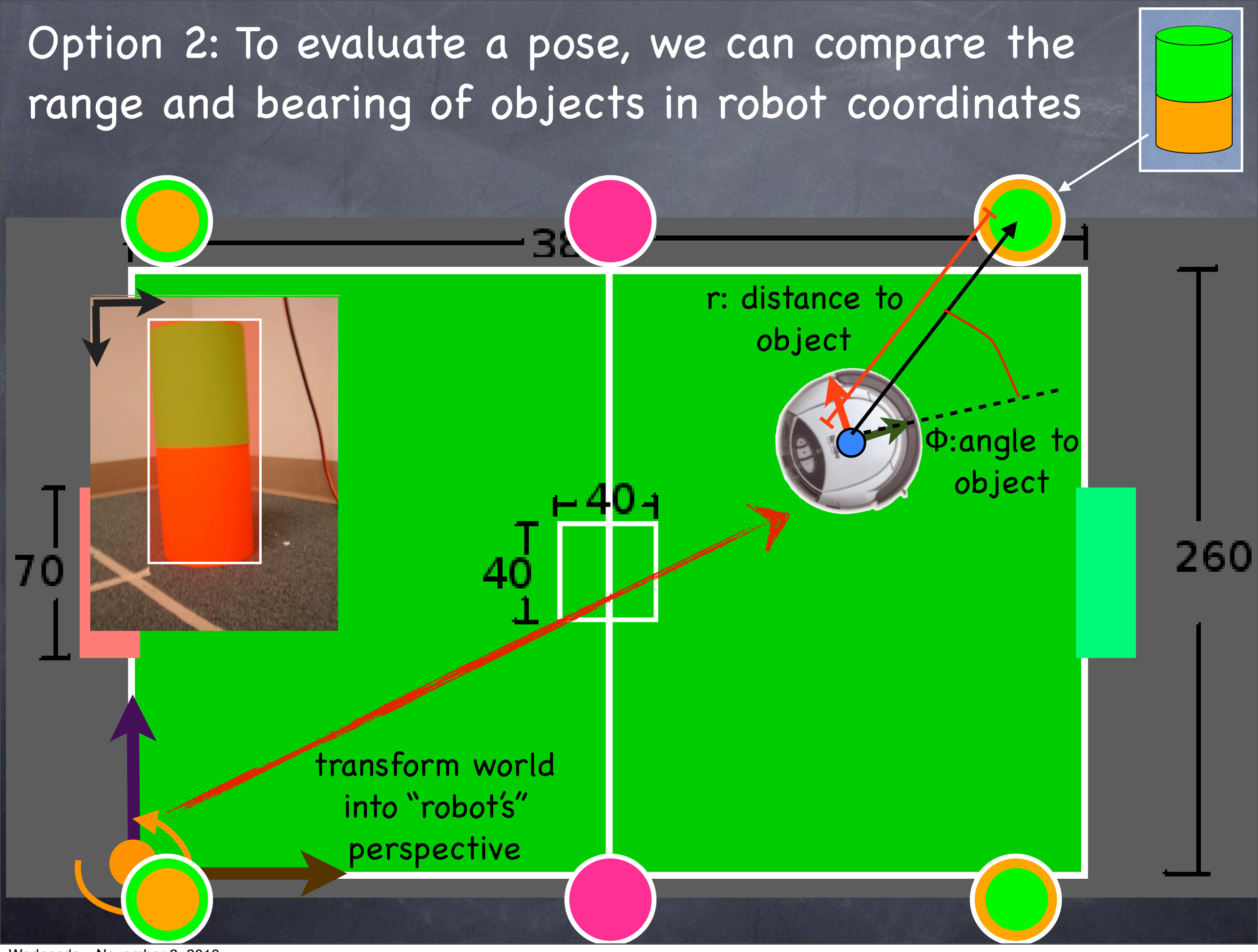


r: distance to object

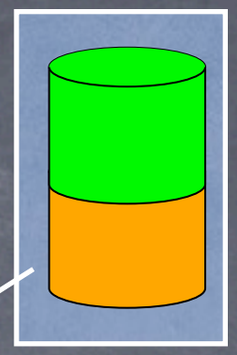
Φ:angle to object

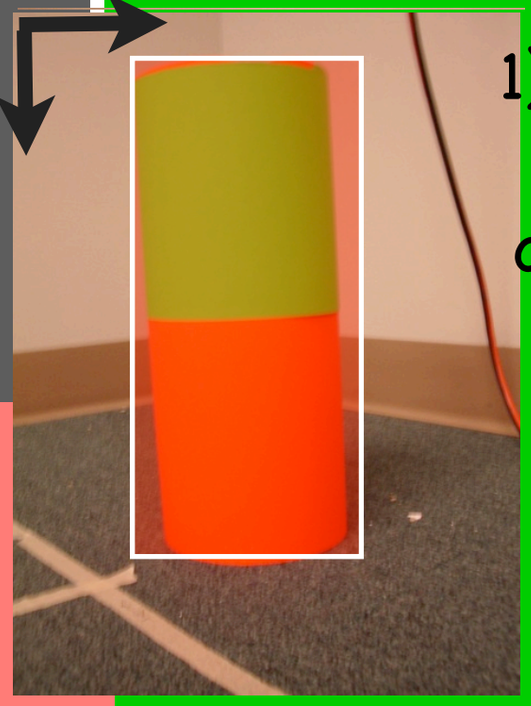
transform world into "robot's" perspective

Option 2: To evaluate a pose, we can compare the range and bearing of objects in robot coordinates



Option 2: To evaluate a pose, we can compare the range and bearing of objects in robot coordinates





1) Estimate (r′,Φ′) from blob of observed object

r: distance to object

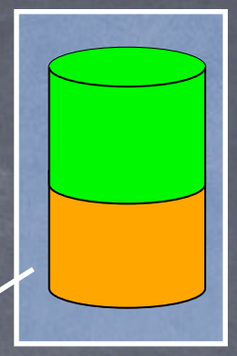
Φ:angle to object

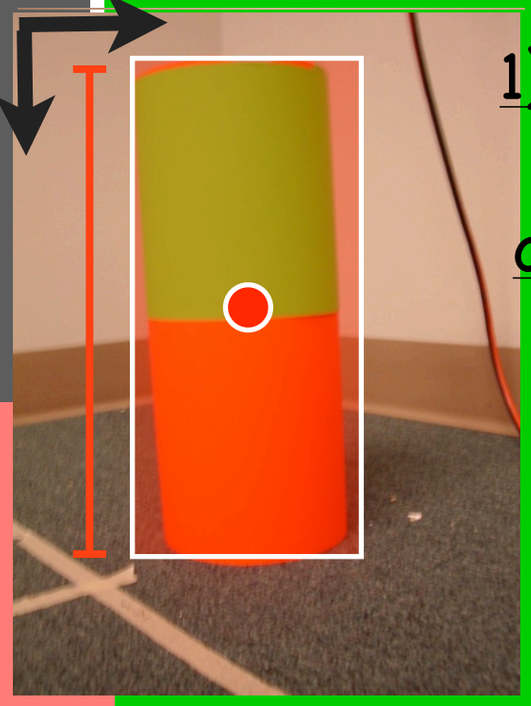
2) calculate (r_x,Φ_x) expected by given a hypothesized pose x

3) compute weight from match between (r_x,Φ_x) and (r′,Φ′)

transform world into "robot's" perspective

Option 2: To evaluate a pose, we can compare the range and bearing of objects in robot coordinates





1) Estimate (r′,Φ′) from blob of observed object

r: distance to object

Φ:angle to object

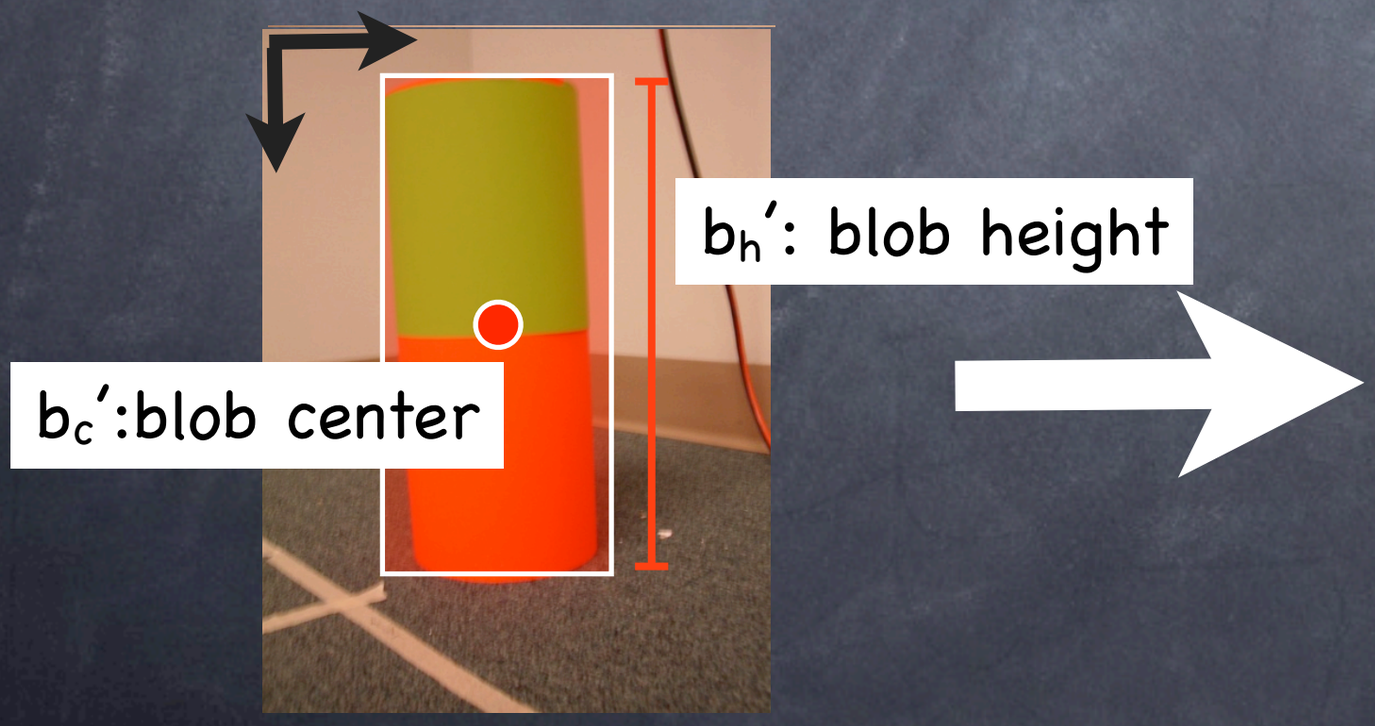
2) calculate (r_x,Φ_x) expected by given a hypothesized pose x

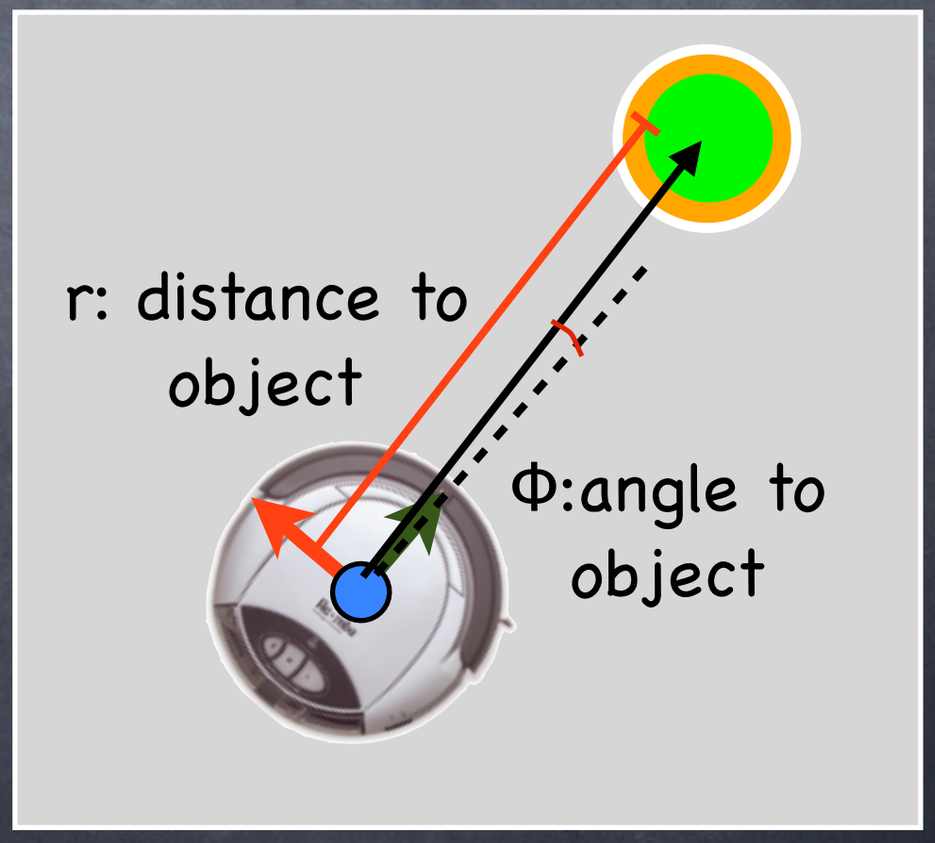
transform world into "robot's" perspective

3) compute weight from match between (r_x,Φ_x) and (r′,Φ′)

