CMPack - High Level Vision

Carnegie Mellon University

September 29, 2003

Scott Lenser
Manuela Veloso

Vision Overview

Vision in CMPack is divided into two parts:

- low-level vision Performs bottom-up processing of image and provides summaries of important features of image.

- high-level vision Performs top-down processing of image. Uses expectations of objects that might be in image and features provided by low-level vision to find objects of interest and estimate their properties.

Vision can be thought of as the art of throwing out the information you don't want, while keeping the information you care about for your task.

High-level Vision Overview

- High-level vision is responsible for finding important objects in the camera image.
- High level vision uses as input the summarized image provided by the low-level vision.
- Many of the object detectors use multiple levels of detail from the summary provided by low-level vision.
- For each object found, high-level vision produces a confidence in [0.0,1.0] indicating the likelihood of this object actually being the object of interest.
- For each object found, high-level vision produces an estimate of its position in egocentric coordinates (relative to robot).

Objects Detected

The following objects are searched for in each image:

- Ball
- Markers
- Goals
- Robots

Object Detection - Rough Outline

Object detection for all objects follows roughly the following steps. The details of each step vary somewhat from object to object.

- Produce a set of candidate objects that might be this object from the lists of regions produced by the low-level vision. Usually this is simply a list of all regions of the appropriate color.
- Compare each candidate object to a set of models that predict features that the object should have when seen through a camera. Each model is referred to as a filter.
- The best match is selected to report to behaviors.
- The location of the best match relative to the robot is calculated.
- The position and quality of match for the best match are reported to behaviors.

Object Detection - Filtering Overview

The overall match of a candidate object to the models is performed using the following steps:

- Each individual filtering model produces a number in [0.0,1.0] representing the level of match between the candidate object and the model.

- The match levels are multiplied together to produce the overall match level.

- Some of the filters (models) are binary and always produce either 0.0 or 1.0. In this case, if the model produces a 0.0 level of match the candidate can be immediately rejected.

Object Detection - Filtering Overview

Most of the filters produce real valued results. This has important benefits.

- This allows for a grey region where the candidate somewhat matches the model but not very well.
- This grey area keeps good observations from being thrown out by one filter.
- Bad observations that are in the grey areas of several filters are still thrown out.

Object Detection - Filtering Overview

We will consider ball detection in detail.

We will cover the detection of the goals and markers in much less detail.

Object Detection - Rough Outline

Object detection for all objects follows roughly the following steps. The details of each step vary somewhat from object to object.

- Produce a set of candidate objects that might be this object from the lists of regions produced by the low-level vision.
- Compare each candidate object to a set of models that predict features that the object should have when seen through a camera.
- The best match is selected to report to behaviors.
- The location of the best match relative to the robot is calculated.
- The position and quality of match for the best match are reported to behaviors.

Candidate Generation

- ball - The candidates objects for the ball are the 10 largest orange regions in the image.
- yellow goal - The candidates objects for the yellow goal are the 10 largest yellow regions in the image.
- cyan/pink marker - The candidates objects for the cyan/pink are the combinations of the 10 largest pink regions and the 10 largest cyan regions in the image.

Object Detection - Rough Outline

Object detection for all objects follows roughly the following steps. The details of each step vary somewhat from object to object.

- Produce a set of candidate objects that might be this object from the lists of regions produced by the low-level vision.
- Compare each candidate object to a set of models that predict features that the object should have when seen through a camera.
- The best match is selected to report to behaviors.
- The location of the best match relative to the robot is calculated.
- The position and quality of match for the best match are reported to behaviors.

Ball Detection - Filtering Models

The following filters are applied to the candidate regions.

- Minimum size - makes sure the ball has a bounding box that is at least 3 pixels tall and wide and 7 pixels in total area. This ensures sufficient discrimination from noise in the image.

- Square bounding box - makes sure the bounding box of the candidate is roughly square. This uses an unnormalized Gaussian for its output. The output is given by the following equations where $C = .2$ if the candidate region is on the edge of the image and .6 otherwise.

$$\begin{aligned} d &= \frac{|w-h|}{w+h} \\ o &= e^{-(\frac{d}{C})^2/2} \end{aligned}$$

Ball - Filtering Models [cont.]

- Area ratio - compares the area covered by the pixels to the area covered by the bounding box. The output is defined by the following equations where $C = .2$ if the candidate is on the image edge and $.6$ otherwise. a is the area of the candidate.

$$\begin{aligned} m &= \pi * w * h/4 \\ d &= \frac{|m - a|}{m + a} \\ o &= e^{-(\frac{d}{C})^2/2} \end{aligned}$$

- Elevation - binary filter which ensures the elevation of the ball is less than $5°$.

