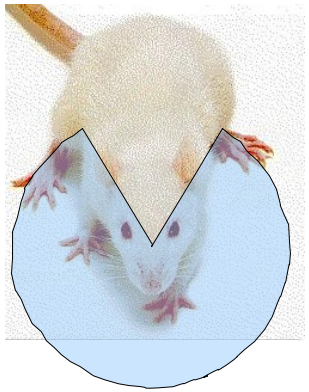
Local and World Maps

15-494 Cognitive Robotics
David S. Touretzky &
Ethan Tira-Thompson

Carnegie Mellon
Spring 2013

Horizontal Field of View



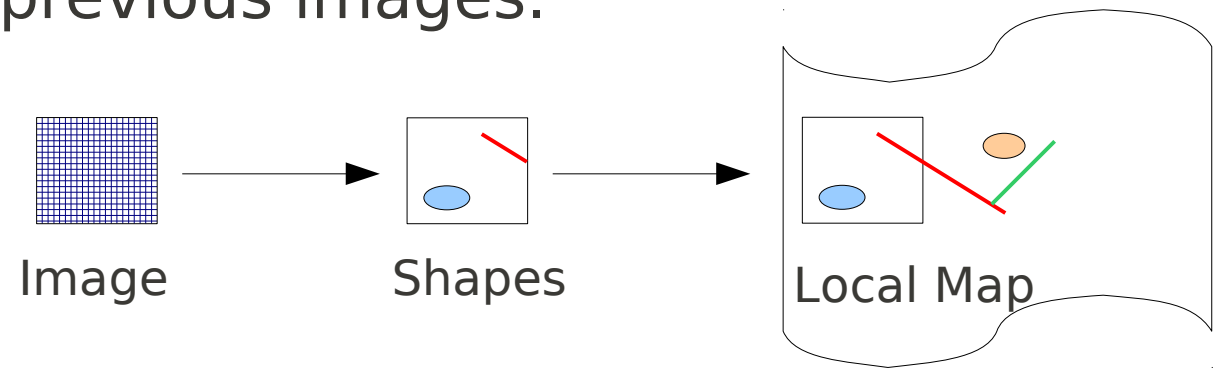
Rat: 300 deg.

Human: 200 deg.

Typical webcam:
60 deg.

Seeing A Bigger Picture

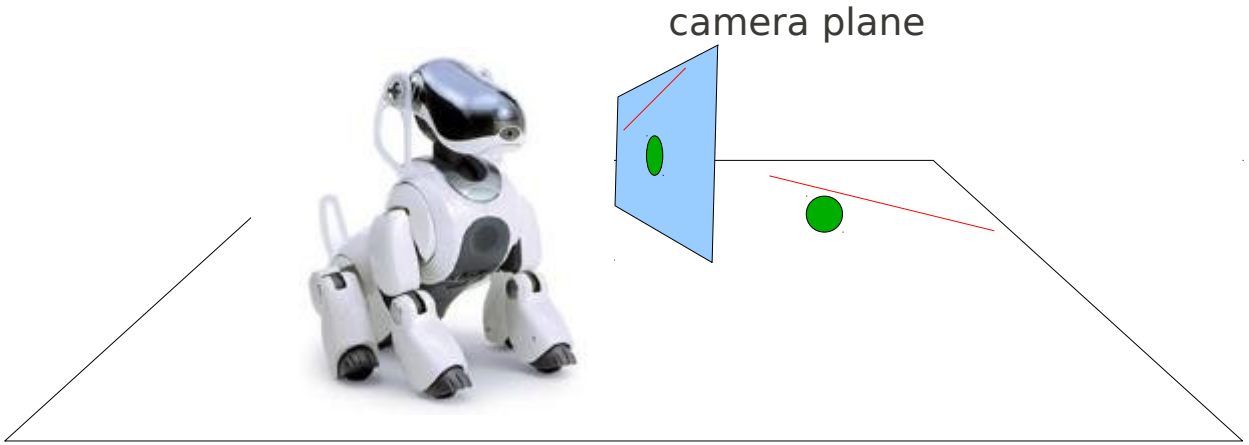
- How can we assemble an accurate view of the robot's surroundings from a series of narrow camera frames?
- First, convert each image to symbolic form: shapes.
- Then, match the shapes in one image against the shapes in previous images.



- Construct a “local map” by matching up a series of camera images.

Can't Match in Camera Space

- We can't match up shapes from one image to the next if the shapes are in camera coordinates. Every time the head moves, the coordinates of the shapes in the camera image change.
- Solution: switch to a body-centered reference frame.
- If we keep the body stationary and only move the head, the coordinates of objects won't change (much) in the body reference frame.



Planar World Assumption

- How do we convert from camera-centered coordinates to body-centered coordinates?
- Need to know the camera pose: can get that from the kinematics system.
- Unfortunately, that's not enough.
- Add a planar world assumption: objects lie in the plane. The robot is standing on that plane.
- Now we can get object coordinates in the body frame.

Shape Spaces

- camShS = camera space
- groundShS = camera shapes projected to ground plane
- localShS = body-centered (egocentric space);
  constructed by matching and importing shapes
  from groundShS across multiple images
- worldShS = world space (allocentric space);
  constructed by matching and importing shapes
  from localShS
- The robot is explicitly represented in worldShS

MapBuilderNode

- MapBuilderNode takes a constructor argument specifying the kind of map to build.