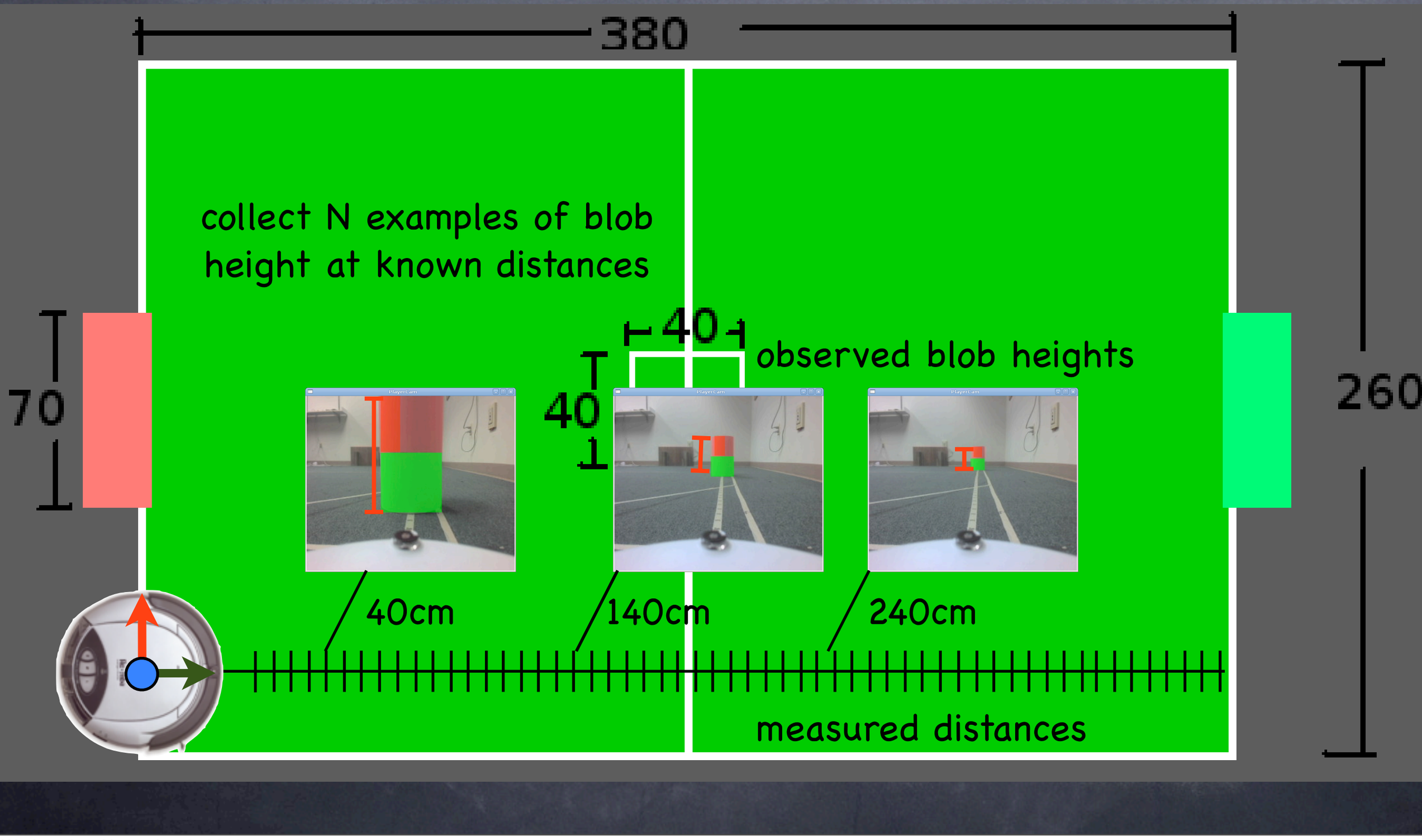
Estimating range and distance from blob

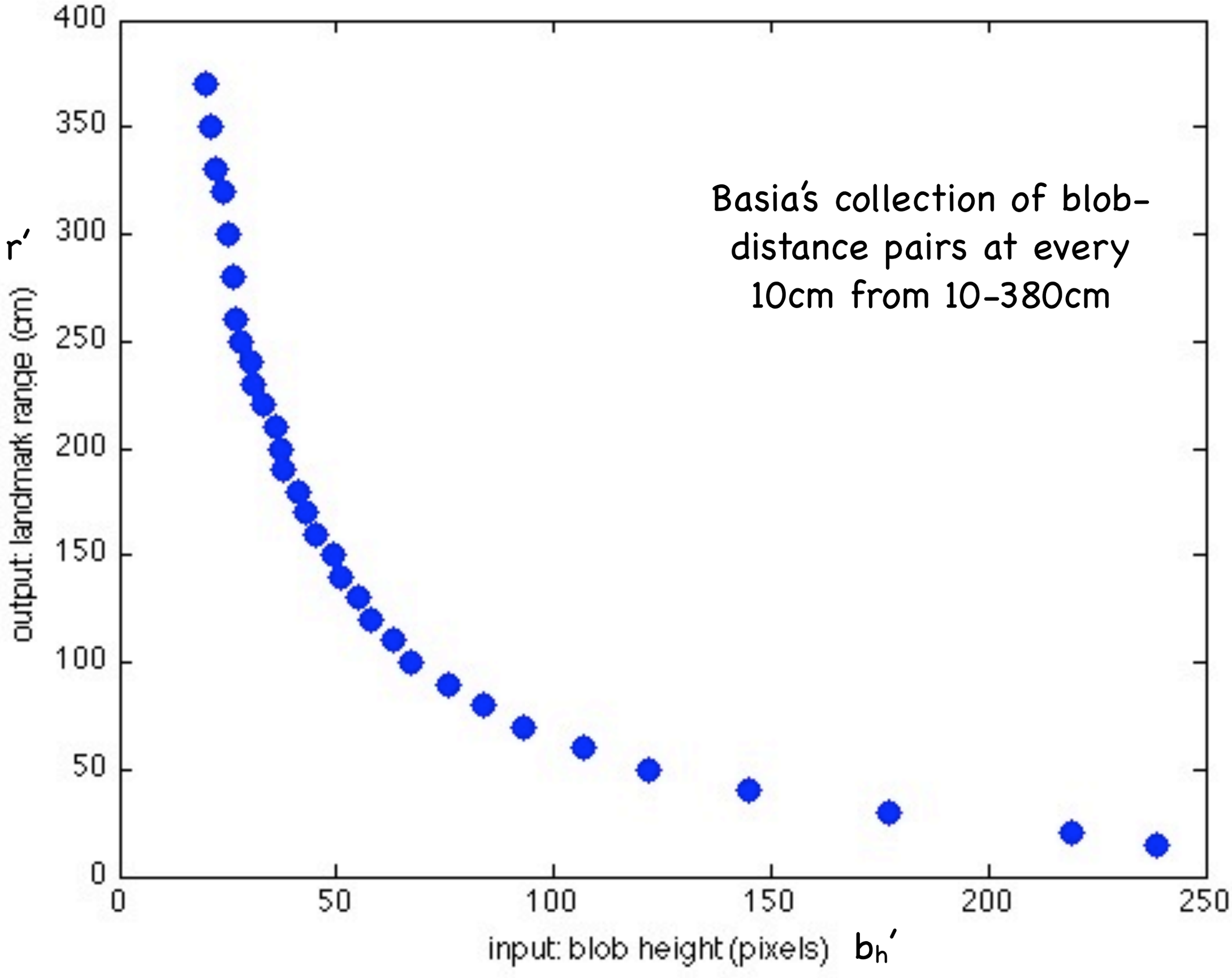
- Construct two functions:
  - Predict object distance from blob height
  - Predict object angle form blob center