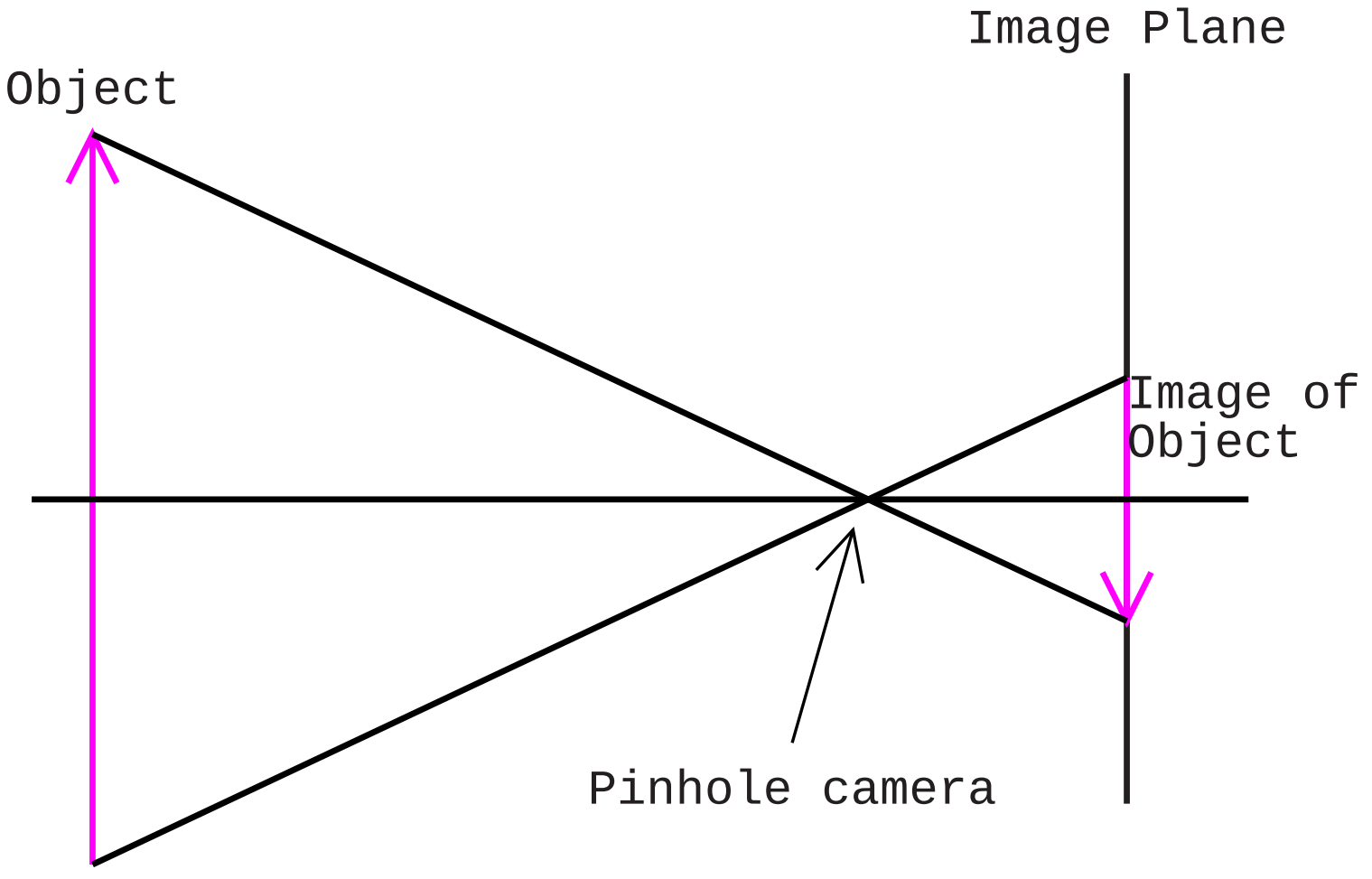
A few more filters are applied after the position is estimated.