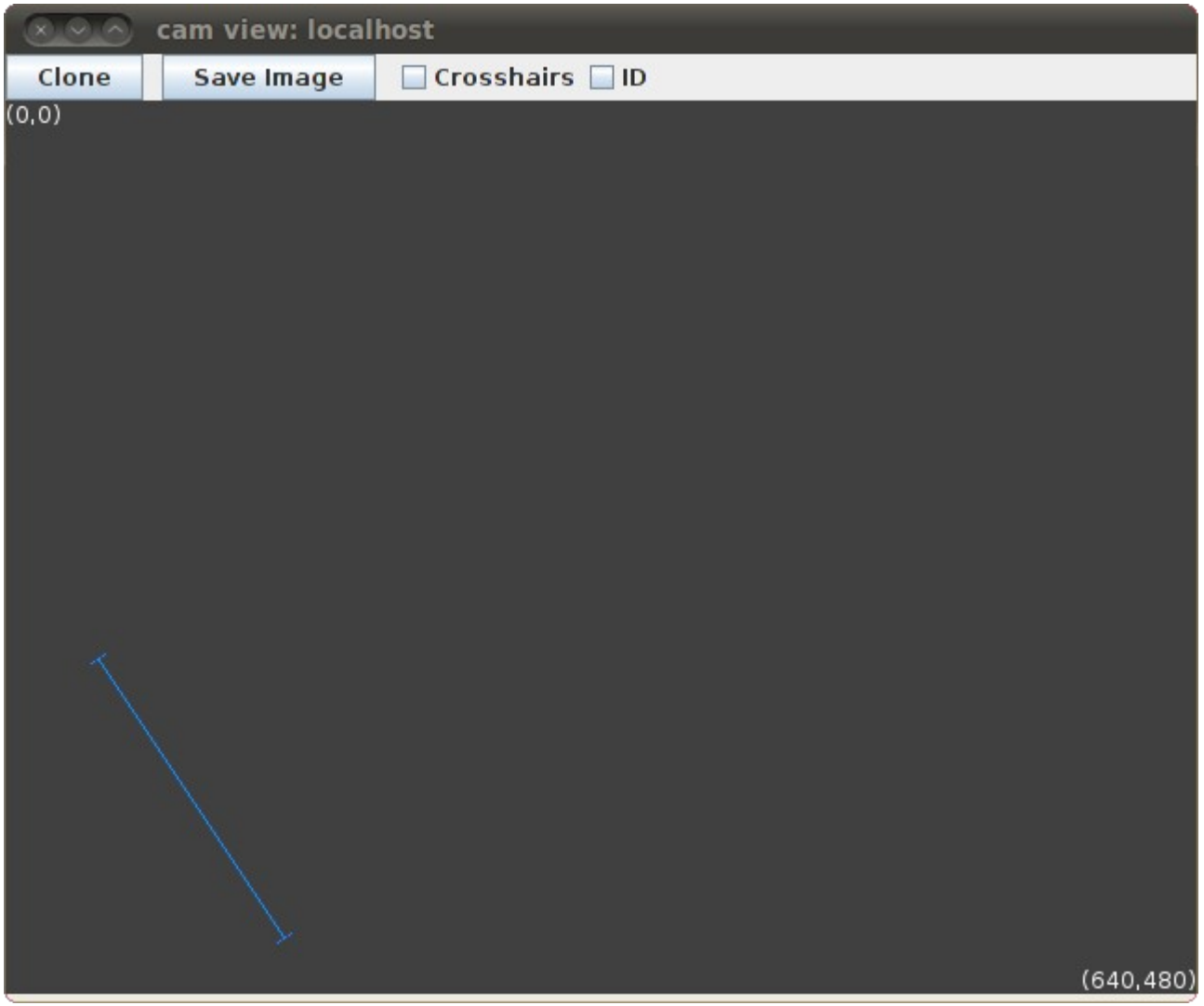
- Types of maps:
  - cameraMap (default)
  - localMap
  - worldMap

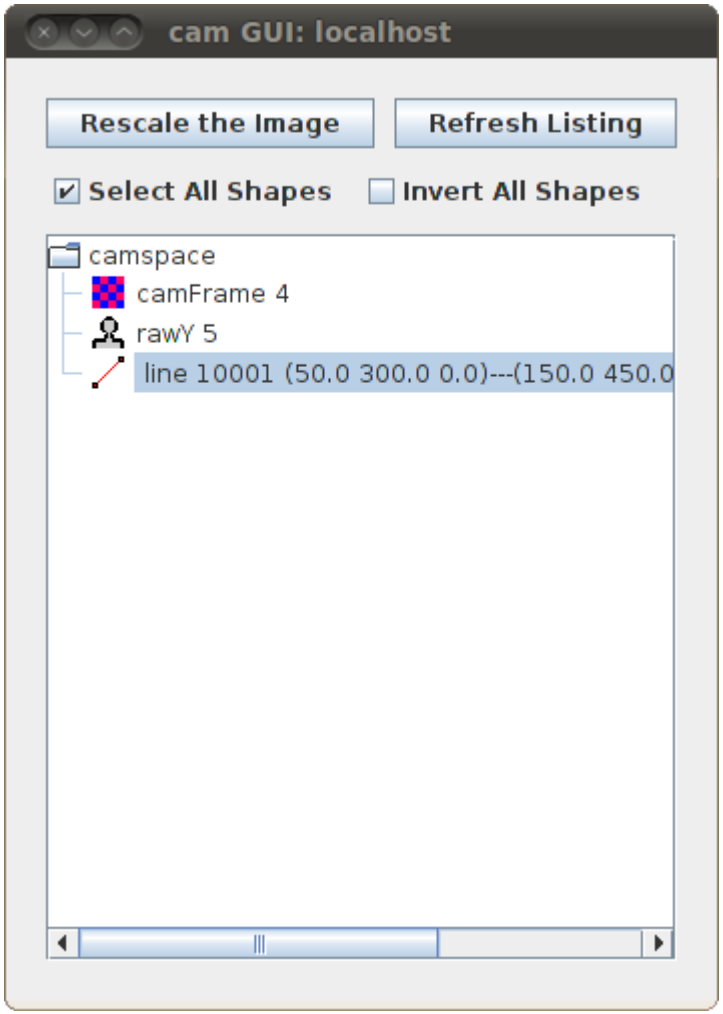
- If we ask for a localMap, the MapBuilder will project shapes from camera space to ground space and then import them into the local map.