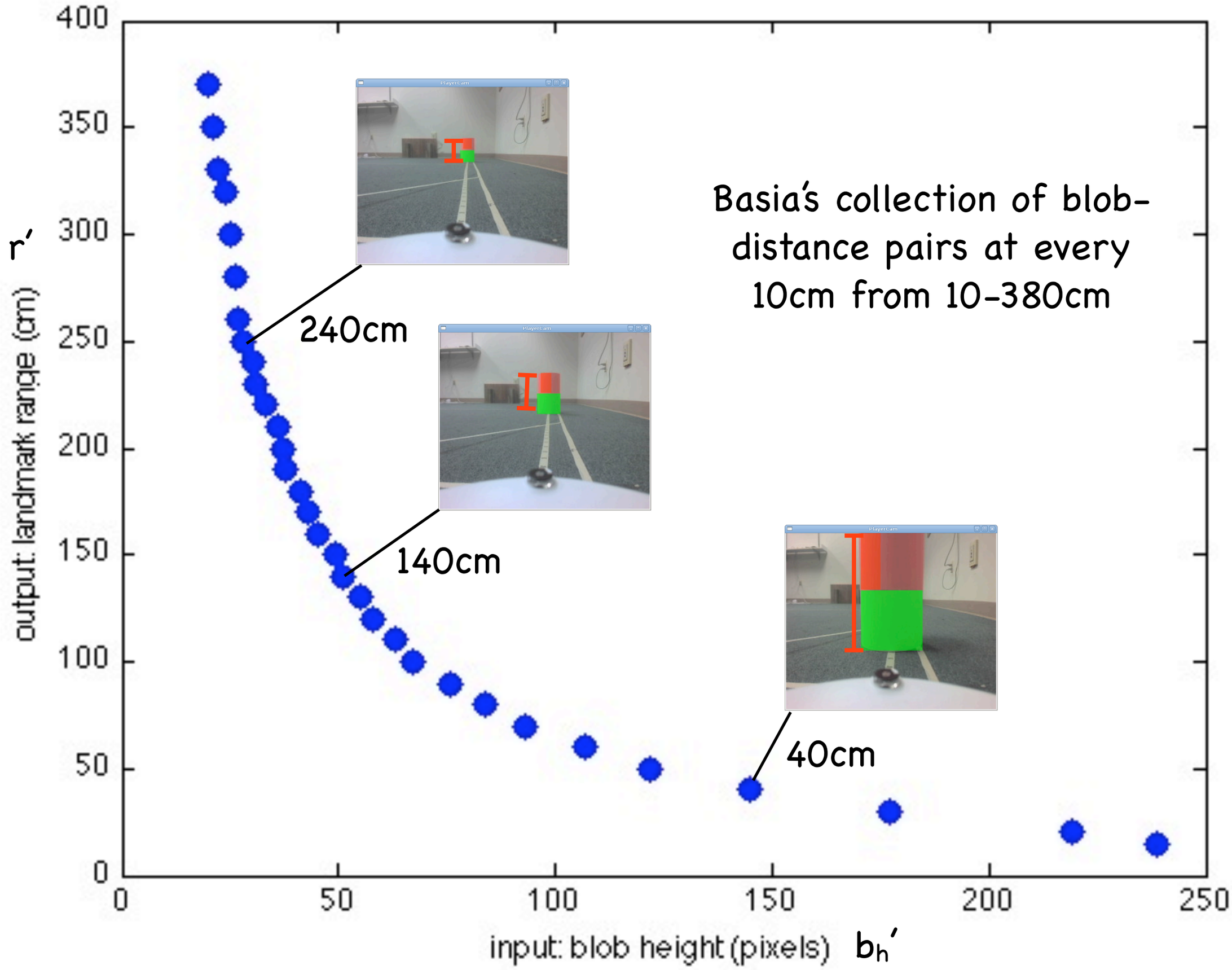


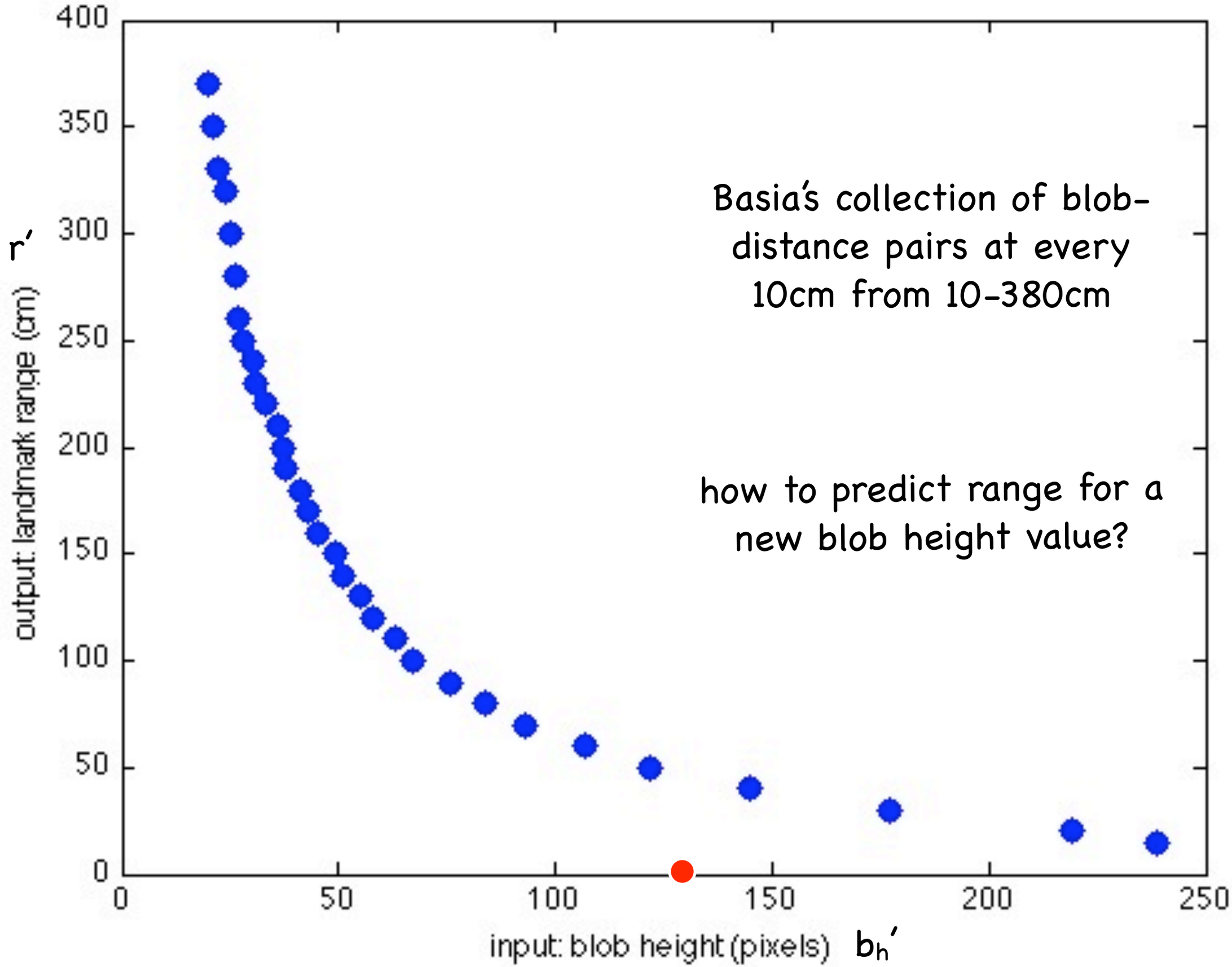


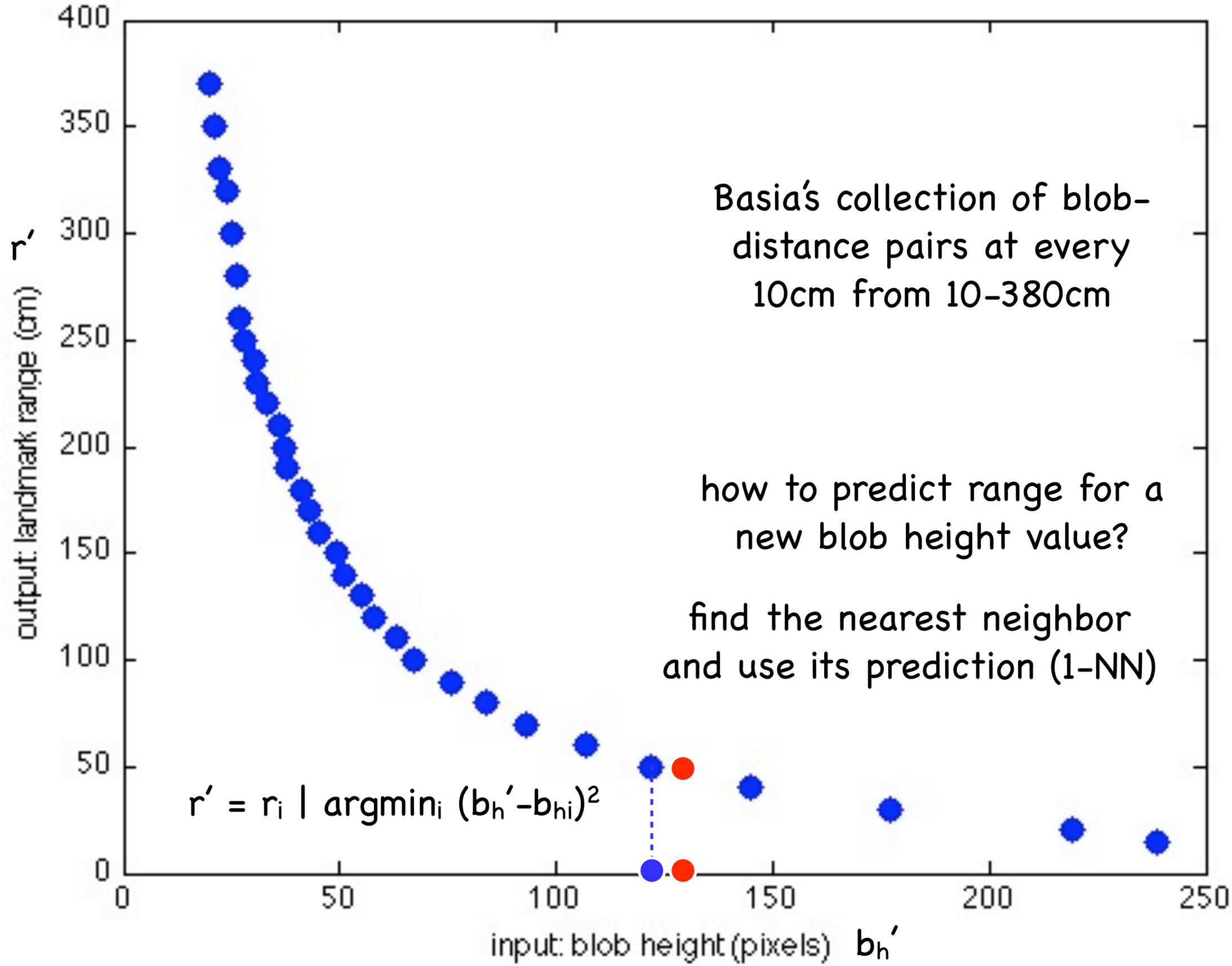
Estimating distance from examples

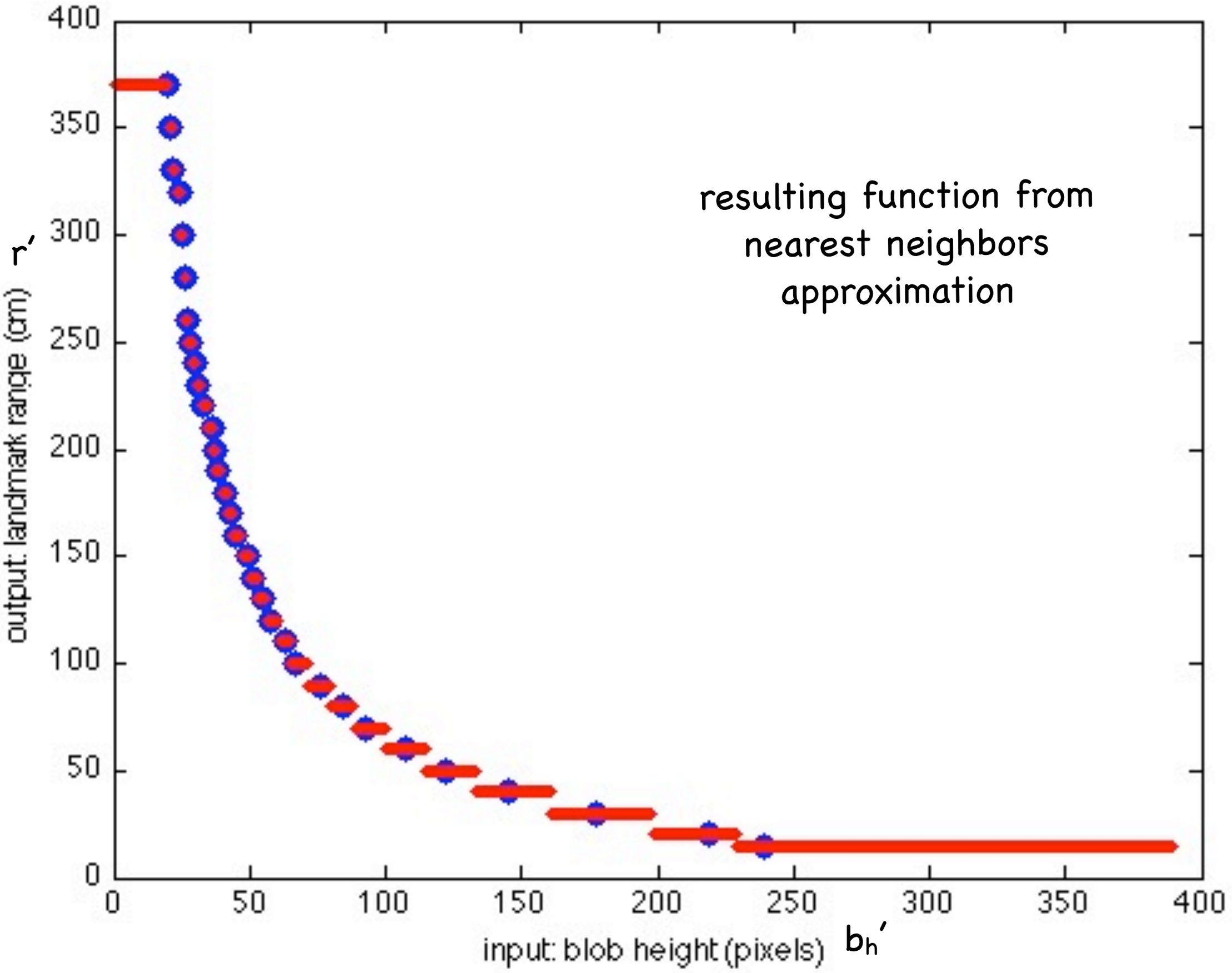


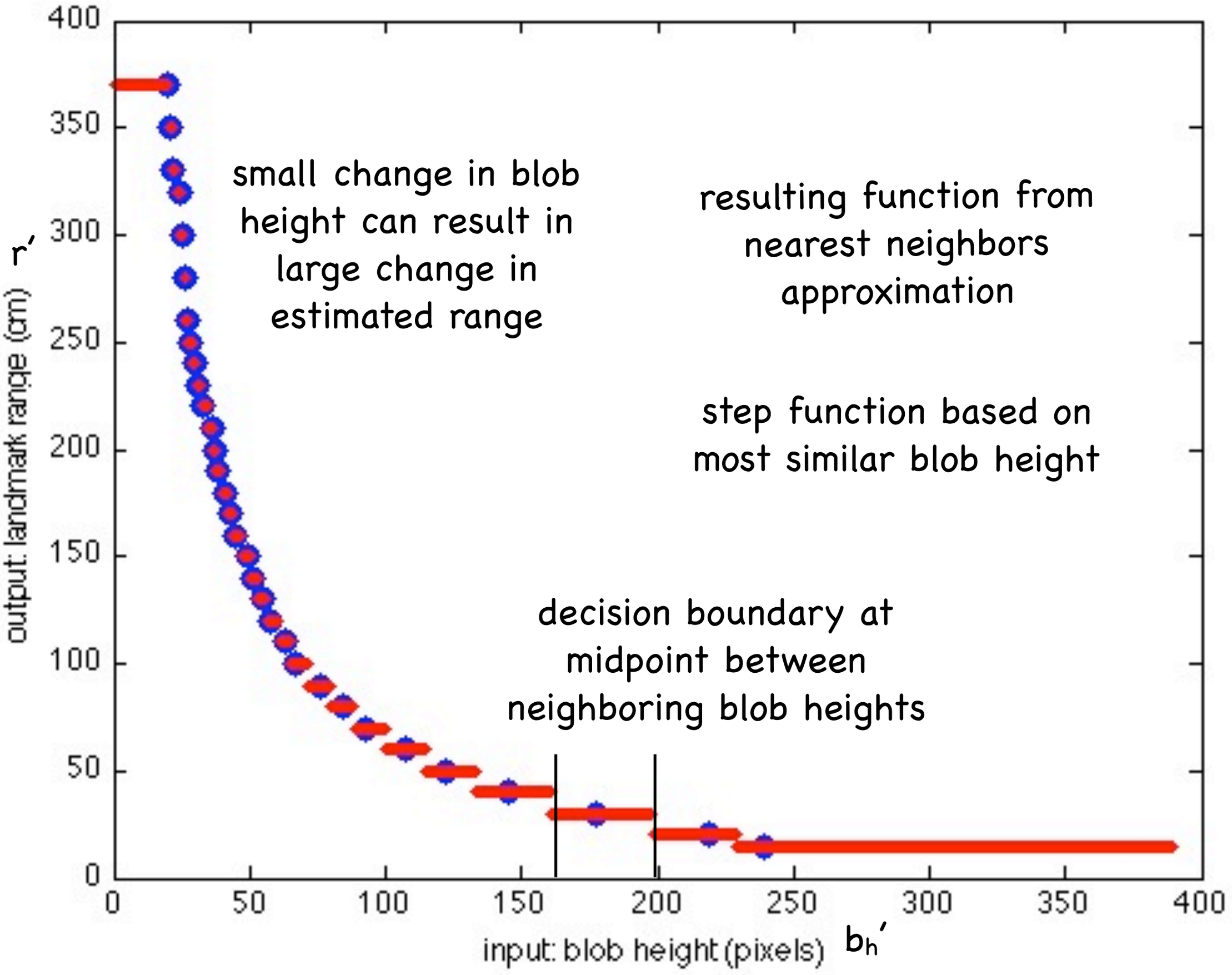

400
350
300
250
200
150
100
50
0
0
50
100
150
200
250
output: landmark range (cm)
r′
input: blob height (pixels)
b_h'
Basia's collection of blob-distance pairs at every 10cm from 10-380cm

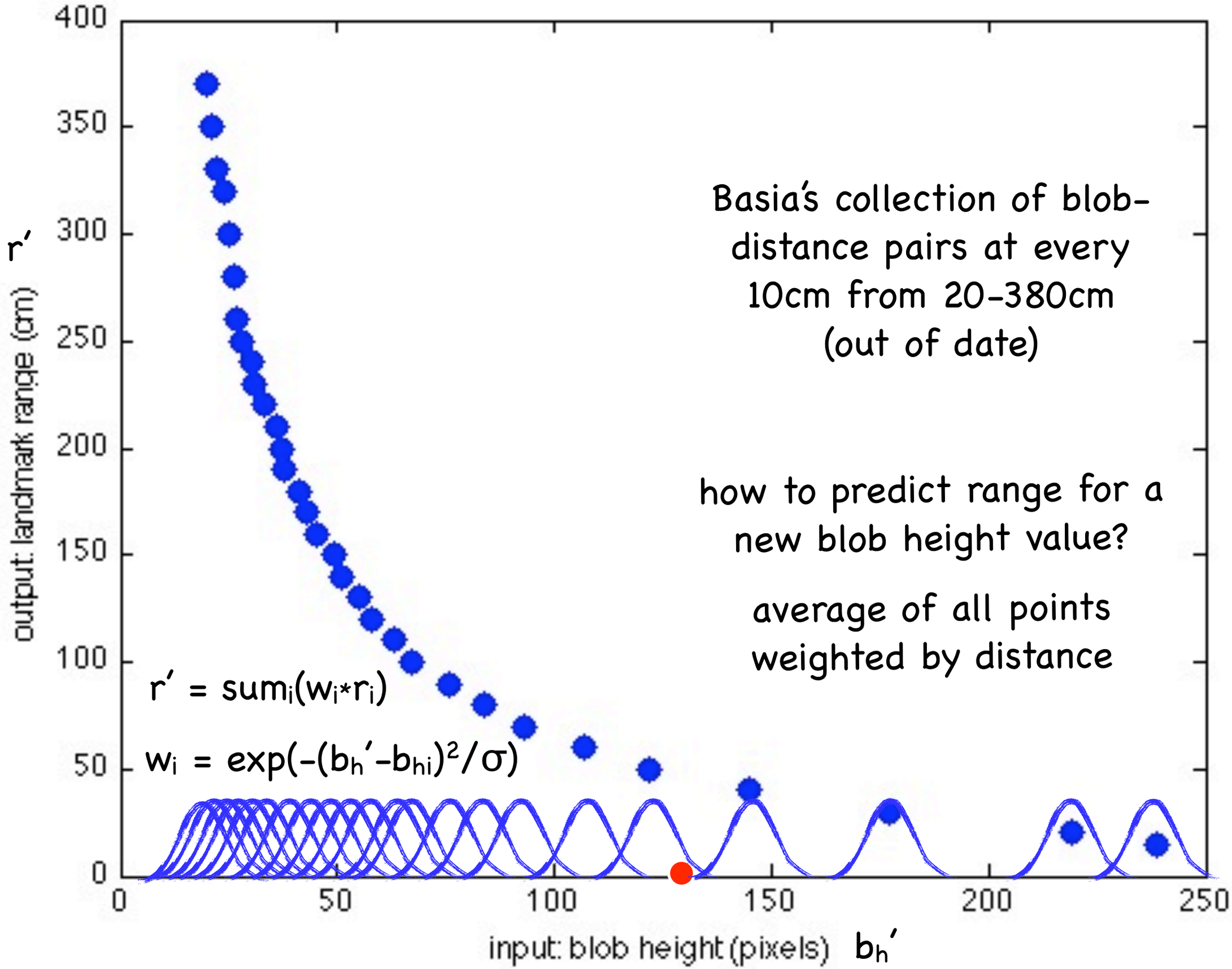

400
350
300
250
200
150
100
50
0
r′
output: landmark range (cm)
0
50
100
150
200
250
input: blob height (pixels) b_h'
240cm
140cm
40cm
Basia's collection of blob-distance pairs at every 10cm from 10-380cm


400
350
300
250
200
150
100
50
0
0
50
100
150
200
250
output: landmark range (cm)
r′
input: blob height (pixels)
b_h'
Basia's collection of blob-distance pairs at every 10cm from 10-380cm
how to predict range for a new blob height value?