Object Detection - Rough Outline

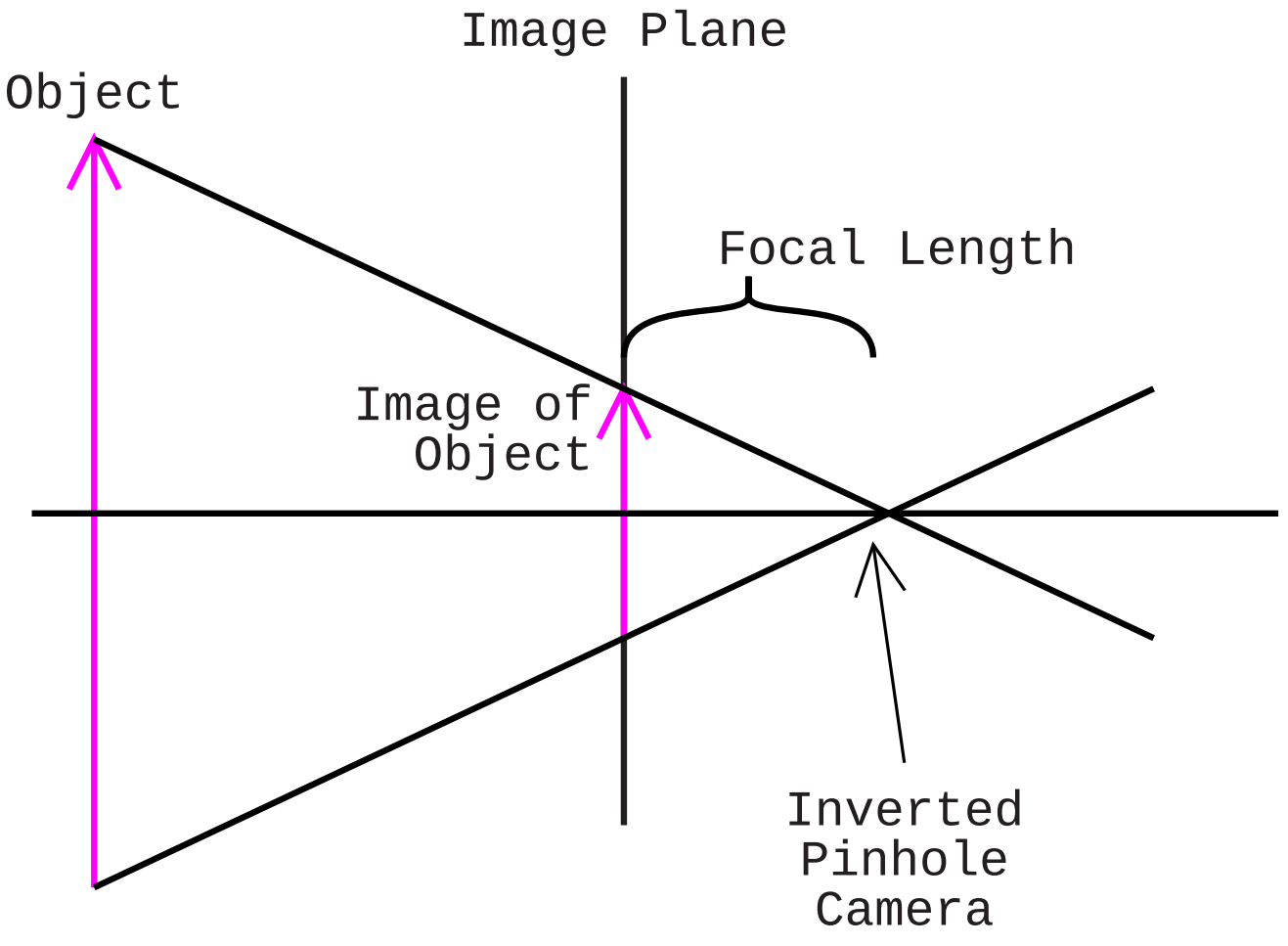
Object detection for all objects follows roughly the following steps. The details of each step vary somewhat from object to object.

- Produce a set of candidate objects that might be this object from the lists of regions produced by the low-level vision.
- Compare each candidate object to a set of models that predict features that the object should have when seen through a camera.
- The best match is selected to report to behaviors.
- The location of the best match relative to the robot is calculated.
- The position and quality of match for the best match are reported to behaviors.