Projecting To Ground

```
$nodeclass Example : VisualRoutinesStateNode {

  // Construct line in camera space; MapBuilder projects to local space

  $nodeclass BuildIt : VisualRoutinesStateNode : doStart {
     NEW_SHAPE(line, LineData, new LineData(camShS, Point(50,300),
                                                    Point(150,450)));
     line->setColor("blue");
  }

  $nodeclass ProjectIt :
        MapBuilderNode(MapBuilderRequest::localMap) : doStart {
     mapreq.clearCamera = false;
  }

  $setupmachine{
    BuildIt =N=> ProjectIt =C=> SpeechNode("Done")
  }

}
```

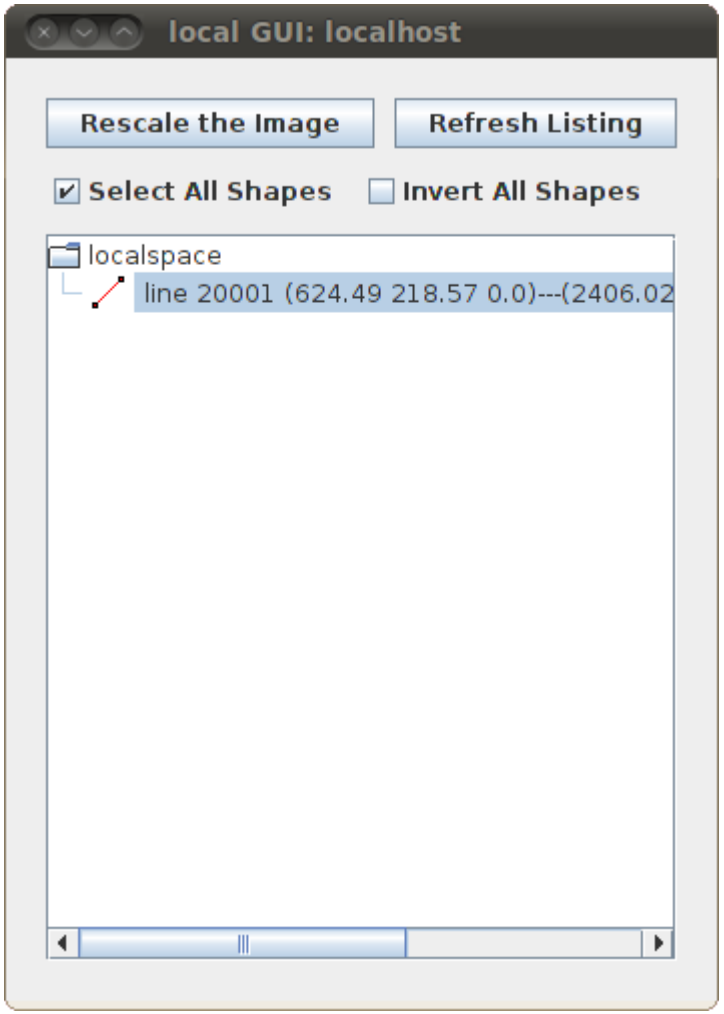
Camera Space





Local Space


local view: localhost
Clone
Save Image
ID
(2887,1458)
(0,0)


local GUI: localhost
Rescale the Image
Refresh Listing
Select All Shapes
Invert All Shapes
localspace
line 20001 (624.49 218.57 0.0)---(2406.02