400
350
300
250
200
150
100
50
0
0
50
100
150
200
250
r′
output: landmark range (cm)
input: blob height (pixels) b_h'
Basia's collection of blob-distance pairs at every 10cm from 10-380cm
how to predict range for a new blob height value?
find the nearest neighbor and use its prediction (1-NN)
$r' = r_i \mid \text{argmin}_i\ (b_h'-b_{hi})^2$


resulting function from
nearest neighbors
approximation
output: landmark range (cm) r'
input: blob height (pixels) b_h'
0
50
100
150
200
250
300
350
400


small change in blob height can result in large change in estimated range
resulting function from nearest neighbors approximation
step function based on most similar blob height
decision boundary at midpoint between neighboring blob heights
output: landmark range (cm) r'
input: blob height (pixels) b_h'
0
50
100
150
200
250
300
350
400

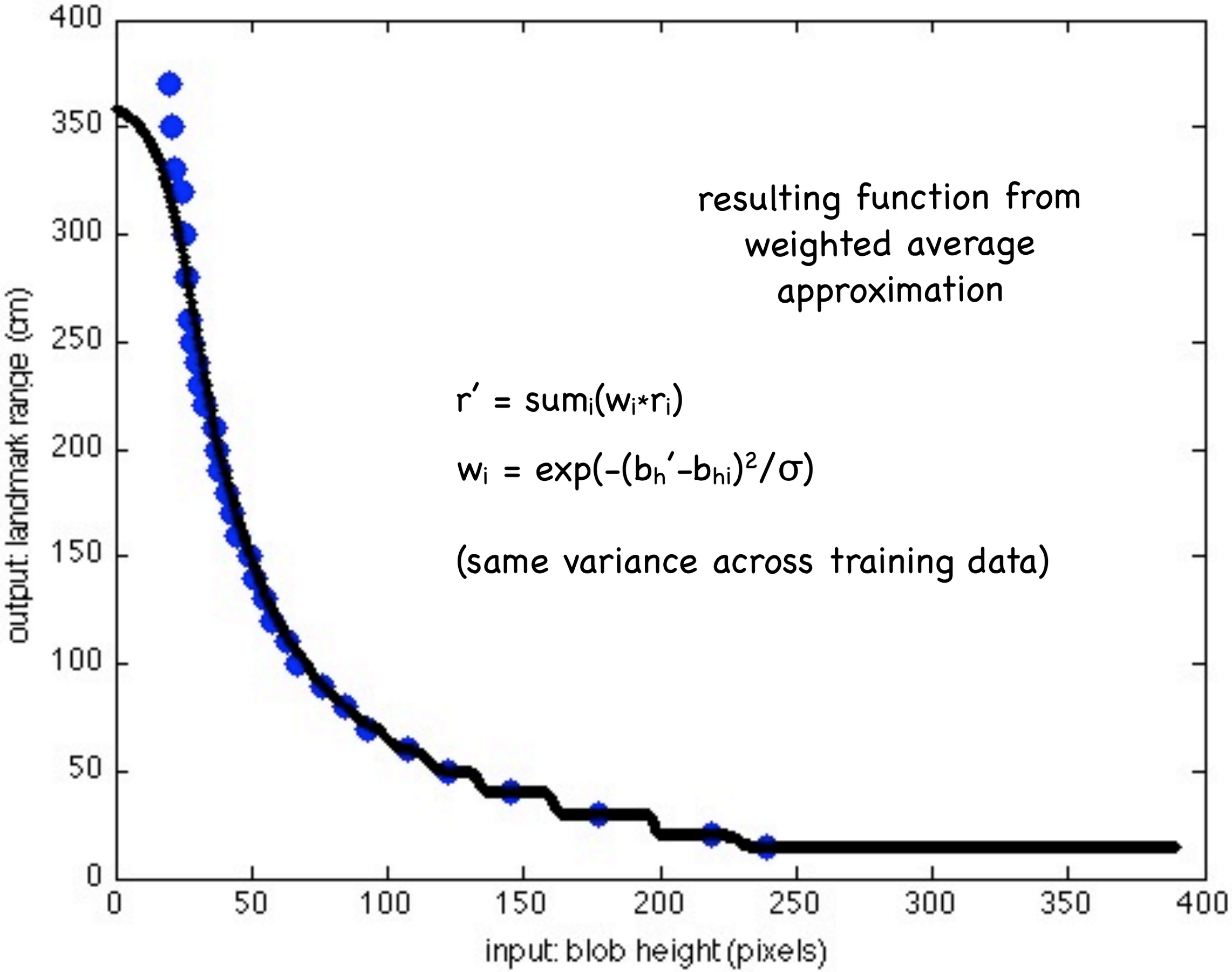

400
350
300
250
200
150
100
50
0
0
50
100
150
200
250
output: landmark range (cm)
r′
input: blob height (pixels)
b_h'
Basia's collection of blob-distance pairs at every 10cm from 20-380cm (out of date)
how to predict range for a new blob height value?
average of all points weighted by distance
$r' = \mathrm{sum}_i(w_i * r_i)$
$w_i = \exp(-(b_h' - b_{hi})^2/\sigma)$

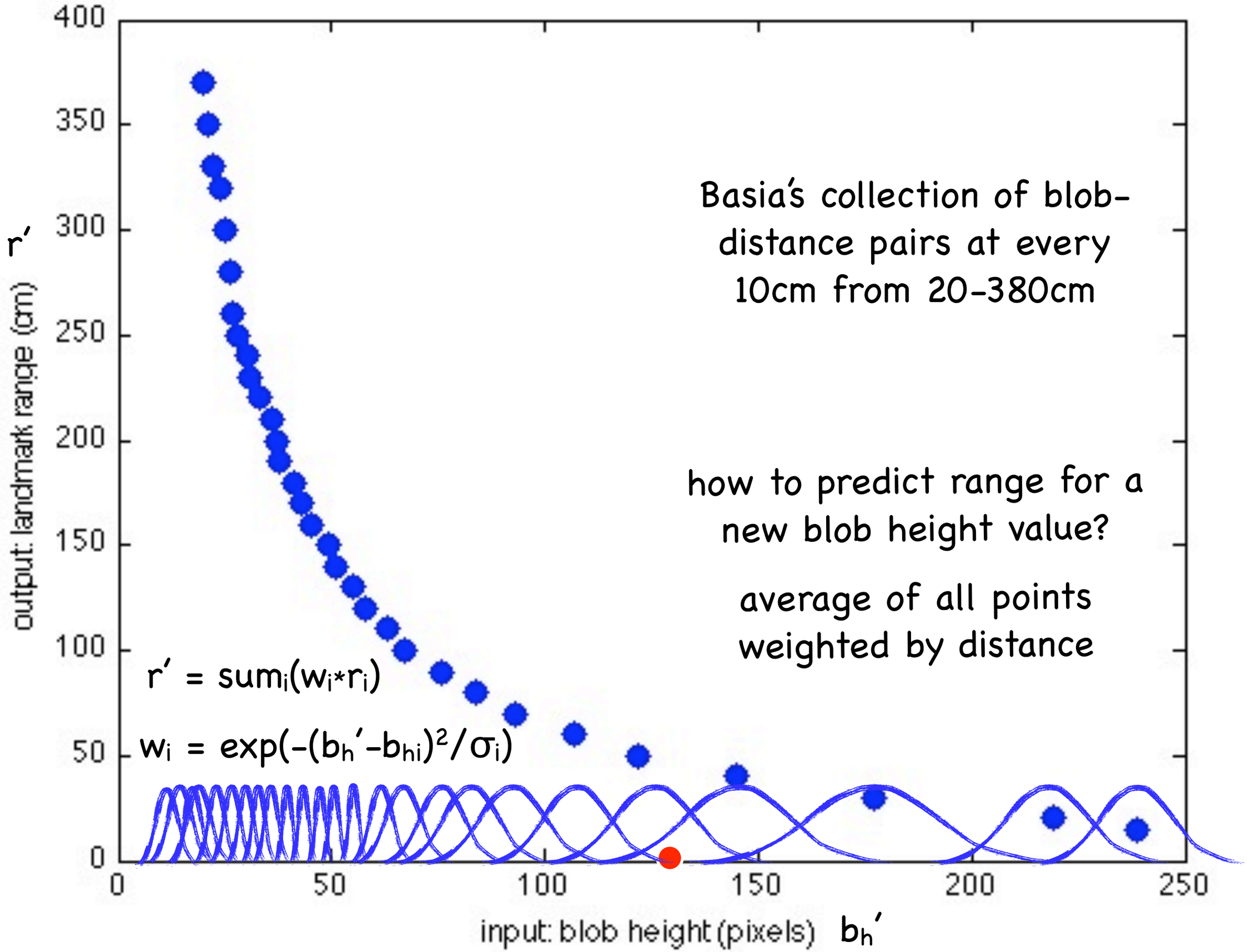
resulting function from
weighted average
approximation

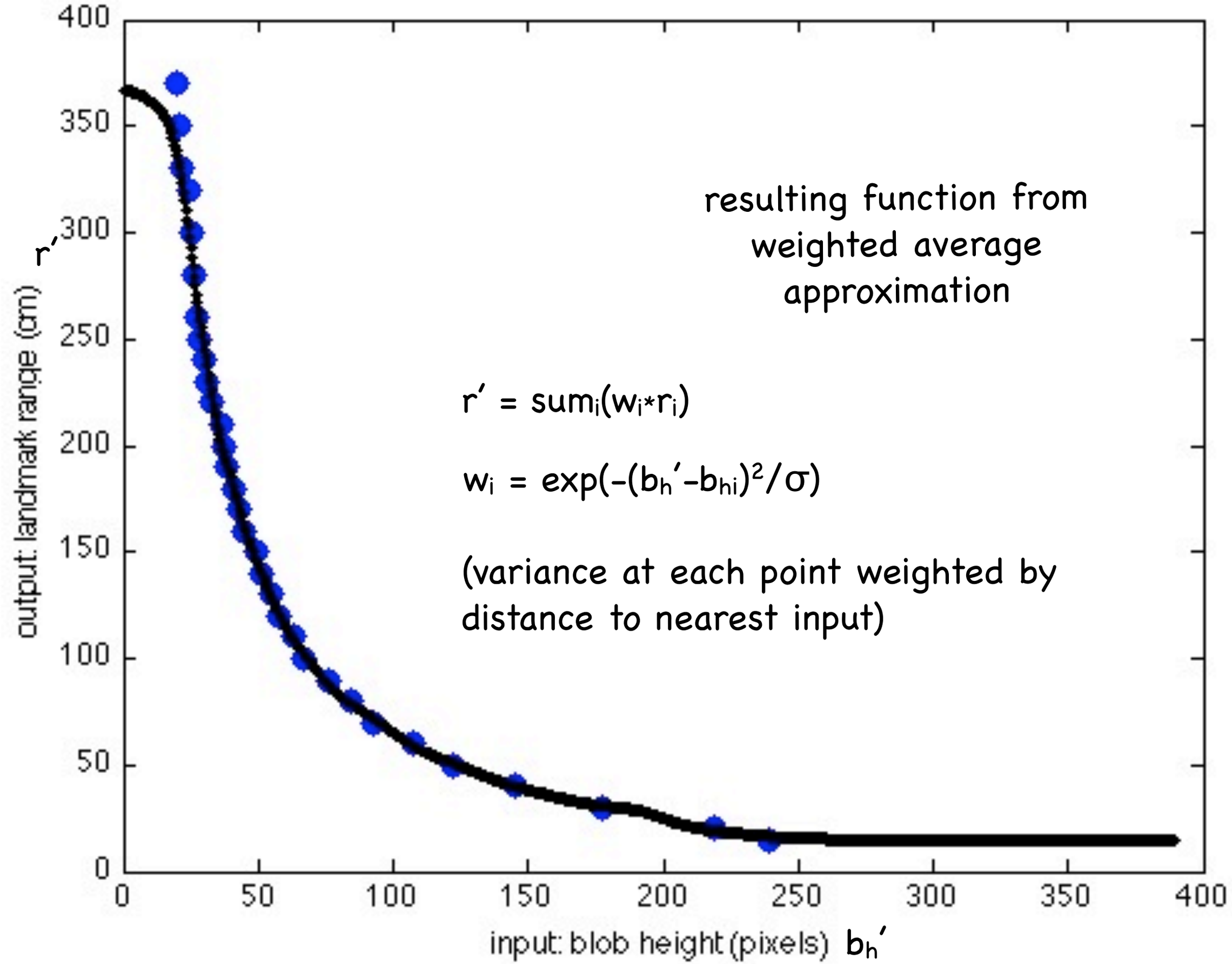
$r' = \text{sum}_i(w_i * r_i)$

$w_i = \exp(-(b_h' - b_{hi})^2/\sigma)$

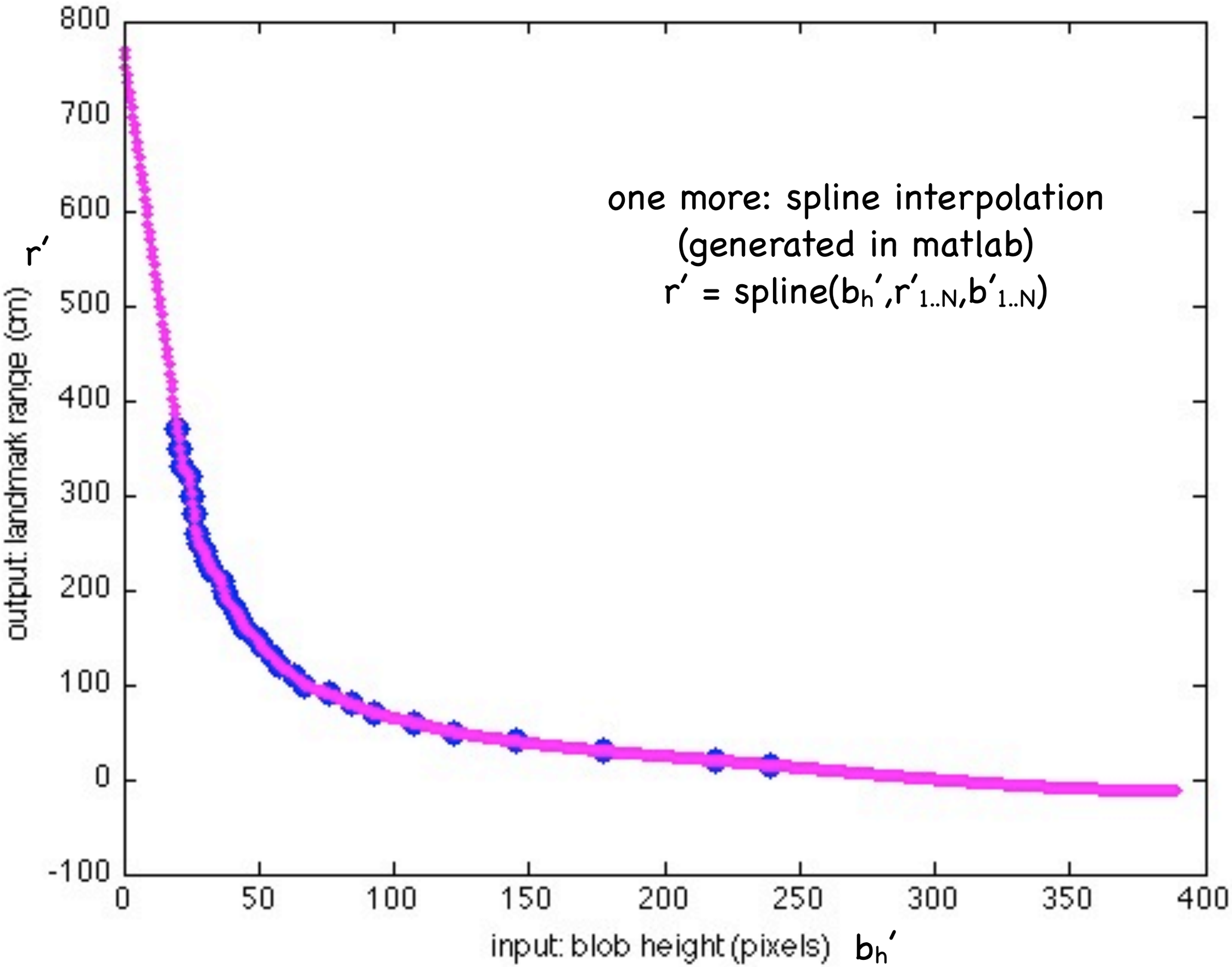
(same variance across training data)




output: landmark range (cm)
r′
400
350
300
250
200
150
100
50
0
0
50
100
150
200
250
input: blob height (pixels)
b_h'
Basia's collection of blob-distance pairs at every 10cm from 20-380cm
how to predict range for a new blob height value?
average of all points weighted by distance
$r' = sum_i(w_i * r_i)$
$w_i = exp(-(b_h' - b_{hi})^2/\sigma_i)$