The Pinhole Camera Model

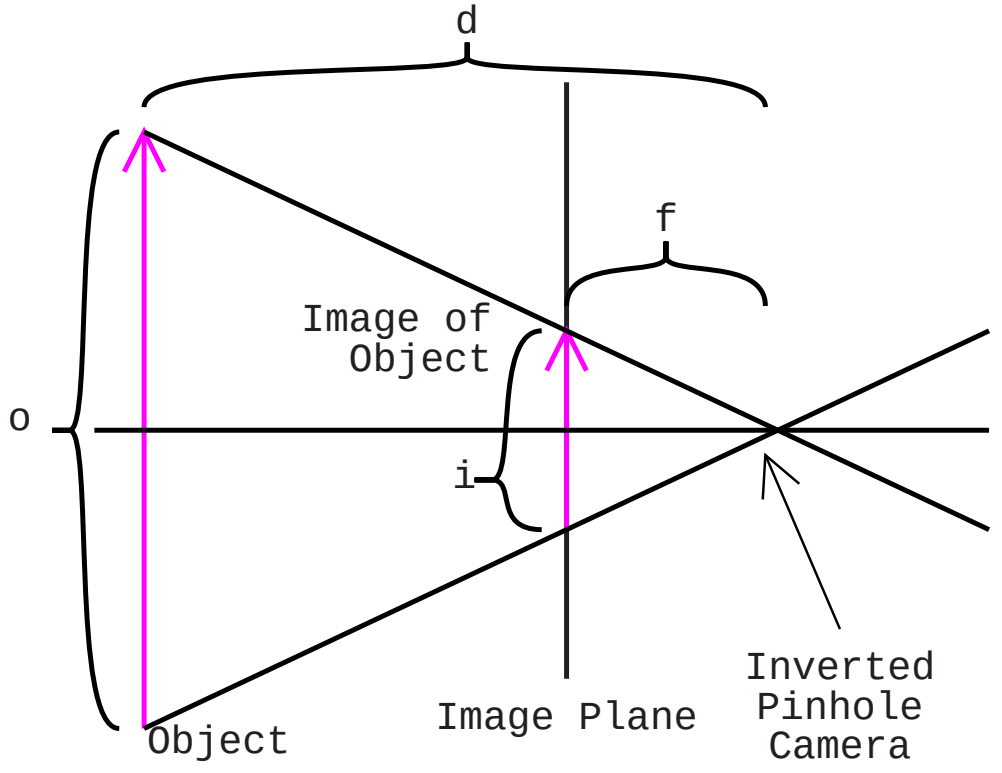


The Inverted Pinhole Camera Model



Calculating Distance

Calculating distance using the inverted pinhole camera model.