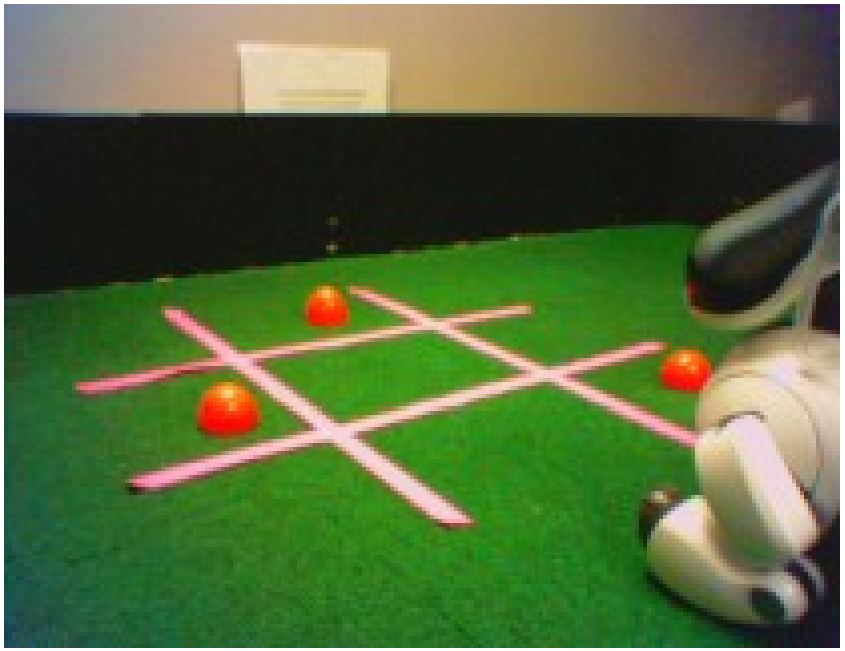
pursueShapes

- By default, the MapBuilder uses one image: whatever view the camera is currently providing.
- We can ask it to move the camera around and construct a local view from multiple images.
- Set pursueShapes = true to give the MapBuilder permission to move the camera..

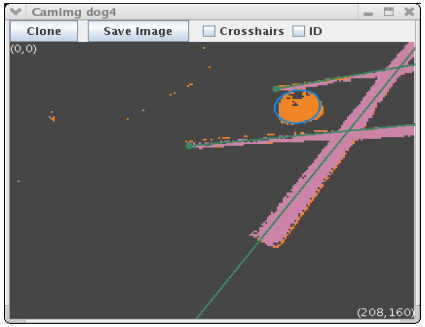
```
$nodeclass FindLines :
        MapBuilderNode(MapBuilderRequest::localMap) : doStart {
    mapreq.addObjectColor(lineDataType,”blue”);
    mapreq.pursueShapes = true;
}
```

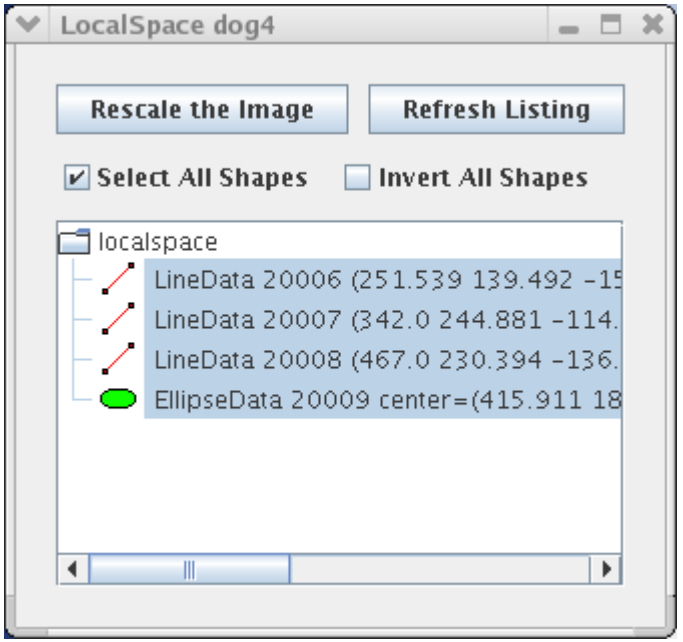
Invoking The Map Builder

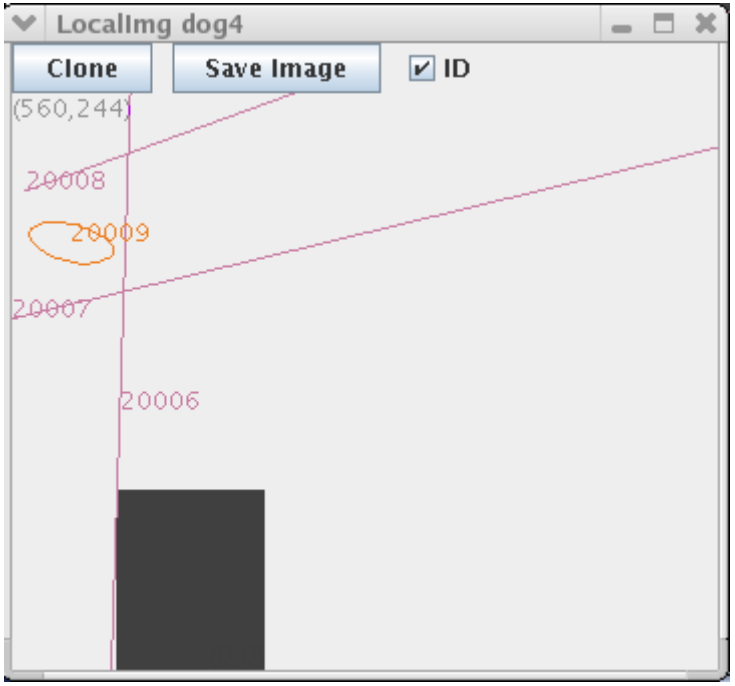
- Let's map the tic-tac-toe board:



Frame 1

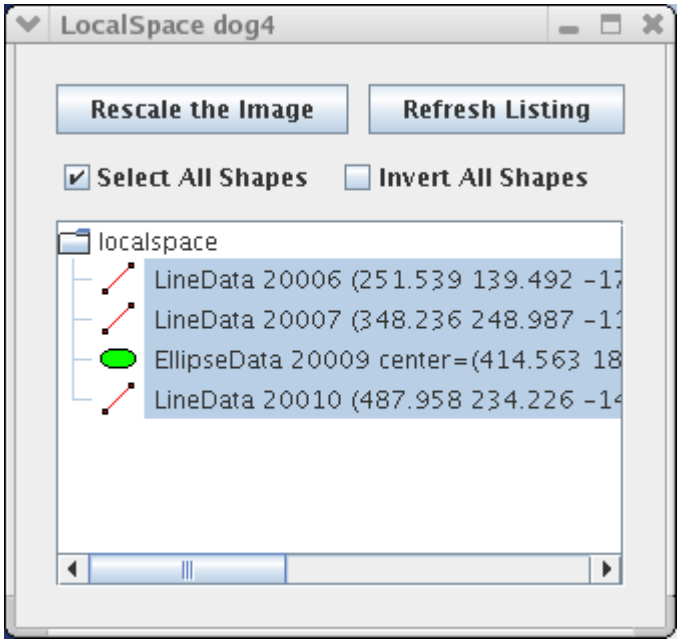

CamImg dog4
Clone
Save Image
Crosshairs
ID
(0,0)
(208,160)

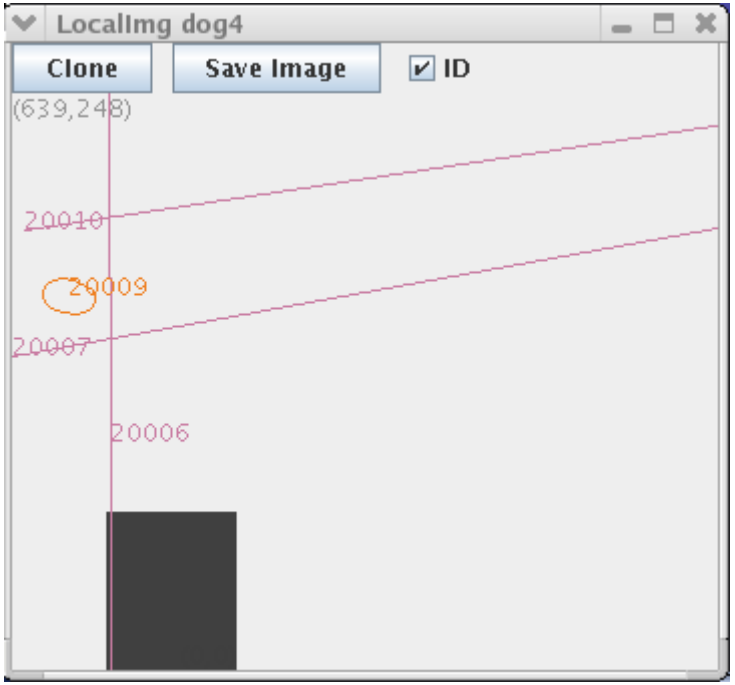

LocalSpace dog4
Rescale the Image
Refresh Listing
Select All Shapes
Invert All Shapes
localspace
LineData 20006 (251.539 139.492 -15
LineData 20007 (342.0 244.881 -114.
LineData 20008 (467.0 230.394 -136.
EllipseData 20009 center=(415.911 18


LocalImg dog4
Clone
Save Image
ID
(560,244)
20008
20009
20007
20006

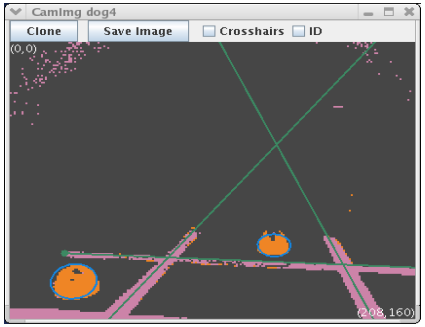
Frame 2

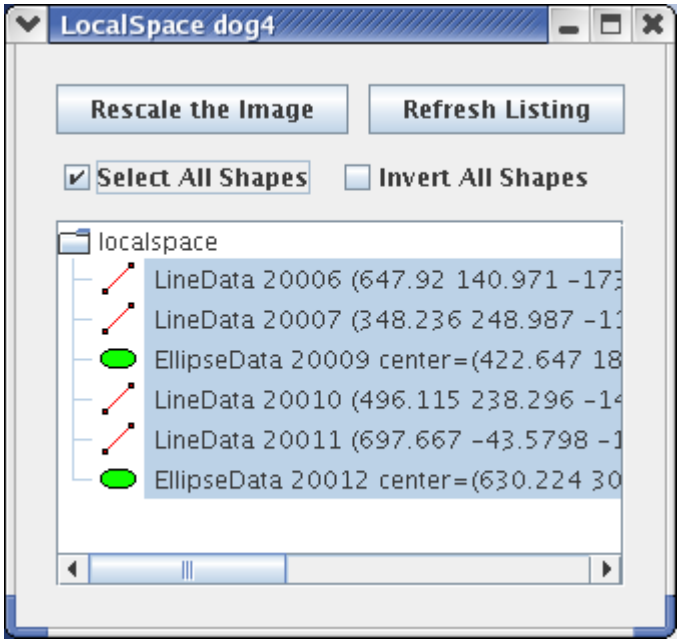

CamImg dog4
Clone
Save Image
Crosshairs
ID
(0,0)
(208,160)