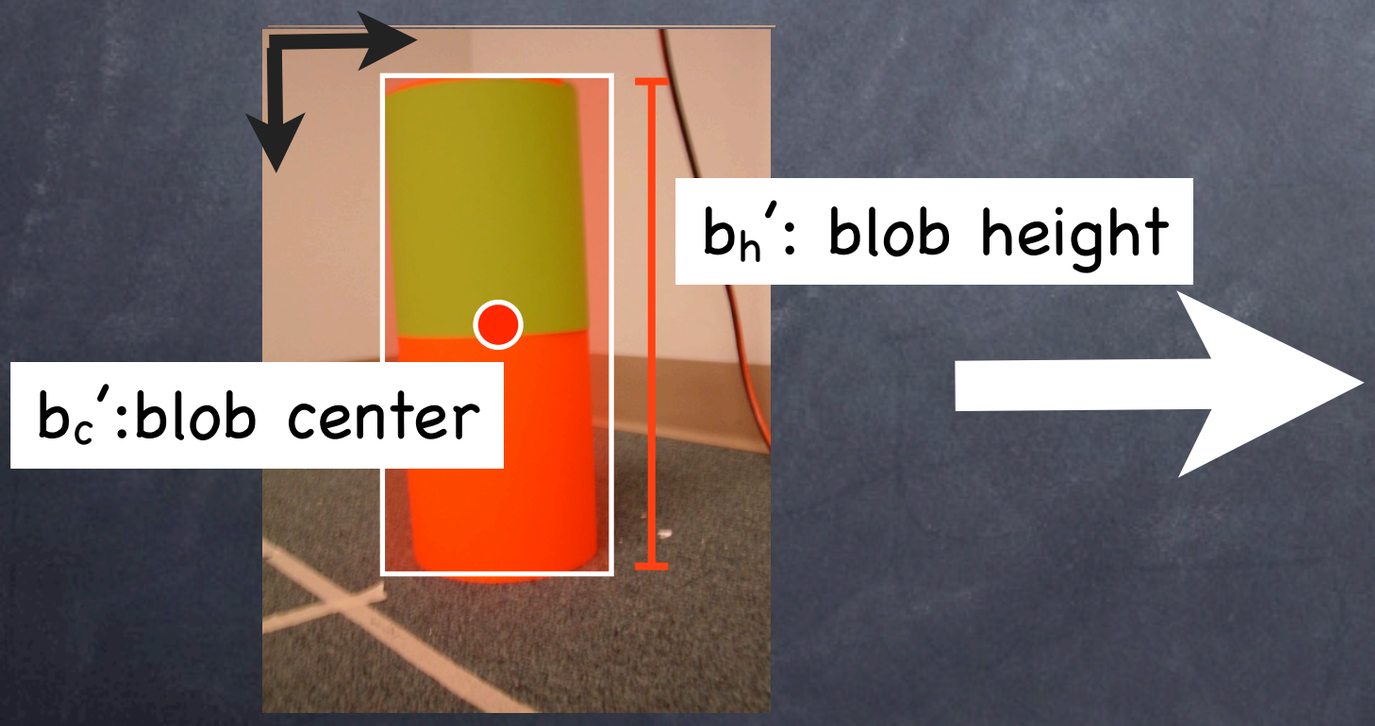

400
350
300
250
200
150
100
50
0
0
50
100
150
200
250
300
350
400
output: landmark range (cm)
r′
input: blob height (pixels) bh′
resulting function from weighted average approximation
r′ = sumi(wi*ri)
wi = exp(-(bh′-bhi)²/σ)
(variance at each point weighted by distance to nearest input)

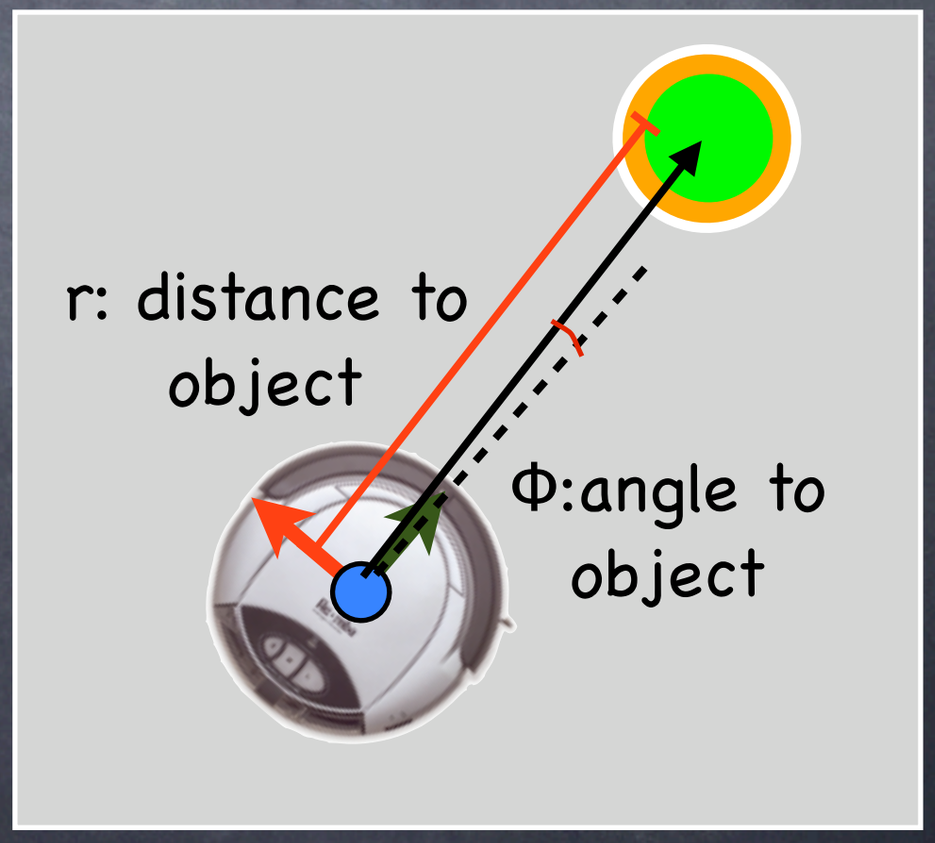
one more: spline interpolation
(generated in matlab)
$r' = \text{spline}(b_h', r'_{1..N}, b'_{1..N})$



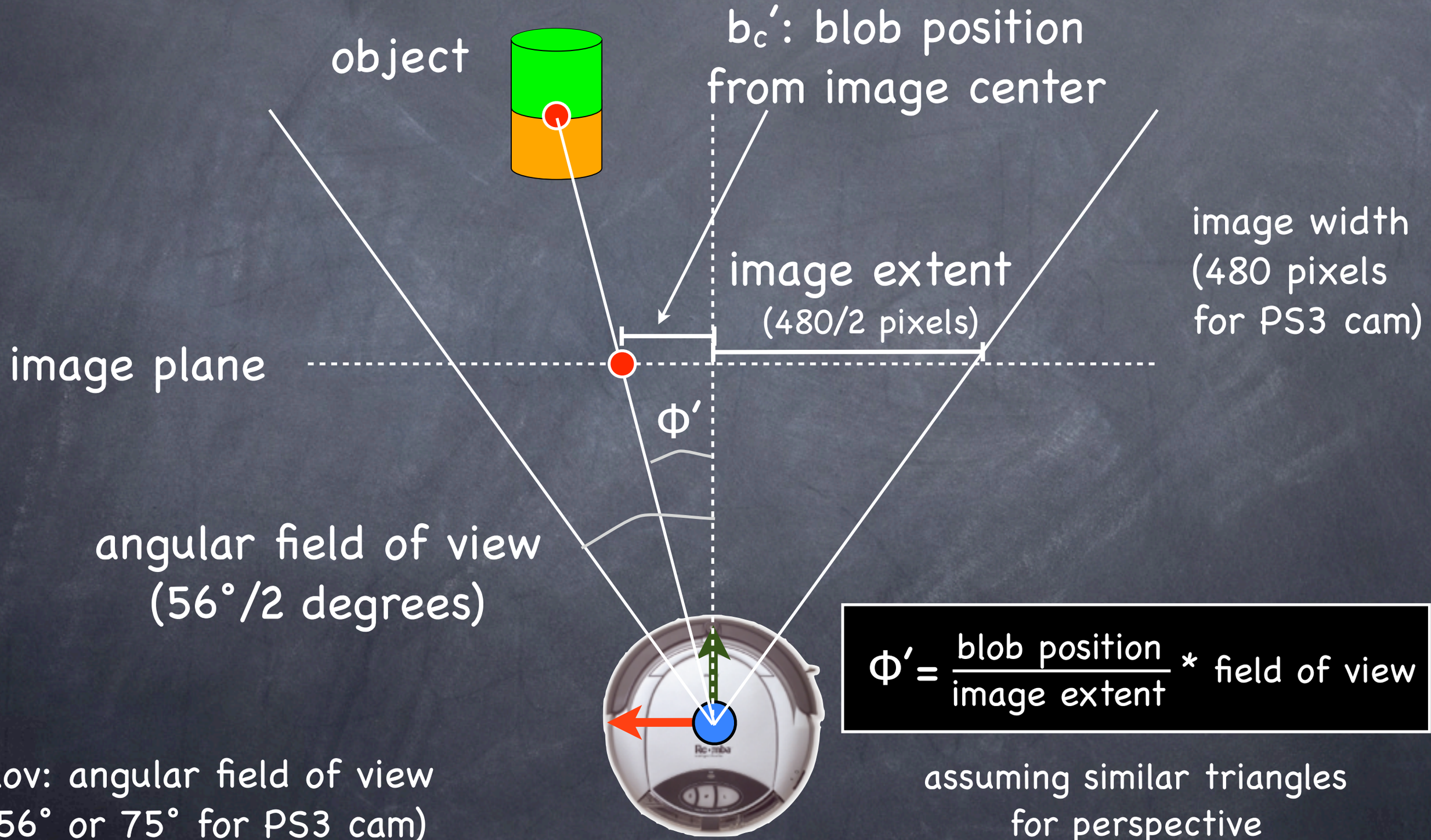
Estimating range and distance from blob

- Construct two functions:
  - Predict object distance from blob height
  - Predict object angle form blob center



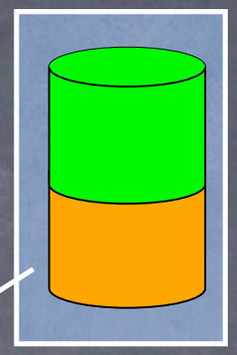


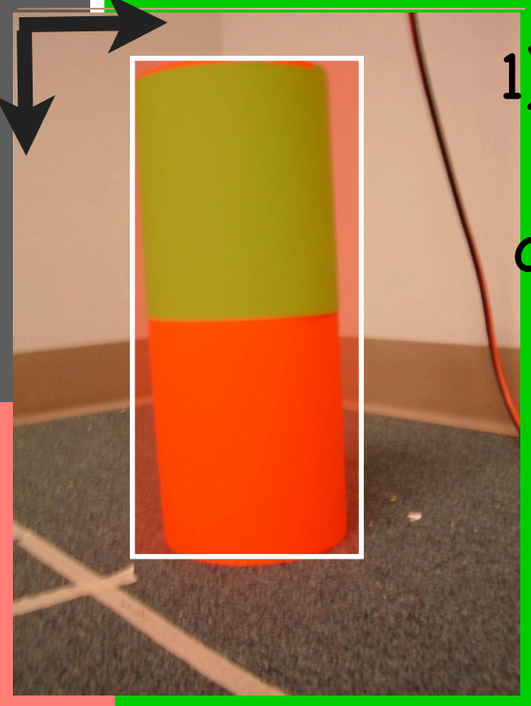
Approximating bearing



$$\Phi' = \frac{\text{blob position}}{\text{image extent}} * \text{field of view}$$

Option 2: To evaluate a pose, we can compare the range and bearing of objects in robot coordinates





1) Estimate (r′,Φ′) from blob of observed object

r: distance to object

Φ:angle to object

2) calculate (r_x,Φ_x) expected by given a hypothesized pose x

transform world into "robot's" perspective

3) compute weight from match between (r_x,Φ_x) and (r′,Φ′)

take every landmark and project it into robot pose coordinates

Each landmark has a 2D location

Remember: we can translate and rotate any 2D point into new coordinates

$$\begin{bmatrix} p_x^r \\ p_y^r \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x^w \\ p_y^w \\ 1 \end{bmatrix}$$

p^r : 2D location of landmark center in robot coordinates

p^w : 2D location of landmark center in field coordinates

view from robot coordinates

p_x^r
p_y^r
1

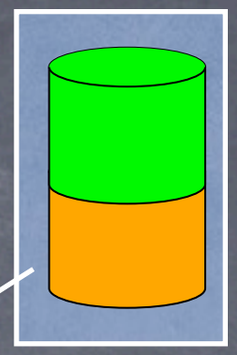
r_x
Φ_x
p_y^r
p_x^r

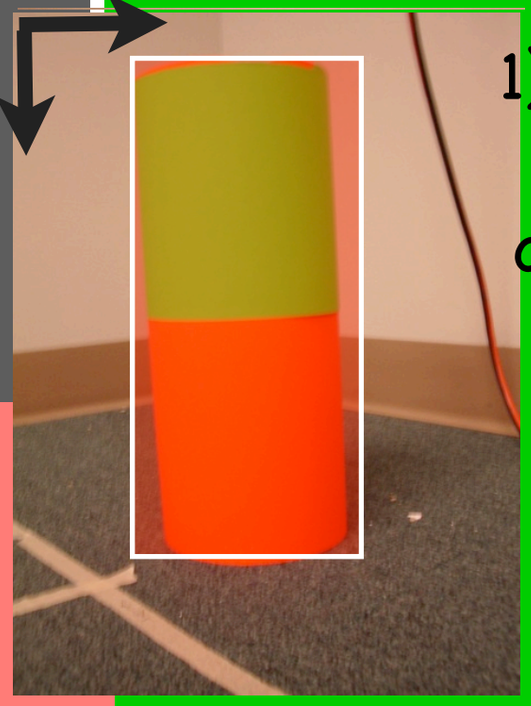
$r_x = \text{sqrt}((p_x^r)^2+(p_y^r)^2)$
distance of vector to object