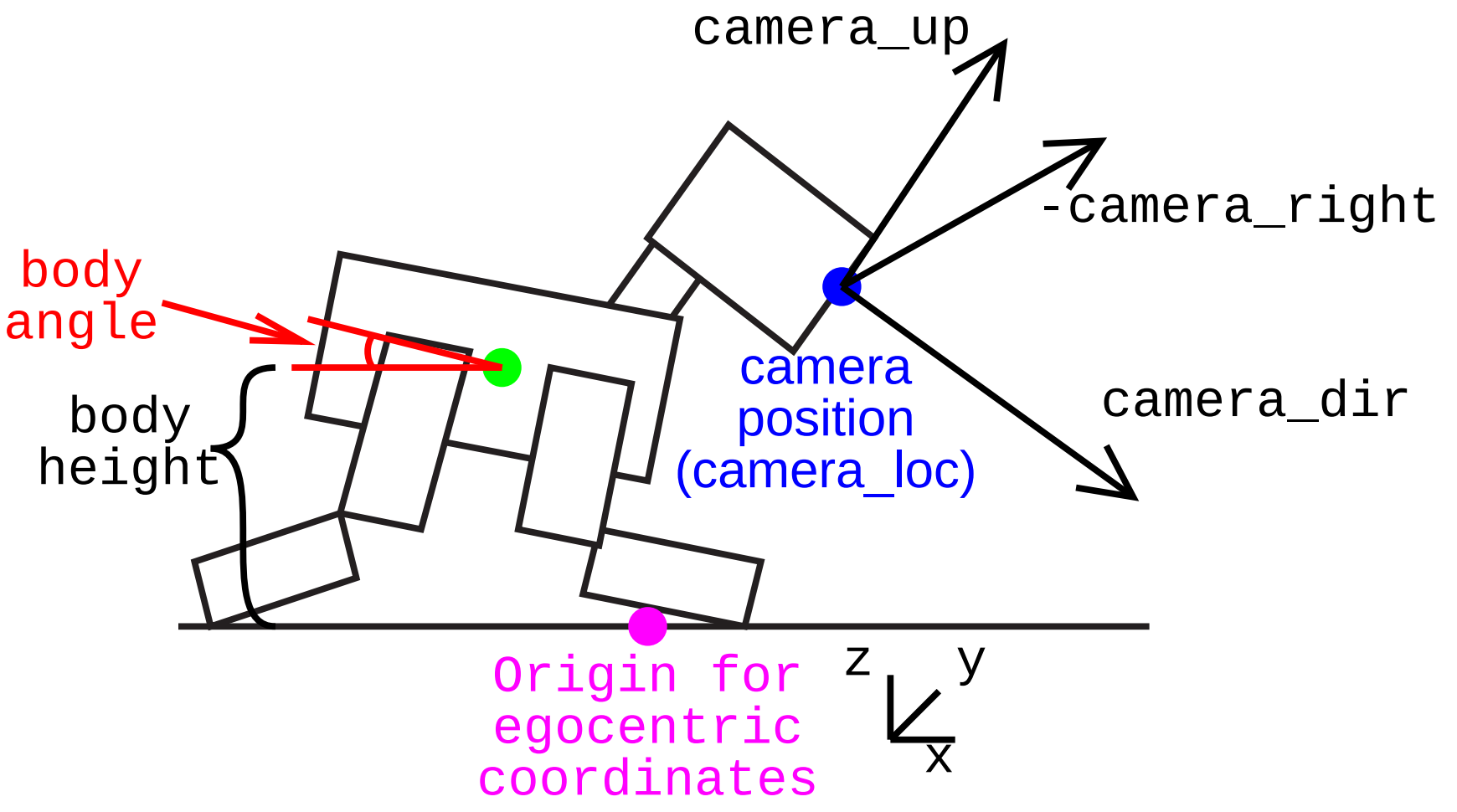


$$\frac{d}{o} = \frac{f}{i}$$

Calculation of Camera Position

- Position of camera can be calculated based on body position and head position relative to body.
- Body position is known from walking engine.
- Head position relative to body can be found from forward kinematics of the head using head joint positions.
- Camera position is characterized using 1 point and three unit vectors.
- camera_loc is defined as the position of the camera relative to the egocentric origin
- camera_dir, camera_up, and camera_right are unit vectors in egocentric space in the directions the camera is facing, up on the image, and right on the image, respectively.

Calculation of Camera Position

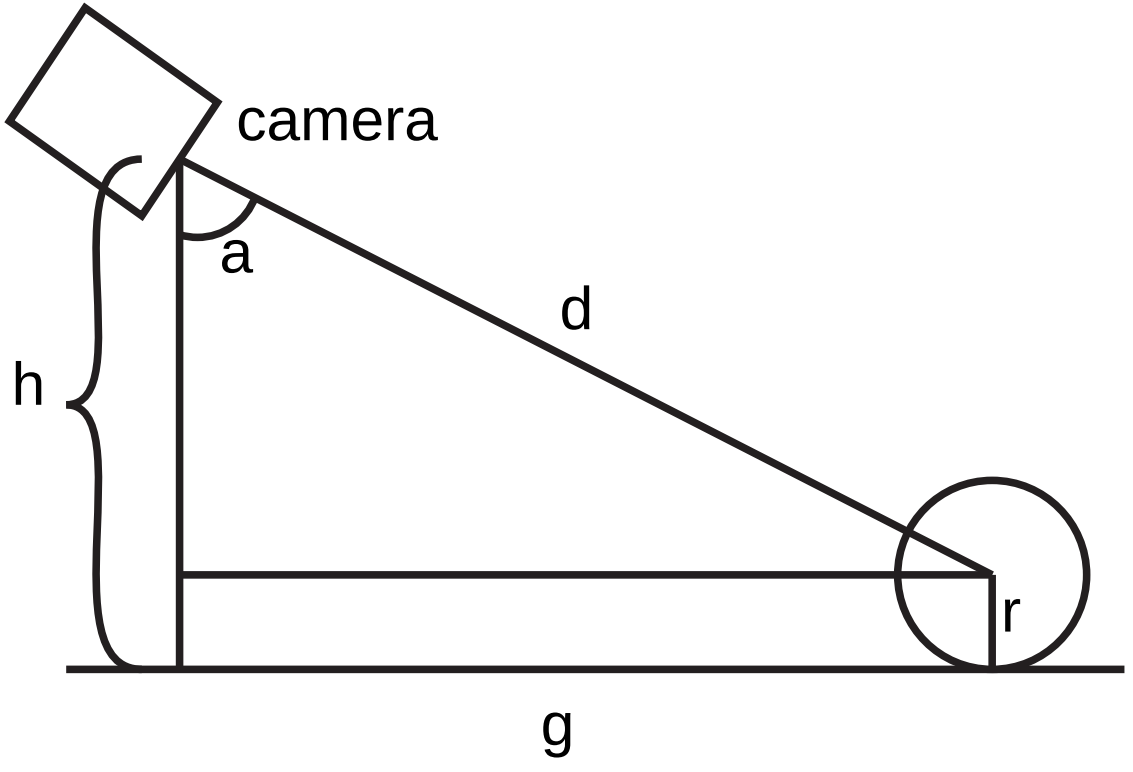


Ball Detection - Position Estimation

- We have 2 different methods for estimating the position of the ball.
- We choose between them based upon whether the ball is on the edge of the image or not.
- The first is more accurate but relies on the pixel size of the ball in the image which may be inaccurate if the ball is partially off the image (or occluded).