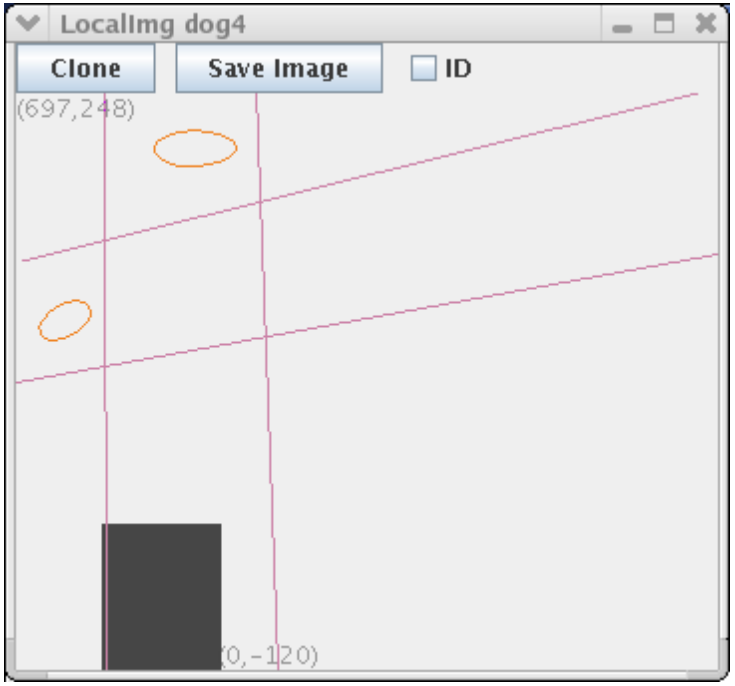

LocalSpace dog4
Rescale the Image
Refresh Listing
Select All Shapes
Invert All Shapes
localspace
LineData 20006 (251.539 139.492 -17
LineData 20007 (348.236 248.987 -11
EllipseData 20009 center=(414.563 18
LineData 20010 (487.958 234.226 -14


LocalImg dog4
Clone
Save Image
ID
(639,248)
20010
20009
20007
20006

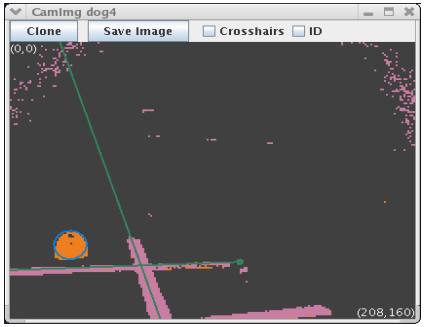
Frame 3

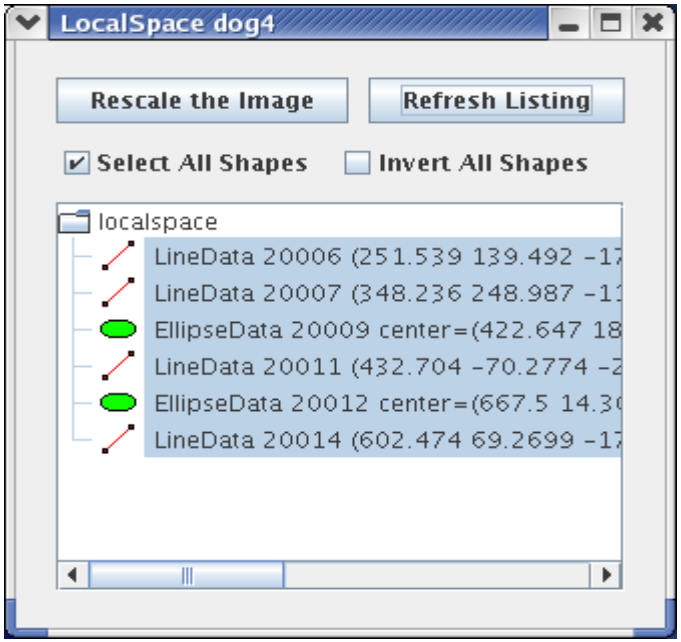

CamImg dog4
Clone
Save Image
Crosshairs
ID
(0,0)
(208,160)

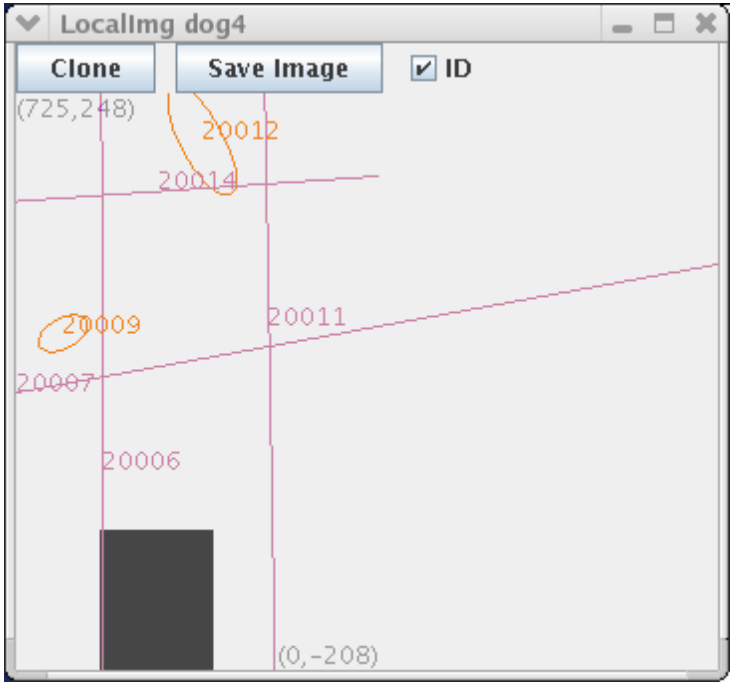

LocalSpace dog4
Rescale the Image
Refresh Listing
Select All Shapes
Invert All Shapes
localspace
LineData 20006 (647.92 140.971 -173
LineData 20007 (348.236 248.987 -1
EllipseData 20009 center=(422.647 18
LineData 20010 (496.115 238.296 -1
LineData 20011 (697.667 -43.5798 -1
EllipseData 20012 center=(630.224 30