$\Phi_x = \text{atan}(p_y^r/p_x^r)$
bearing of vector to object

p^r : 2D location of landmark center in field coordinates

Option 2: To evaluate a pose, we can compare the range and bearing of objects in robot coordinates





1) Estimate (r′,Φ′) from blob of observed object

r: distance to object

Φ:angle to object

40

40

1

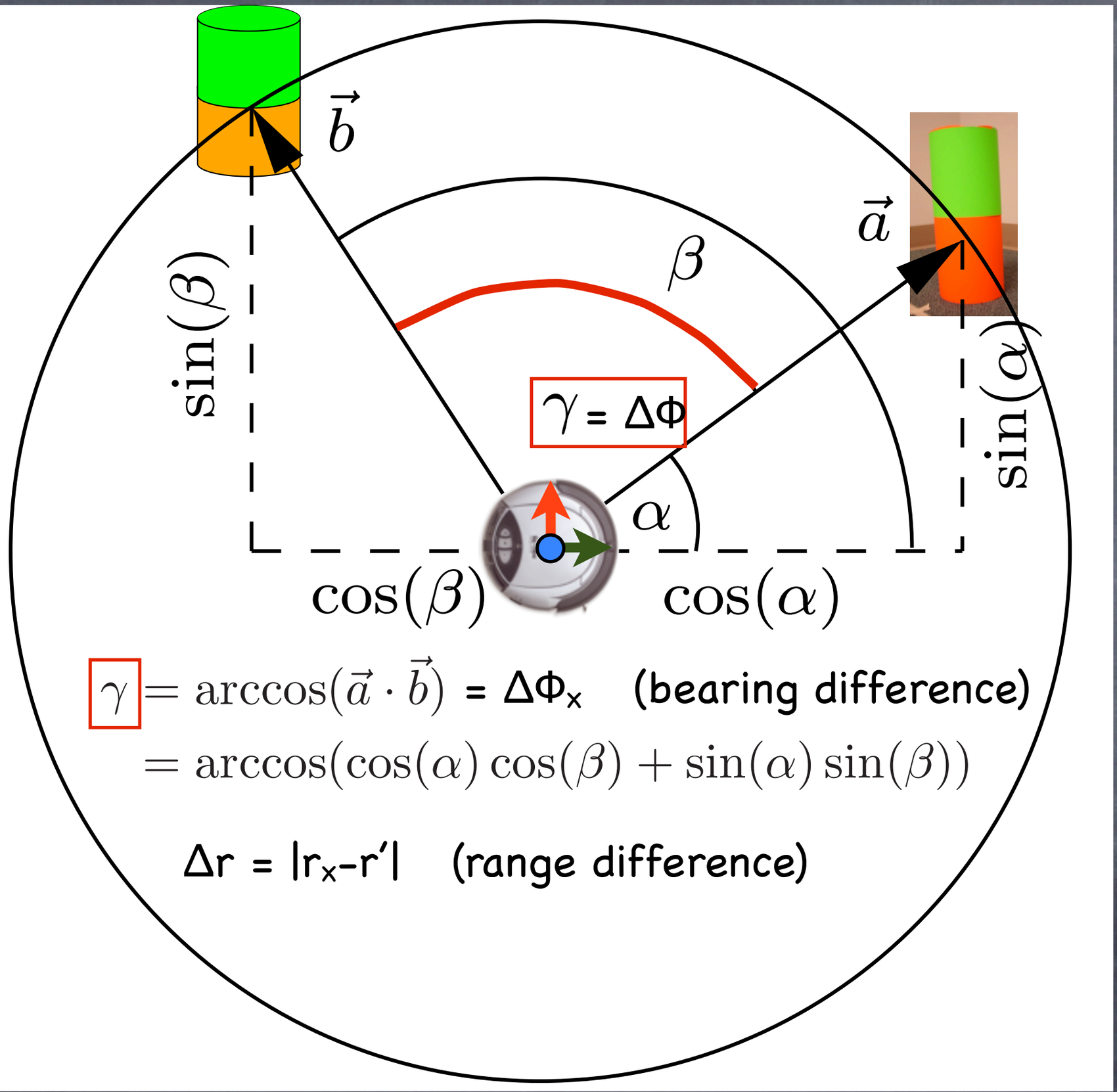
70

260

2) calculate (r_x,Φ_x) expected by given a hypothesized pose x

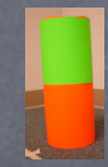
transform world into "robot's" perspective

3) compute weight from match between (r_x,Φ_x) and (r′,Φ′)



$l(z=$  $|x) \approx$

{

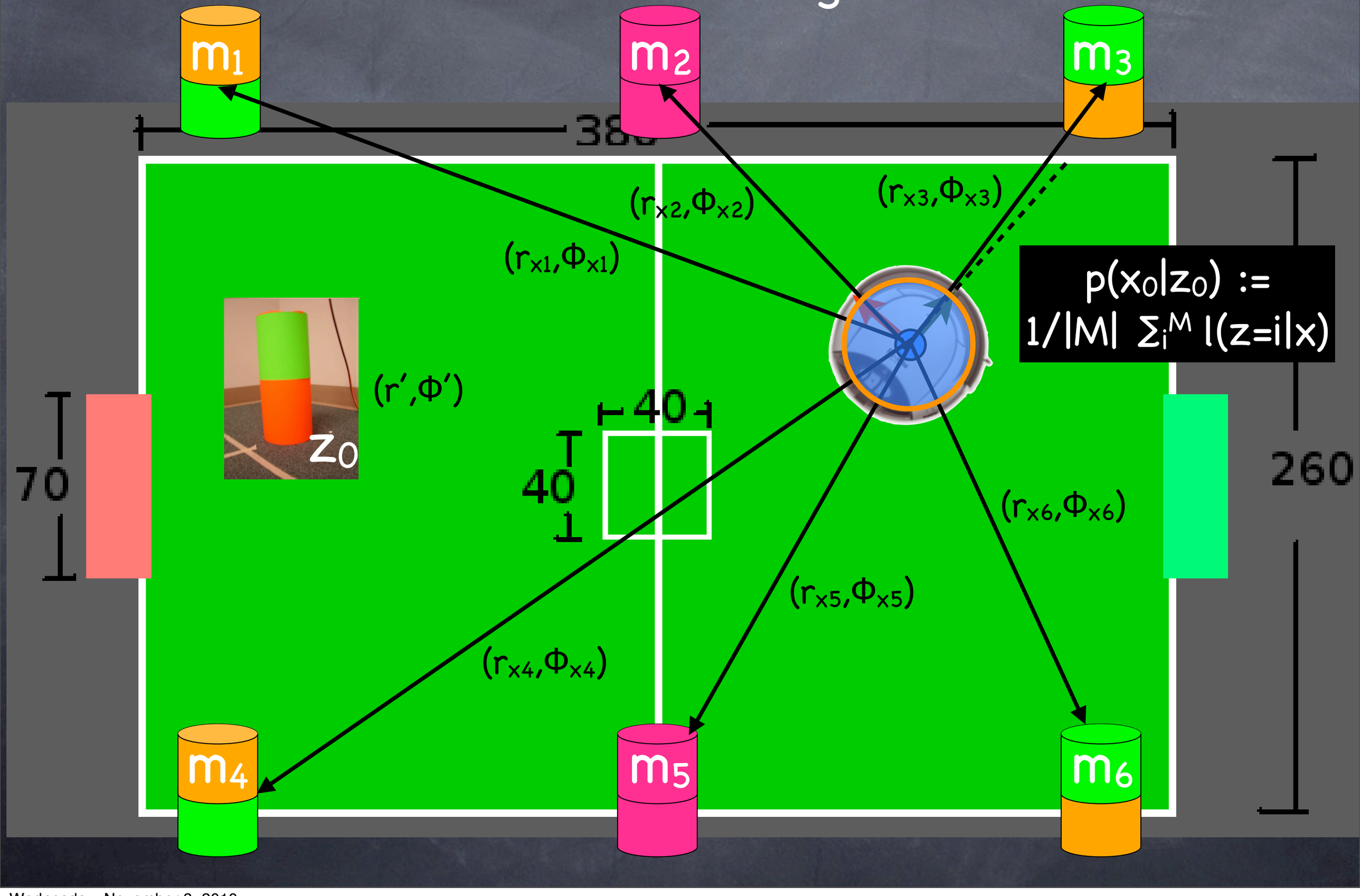
"low", if  out view and $-aov/2 < \Phi_x < aov/2$

"high", if  out view and $-aov/2 > \Phi_x < aov/2$

$\exp(-\Delta\Phi * \Delta r)$,

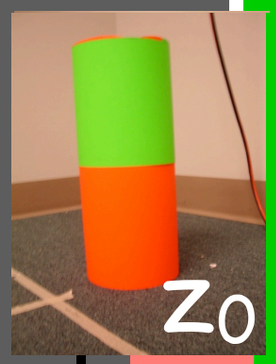
otherwise

for multiple landmarks (M), approximate by computing likelihood for each and then average

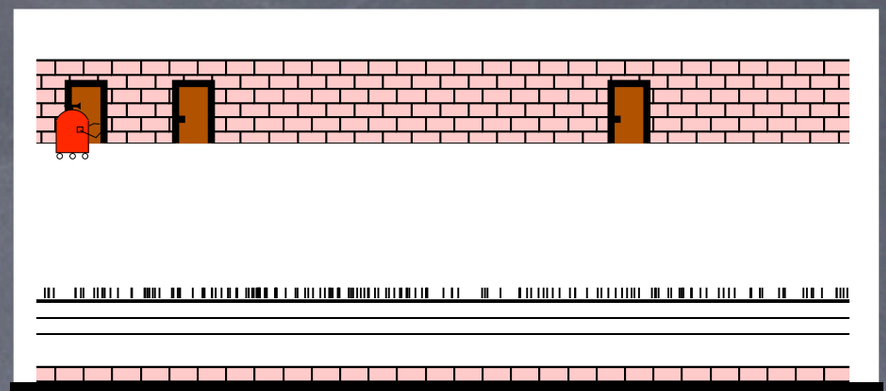


evaluate every pose sample

$$p(x_0|z_0) := 1/|M| \sum_i^M l(z=i|x)$$



Particle filter recap



What are we still missing to compute localization?

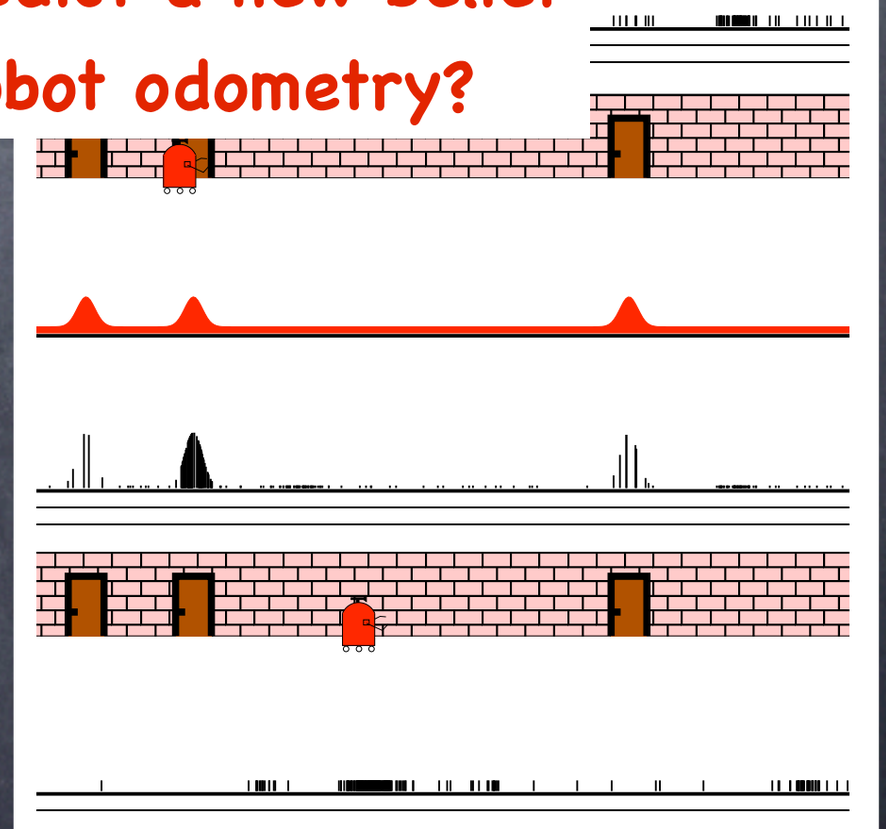
posterior likelihood dynamics prior

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \alpha p(\mathbf{Z}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})$$

"belief at t=k" "update" "predict" "belief at t=k-1"

How to evaluate the likelihood of a pose given robot observations?

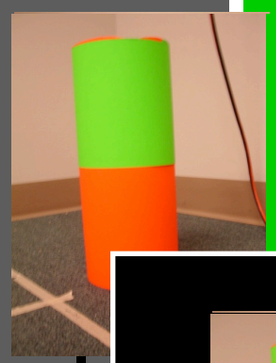
How to predict a new belief given robot odometry?