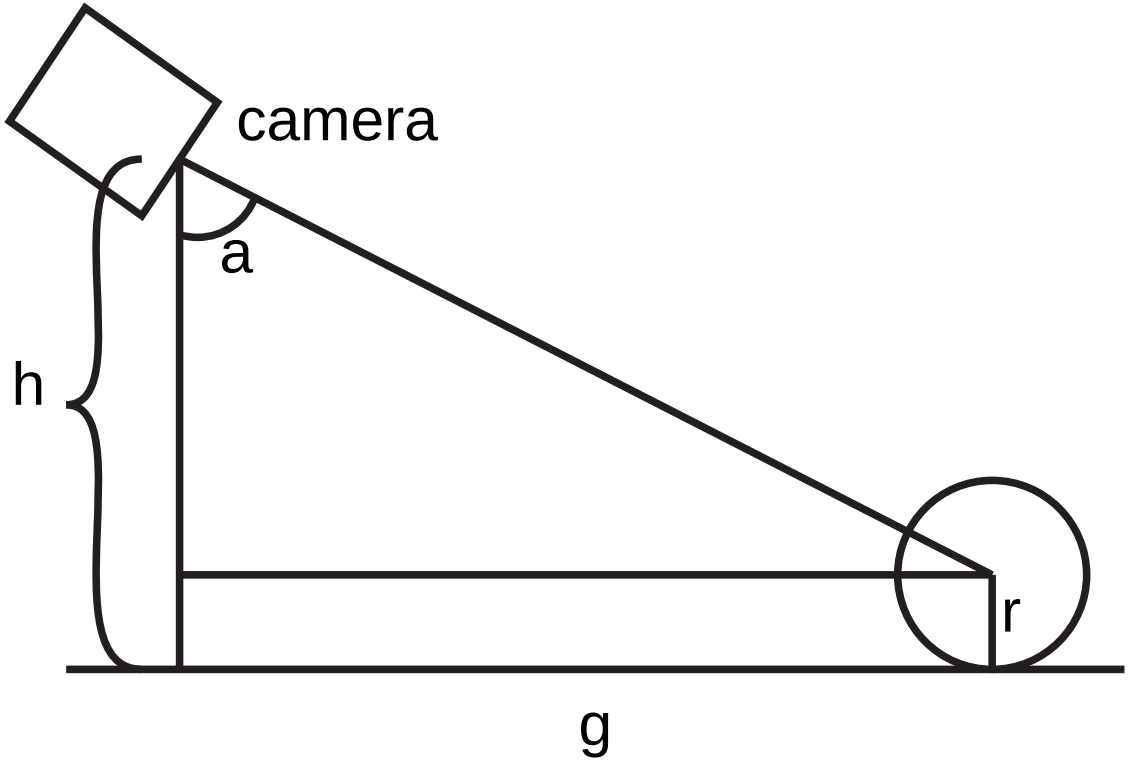
Ball Detection - Position Estimation

The ball position estimation problem is over constrained, here is a diagram of the problem. g is the unknown.