LocalImg dog4
Clone
Save Image
ID
(697,248)
(0,-120)

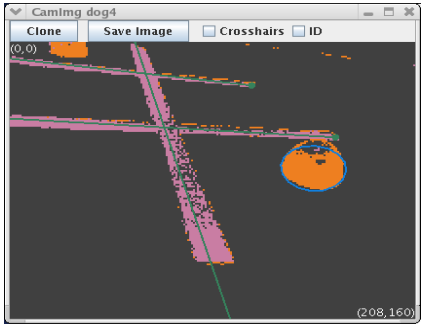
Frame 4

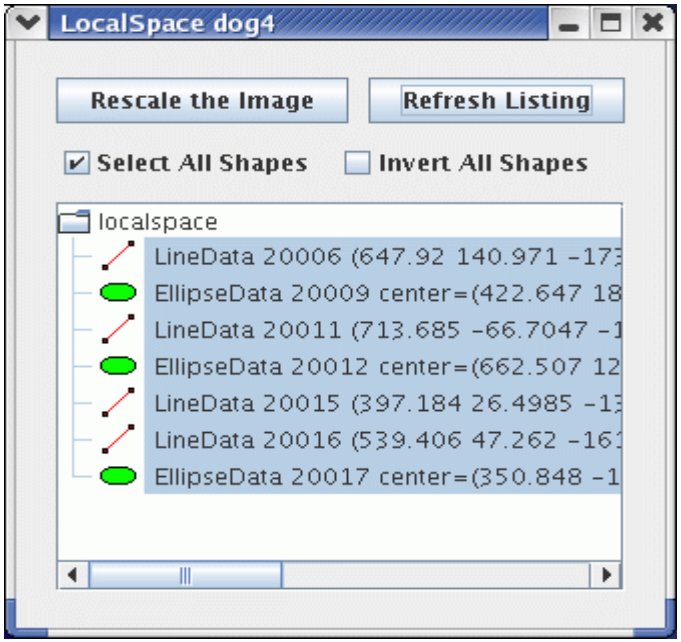

CamImg dog4
Clone
Save Image
Crosshairs
ID
(0,0)
(208,160)

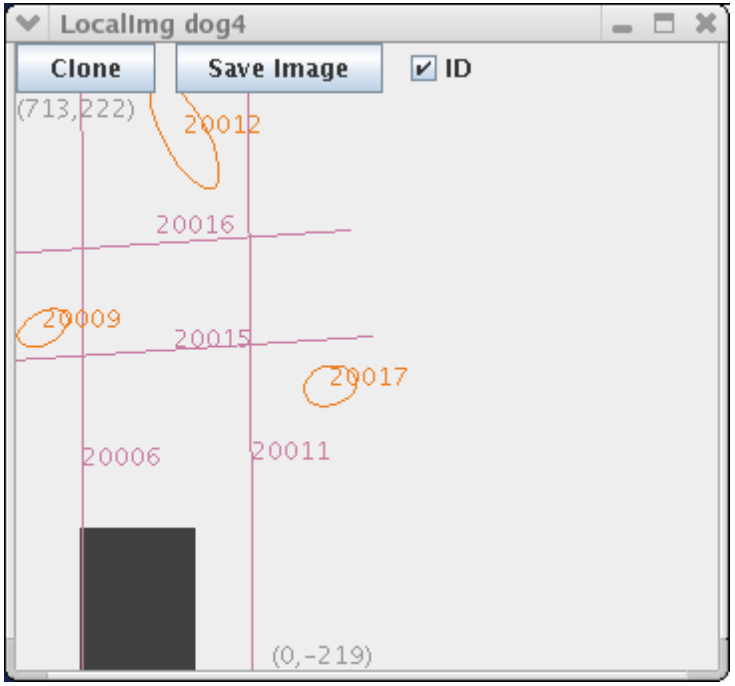

LocalSpace dog4
Rescale the Image
Refresh Listing
Select All Shapes
Invert All Shapes
localspace
LineData 20006 (251.539 139.492 -17
LineData 20007 (348.236 248.987 -11
EllipseData 20009 center=(422.647 18
LineData 20011 (432.704 -70.2774 -2
EllipseData 20012 center=(667.5 14.3
LineData 20014 (602.474 69.2699 -17


LocalImg dog4
Clone
Save Image
ID
(725,248)
20012
20014
20009
20011
20007
20006
(0,-208)