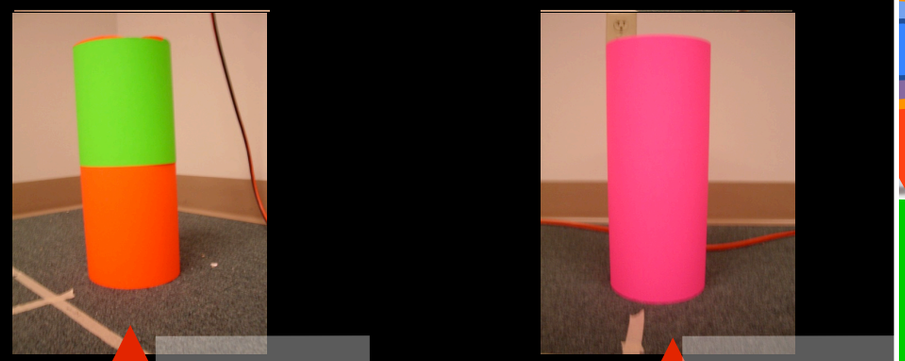


- recursive Markovian inference over time with predict-update loop
- Particle filter algorithms predict-update with sample set of poses

note: belief for particle filter are pose samples weighted by probability (noted by size of ellipse)

$p(x_0|z_0)$





Particle filter steps

1.(Re)sample particles

2.Predict forward in time

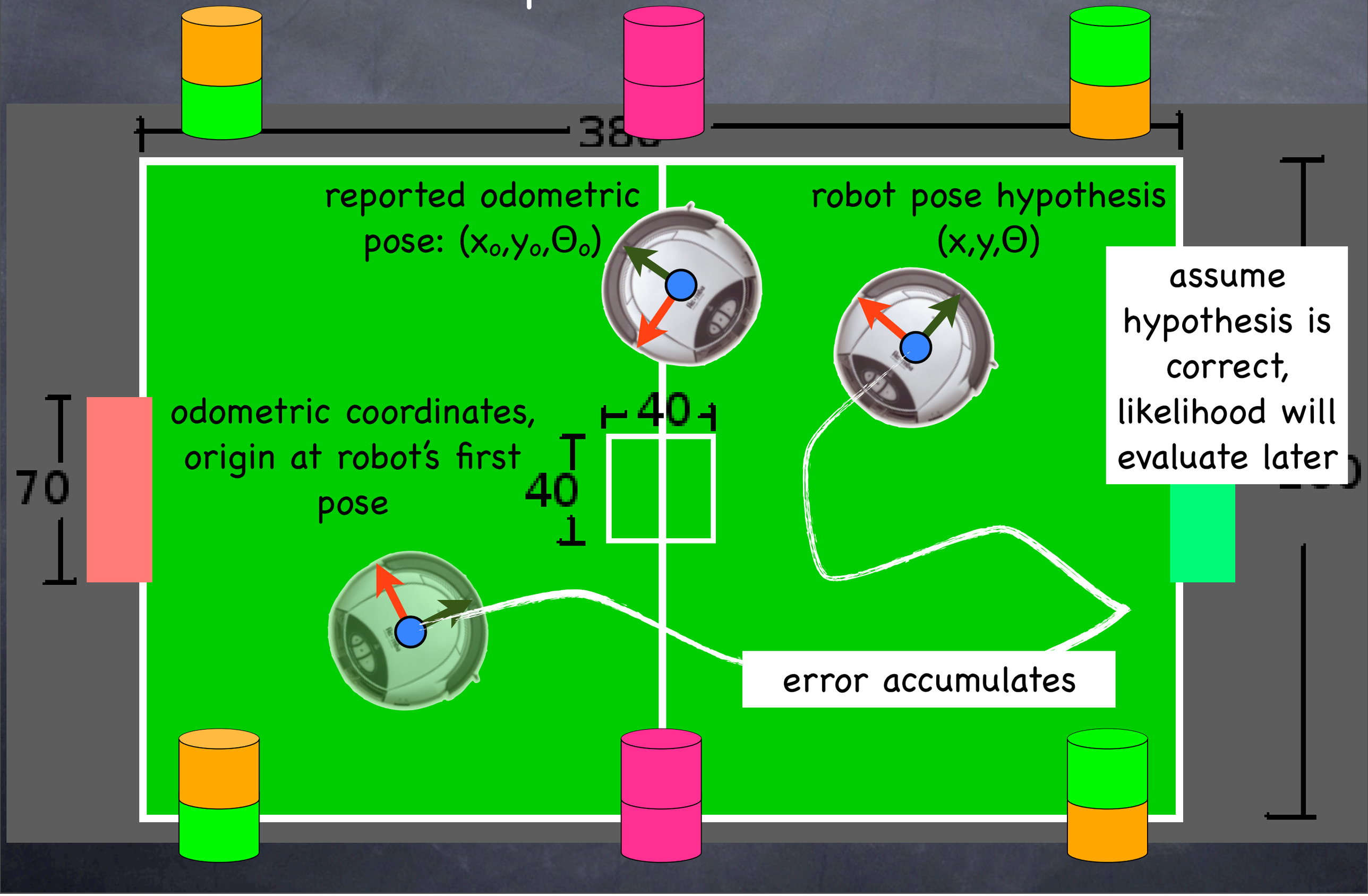
3.Update particle weights

4.Repeat

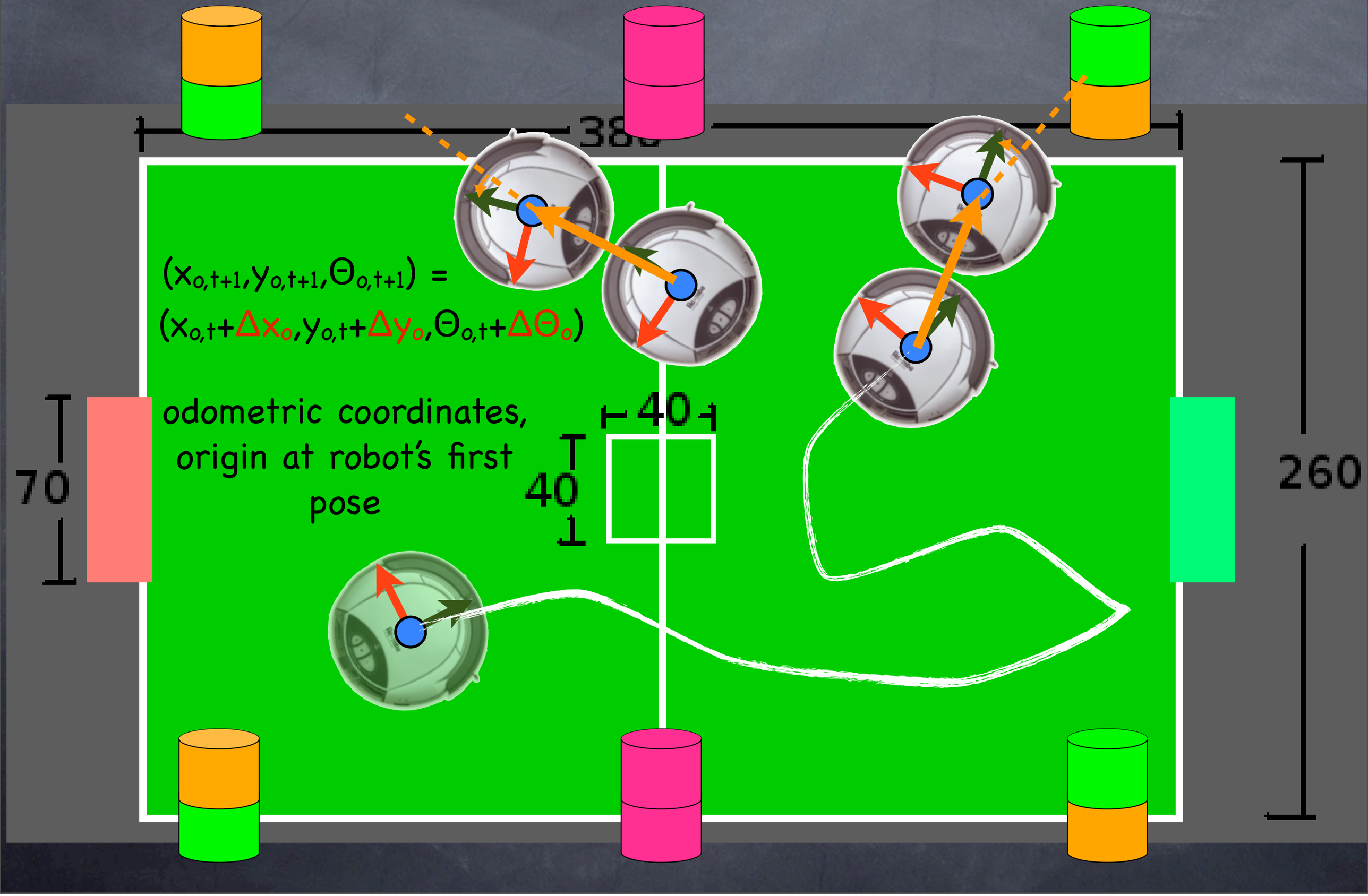
70

260

robot reports a pose estimate in odometric coordinates; we cannot trust this pose due to accumulated error



however, odometry over a short time is usably accurate

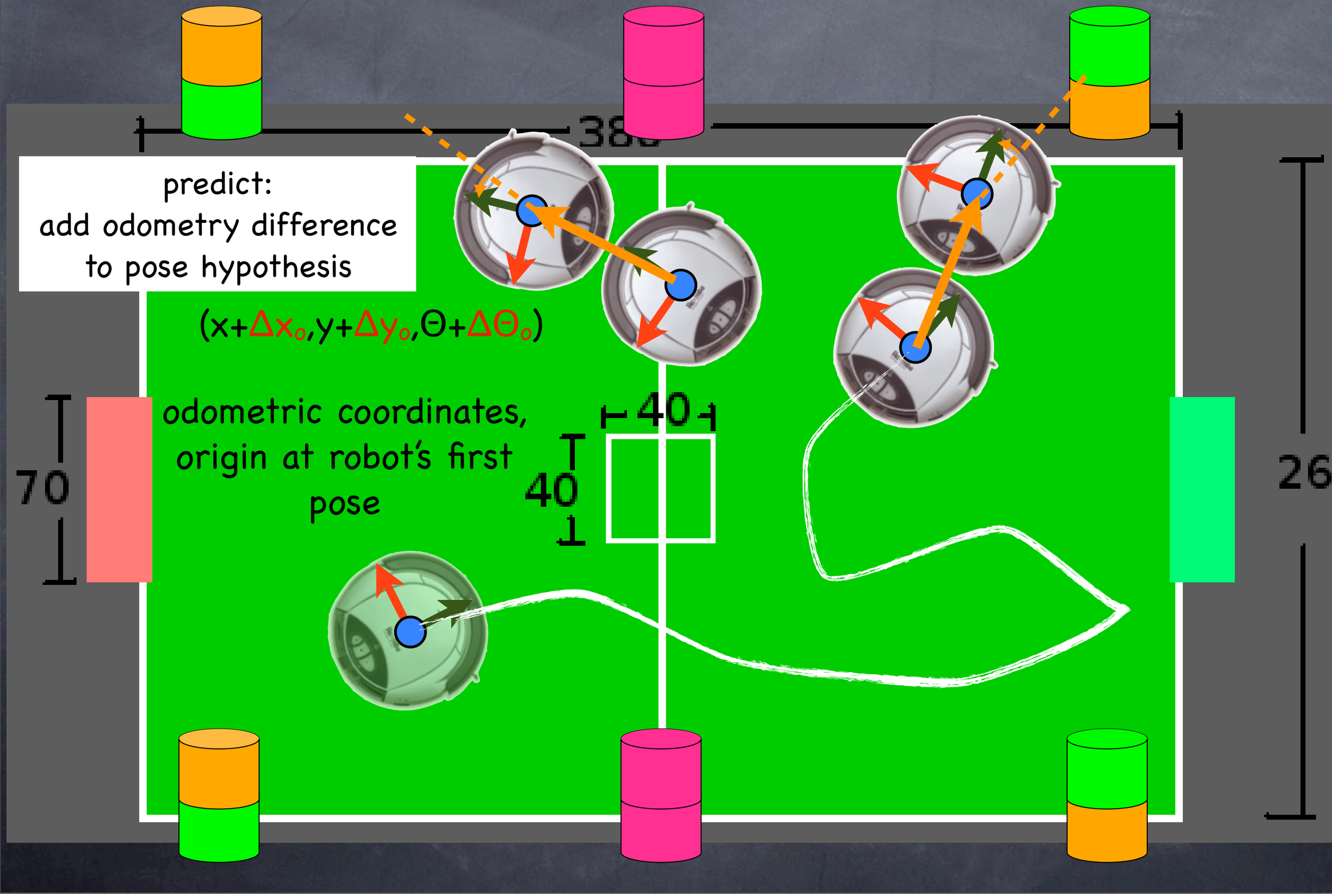


add odometric update to pose hypothesis?

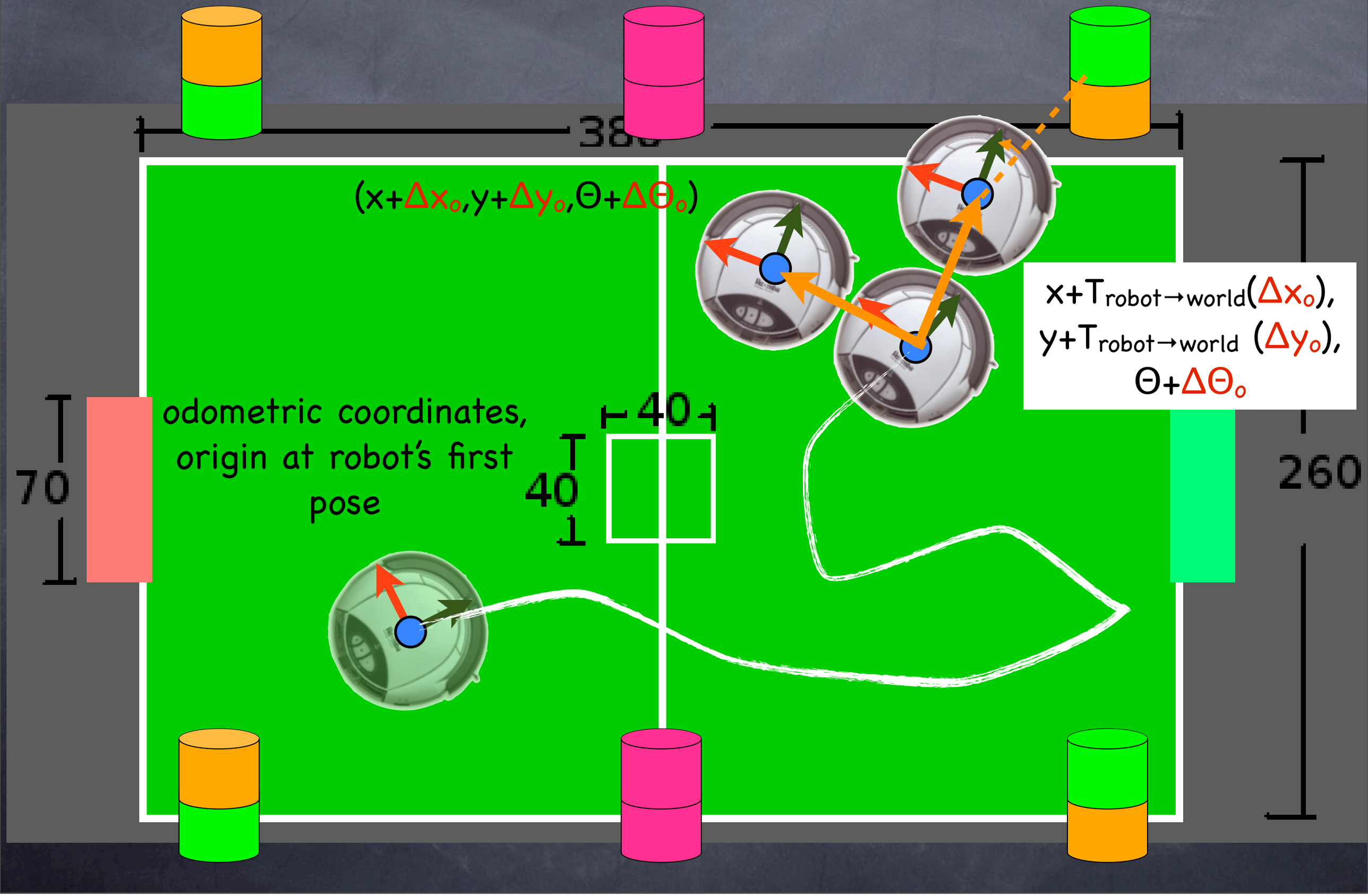
predict:
add odometry difference
to pose hypothesis

$(x+\Delta x_o, y+\Delta y_o, \Theta+\Delta\Theta_o)$

odometric coordinates,
origin at robot's first
pose



account rotation in field vs. robot coordinate spaces



remember from coordinate spaces lecture,
how do we actually compute $odom_{world}$?