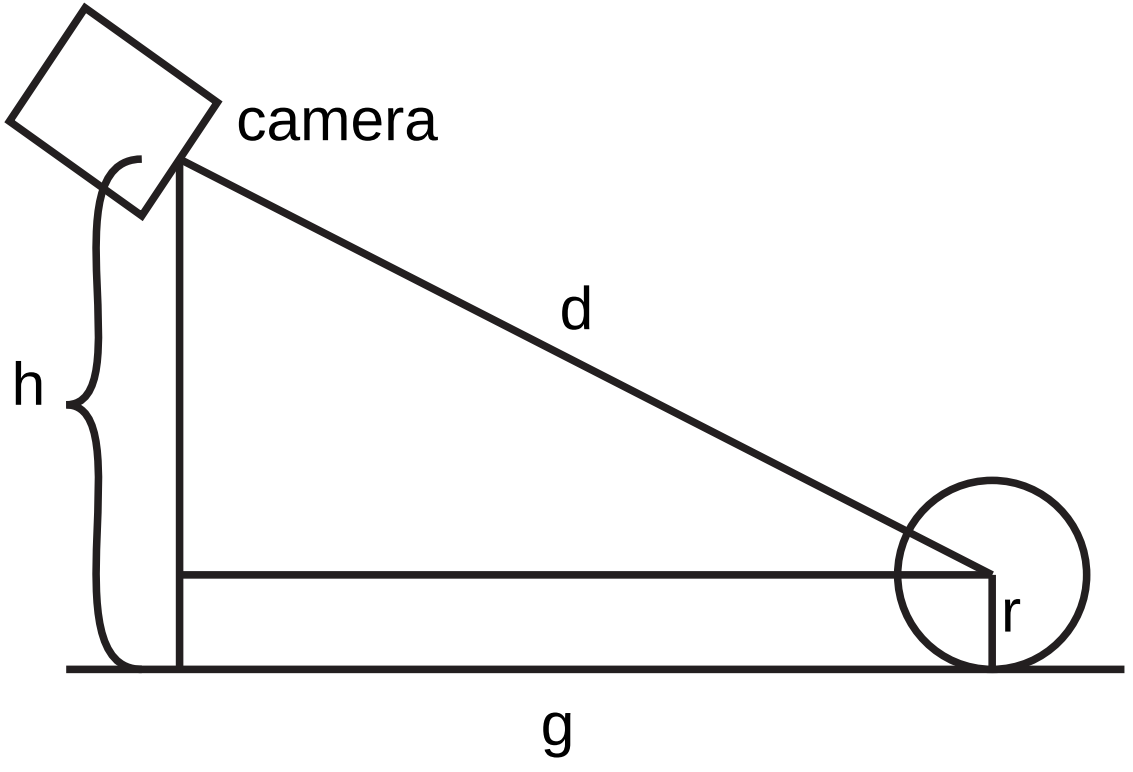
Ball Detection - Position Estimation

This method works all of the time. We use it when the ball is on the edge of the image.



Ball Detection - Position Estimation

This method is more accurate but only works when all of the ball can be seen. We use it when the ball is NOT on the edge of the image.



Object Detection - Rough Outline

Object detection for all objects follows roughly the following steps. The details of each step vary somewhat from object to object.

- Produce a set of candidate objects that might be this object from the lists of regions produced by the low-level vision.
- Compare each candidate object to a set of models that predict features that the object should have when seen through a camera.
- The best match is selected to report to behaviors.
- The location of the best match relative to the robot is calculated.
- The position and quality of match for the best match are reported to behaviors.

Ball - Filtering Models [cont.]

- Projection error - models the expected relative error in projection between the 2 methods of estimating the ball position. The output is generated by the following equations where d is the angular difference between the angle of the ball in the image and the angle of the ball calculated from the higher quality position estimate.

$$\begin{aligned} x &= \max(|d/5^\circ| - .5, 0) \\ o &= e^{-(\frac{x}{.75})^2/2} \end{aligned}$$

Ball - Filtering Models [cont.]

After a candidate ball has passed these filters, the color of items around it in the image is considered in the following steps.

- The bounding box of the candidate is expanded 3 pixels in every direction.
- A scan is made through the segmented image along this rectangle.
- A histogram is constructed counting the occurrence of each symbolic color class.
- Parts of the rectangle off the edge of the image are ignored.

Ball - Filtering Models [cont.]

Here is an example scanning box shown in bright purple.

Ball - Filtering Models [cont.]

These counts are used in the following 2 filters.

- Red vs. area - filters out some candidate balls that are actually part of the red robot uniforms. The output is given by

$$\begin{aligned} n &= 5 * g + 3 * w + 3 * b - r - 2 * y \\ o &= \text{bound}(1 + n/a, 0.0, 1.0) \end{aligned}$$

- Green filter - ensures ball is near green. The output is given by (p is number of pixels in scan)

$$\begin{aligned} g &= 2 * g + 1.5 * w + r + b - .5 * y \\ o &= \text{bound}(\max(g + 1, 1)/(p + 1), 0.0, 1.0) \end{aligned}$$

Ball - Filtering Models [cont.]

- The final filter applied is used to distinguish between distant balls and the fringe of the red robot uniforms.
- This binary filter is only applied to balls which are estimated to be over 1.5m from the robot.
- The filter output is based on the following equation.
- $\vec{c}$,$\vec{o}$,$\vec{r}$ are the avg. YUV color of the candidate ball, orange, and red (which has been misclassified as orange), respectively.