Now, the direct odometry update can be applied

$pose_{world}\_x \rightarrow pose_{world}\_x + odom_{world}\_x$

$pose_{world}\_y \rightarrow pose_{world}\_y + odom_{world}\_y$

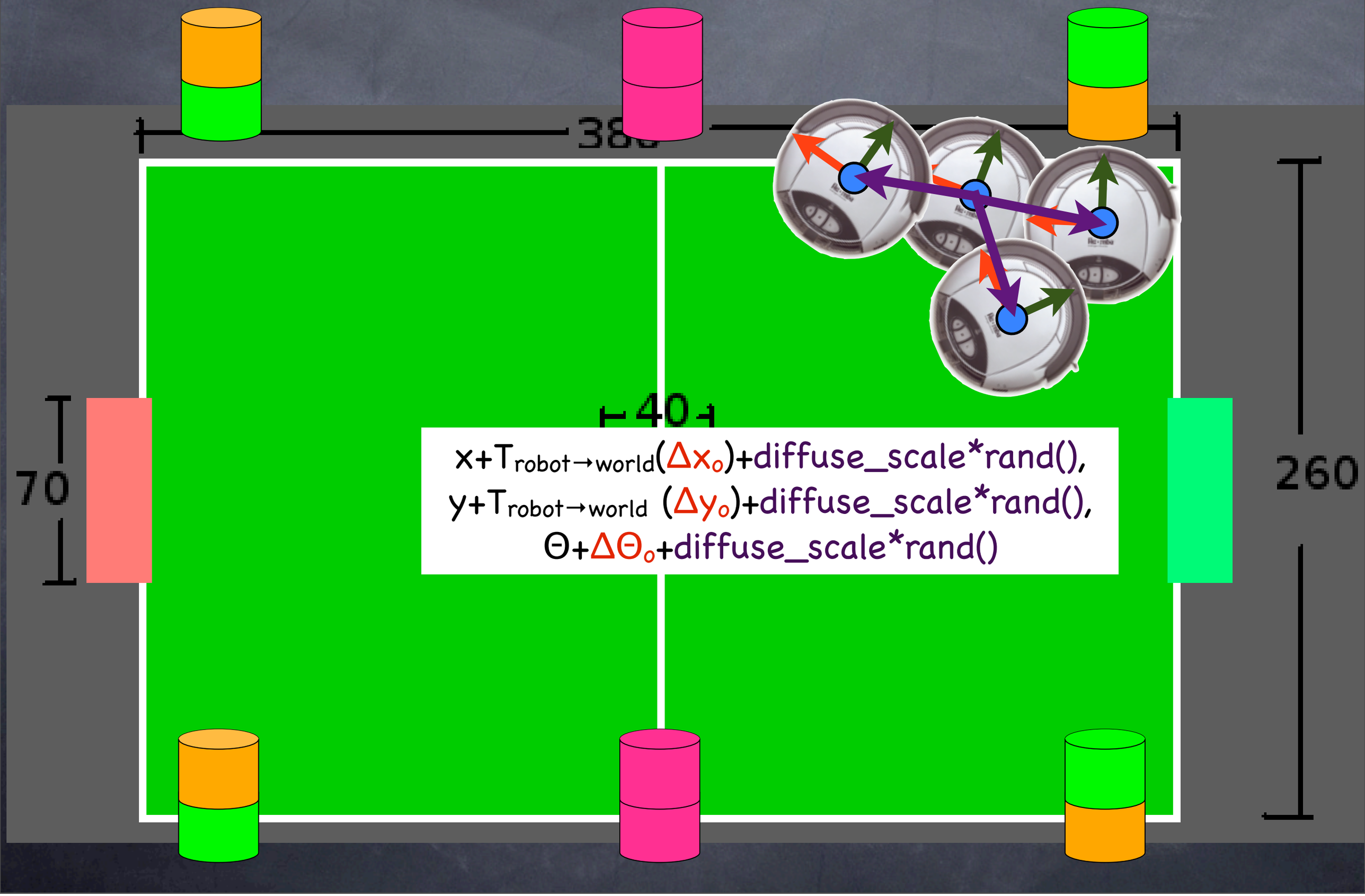
$pose\_a \rightarrow (pose\_a + \mathbf{angle}) \% 2pi$

where,
tx = pose_x
ty = pose_y
theta = pose_a

$$\begin{bmatrix} odom_{world}\_x \\ odom_{world}\_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} odom_{robot}\_x \\ odom_{robot}\_y \\ 1 \end{bmatrix}$$

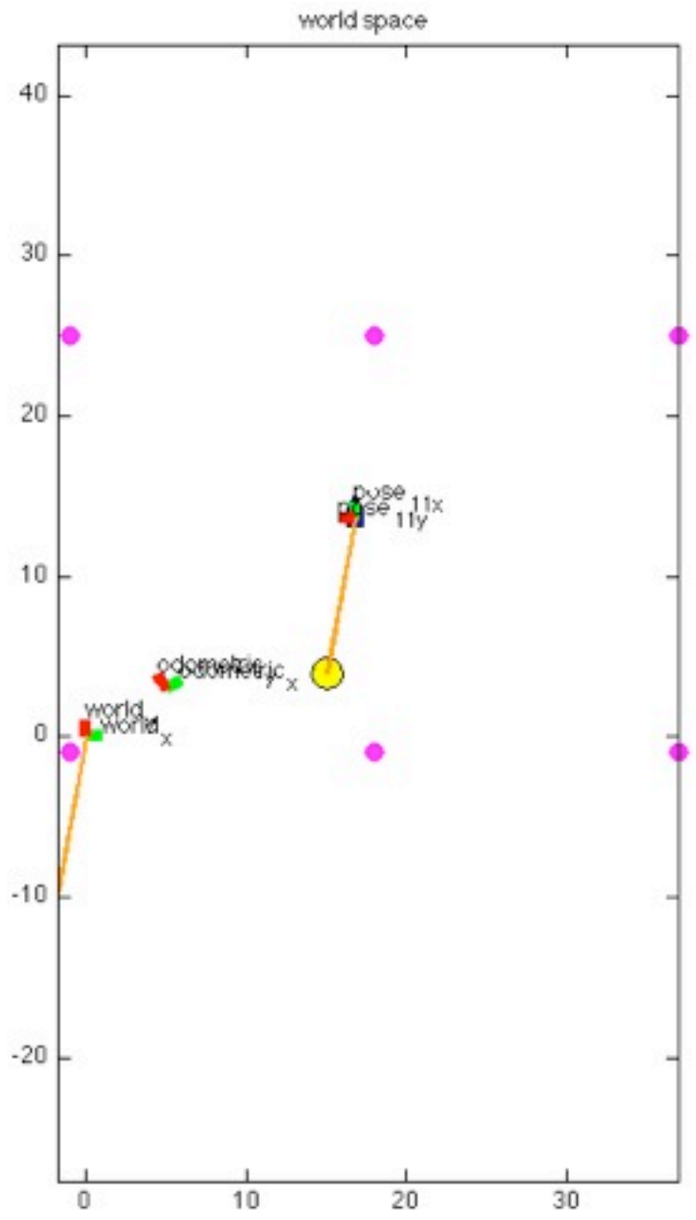
$odom_{world}$ =
translate(pose_x,pose_y) * rotate(pose_a) * $odom_{robot}$

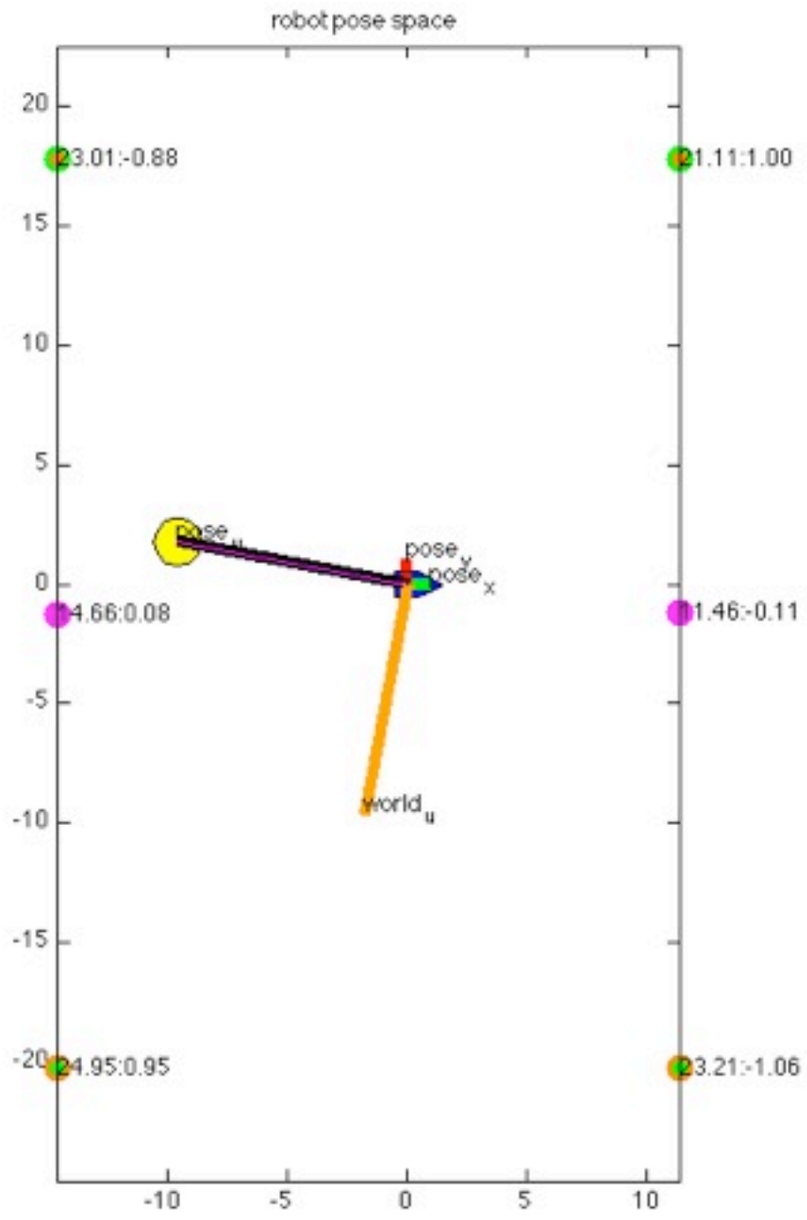
don't forget to resample with random offsets



matlab example

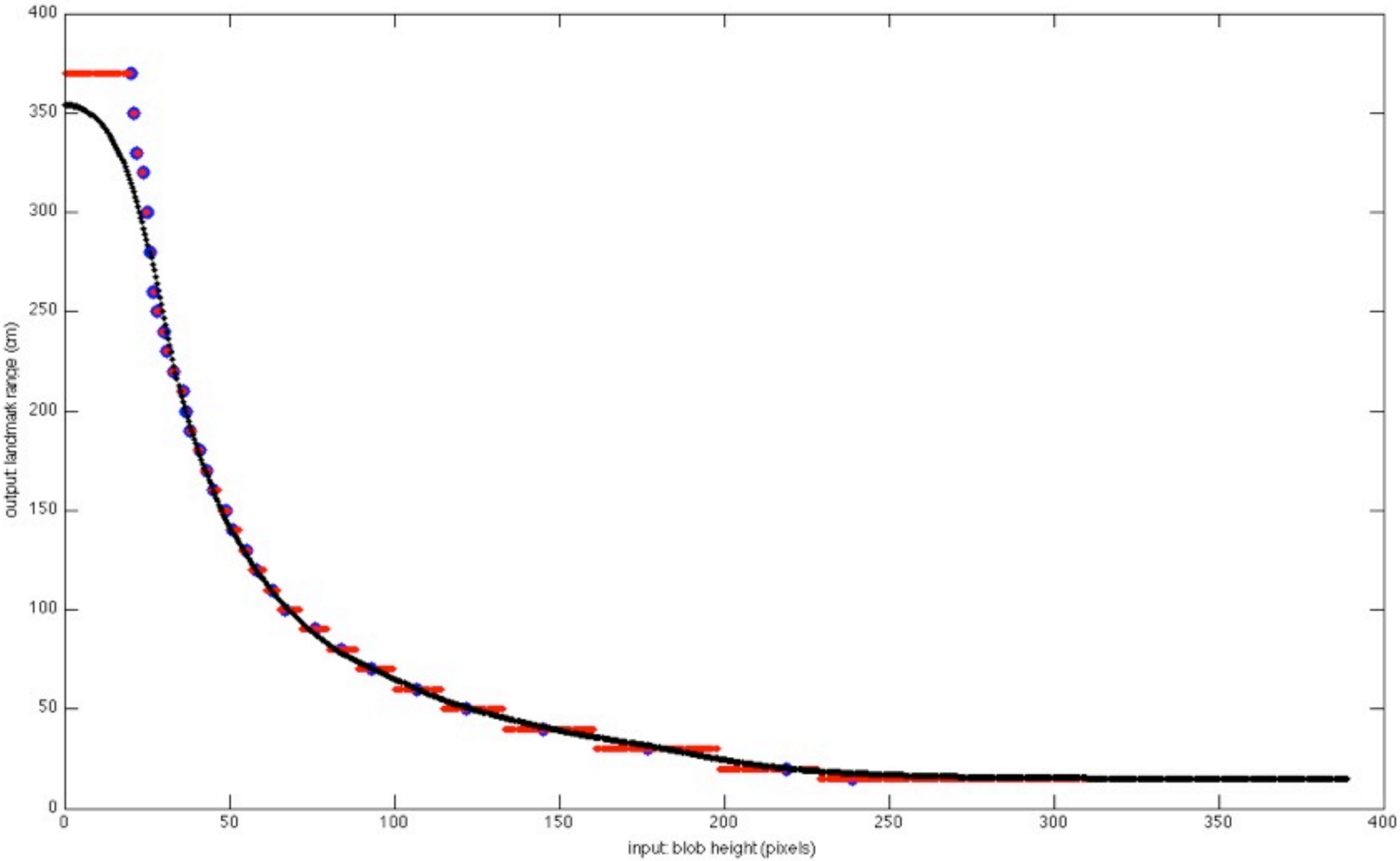
- field_coordinate_transform.m





matlab example

- range_function_approximation.m



videos!

- create particle filter
- Minerva