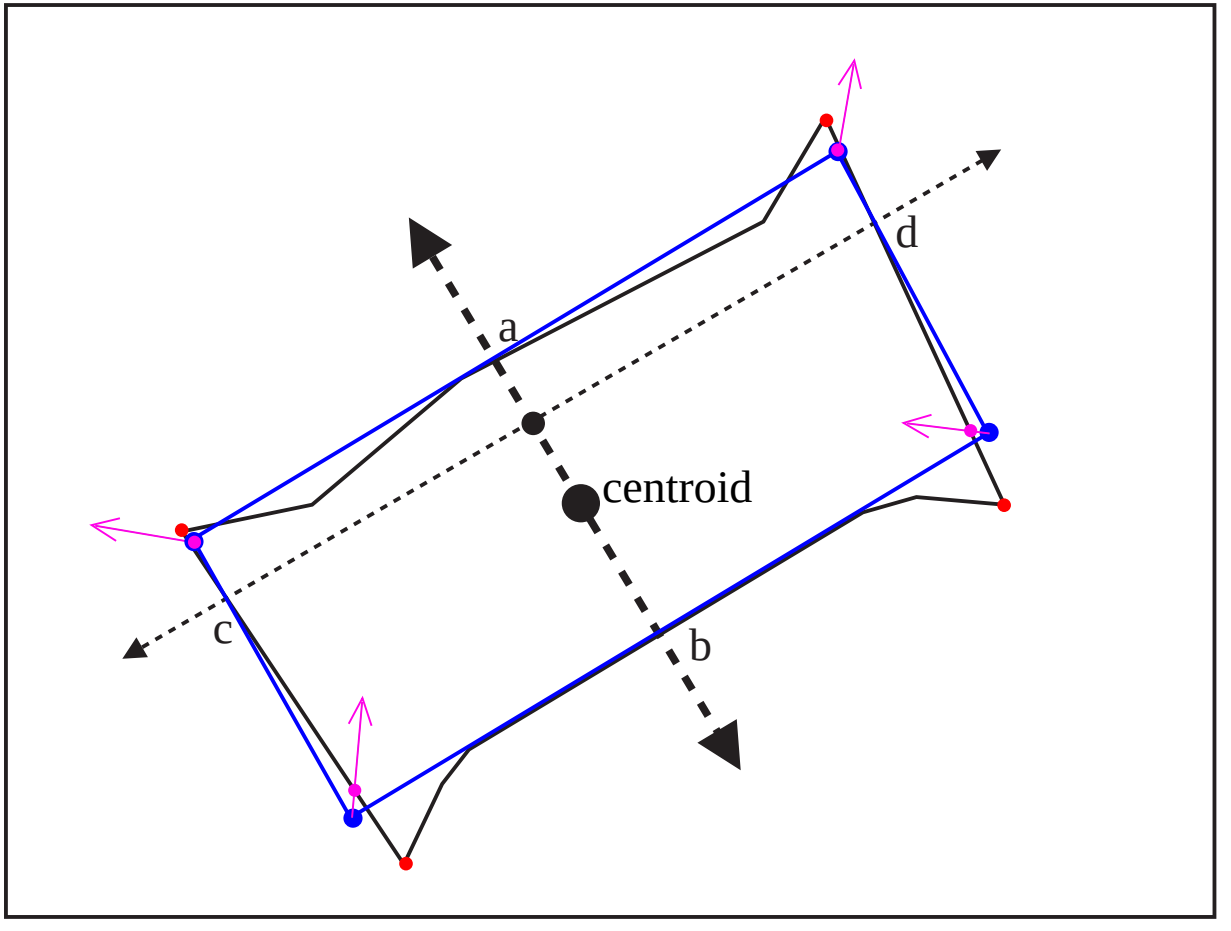
$$r = \frac{1}{|\vec{r} - \vec{o}|} \cdot \left((\vec{c} - \vec{o}) \cdot \frac{\vec{r} - \vec{o}}{|\vec{r} - \vec{o}|} \right)$$

  The ball is rejected if $r > .75$ (r is the roughly the redness of the ball). The ball is accepted otherwise.
- Note that if $\vec{c} = \vec{o}$ then $r = 0$ and if $\vec{c} = \vec{r}$ then $r = 1$.

Goals - Region preprocessing

- Before filters are applied to the goals, the candidate goal region is prepocessed to find the corners of the goal.
- The corners are found using a multi-step process involving calculation of several approximate corner positions, several scans through the image, and gradient descent. This will be covered a little more in the next slide.
- Goals are represented as three distinct objects, a left goal post, a right goal post, and a central goal area. This is because goal posts have more well defined locations but may not always be visible.

Goals - Corner Finding



Goals - Filtering Models

Goals posts are filtered using the following models/filters:

- Minimum area filter
- Minimum height filter
- Minimum width filter
- Verticality of vector from bottom corner to top corner
- Ratio of width to height
- Size of green region located underneath goal post
- Size of white region located to side of goal post near bottom of goal post.

Markers - Filtering Models

Markers are filtered using the following models/filters:

- Minimum area of regions filter
- Verticality of vector from bottom region to top region
- Ratio of area of regions compared to distance between regions
- Ratio of area of top region to area of bottom region

Object Detection - Rough Outline

Object detection for all objects follows roughly the following steps. The details of each step vary somewhat from object to object.

- Produce a set of candidate objects that might be this object from the lists of regions produced by the low-level vision.
- Compare each candidate object to a set of models that predict features that the object should have when seen through a camera.
- The best match is selected to report to behaviors.
- The location of the best match relative to the robot is calculated.
- The position and quality of match for the best match are reported to behaviors.

Position Estimation

The following method is used to estimate distances to goal posts and markers.

- Project rays through the centroid of each region for markers or the goal corners for goals.
- Rotate these rays around the angular bisector of the rays until the rays lie in the same vertical plane.
- Find the distance from the robot at which points on these rays which are directly over one another are separated by the height of the object.
- Use this distance to calculate the position of the object.