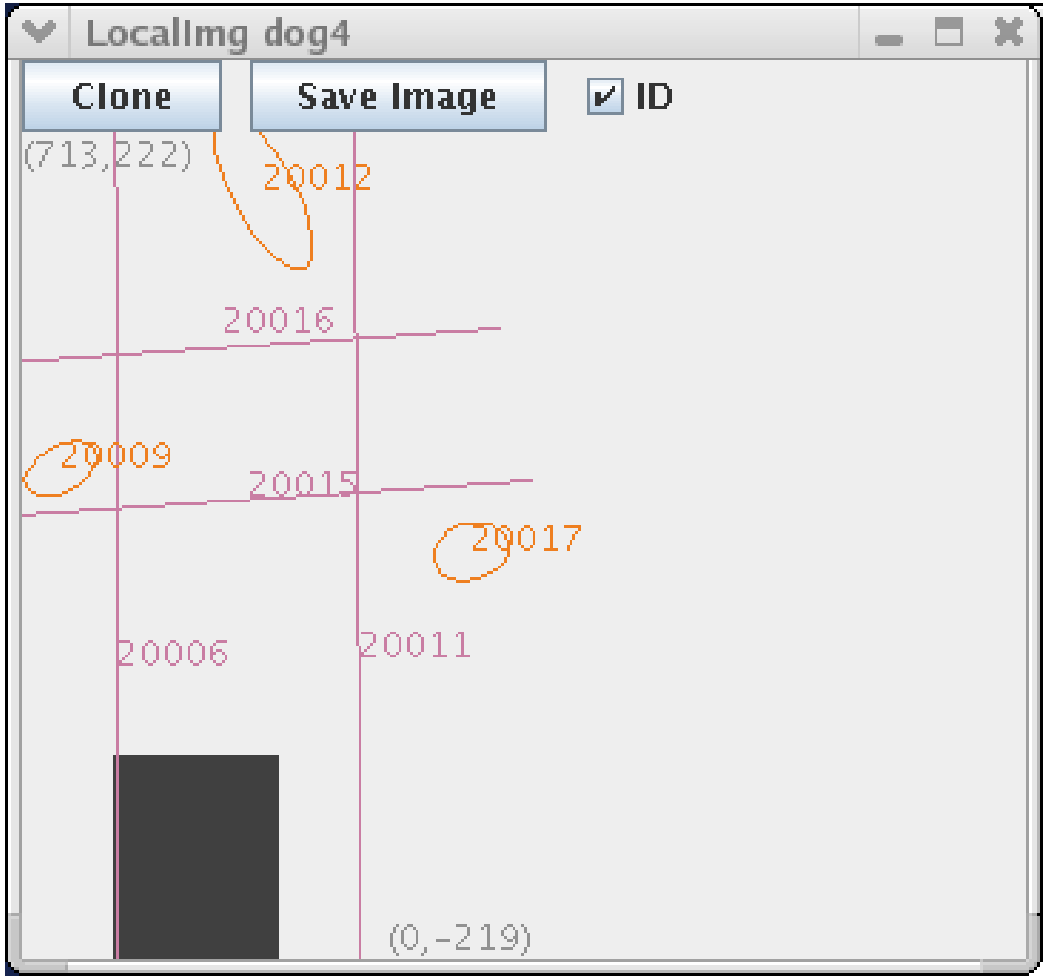
Frame 5

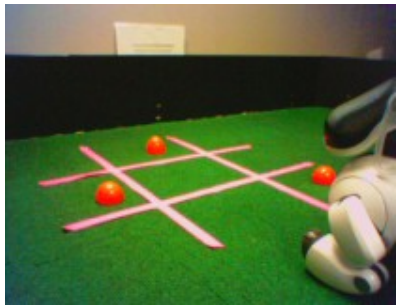

CamImg dog4
Clone
Save Image
Crosshairs
ID
(0,0)
(208,160)


LocalSpace dog4
Rescale the Image
Refresh Listing
Select All Shapes
Invert All Shapes
localspace
LineData 20006 (647.92 140.971 -17
EllipseData 20009 center=(422.647 18
LineData 20011 (713.685 -66.7047 -1
EllipseData 20012 center=(662.507 12
LineData 20015 (397.184 26.4985 -1
LineData 20016 (539.406 47.262 -16
EllipseData 20017 center=(350.848 -1


LocalImg dog4
Clone
Save Image
ID
(713,222)
20012
20016
20009
20015
20017
20006
20011
(0,-219)

Final Local Map





Where to Look?

- Start with the shapes visible in the camera frame.
- Move the camera to fixate each shape: get a better look.
- If a line runs off the edge of the camera frame, move the camera to try to find the line's endpoints.
  - If the head can't rotate any further, give up on that endpoint.
- If an object is partially cut off by the camera frame, don't add it to the map because we don't know its true shape.
  - Move the camera to bring the object into view.

Shape Matching Algorithm

- Shape type and color must match exactly.
- Coordinates must be a reasonably close match for points, blobs, and ellipses.
- Lines are special, because endpoints may be invalid:
  - If endpoints are valid, coordinates should match.
  - If invalid in local map but valid in ground space, update the local map to reflect the true endpoint location.
- Coordinates are updated by weighted averaging.

Noise Removal

- Noise in the image can cause spurious shapes. A long line might appear as 2 short lines separated by a gap, or a noisy region might appear as a short line.
- Assign a confidence value to each shape in local map.
- Each time a shape is seen: increase its confidence.
- If a shape *should* be seen but is not, decrease its confidence.
- Delete shapes with negative confidence.

MapBuilderRequest Parameters

- RequestType
  - cameraMap
  - groundMap
  - localMap
  - worldMap
- Shape parameters:
  - objectColors
  - occluderColors
  - maxDist
  - minBlobArea
  - aprilTagFamily
  - markerTypes
- Utility functions:
  - clearCamera, clearLocal, clearWorld
  - RawY
- Lookout control:
  - motionSettleTime
  - numSamples
  - sampleInterval
  - pursueShapes
  - searchArea
  - doScan, dTheta
  - manualHeadMotion

World Maps

- worldShS (the world shape space) holds the robot's world map for navigation and localization.
- Examples of pre-specified world maps:
  - Tic-tac-toe board of known size.
  - Maze with known layout.
- Dynamically constructed world maps:
  - Initialize from what's locally visible (localShS).
  - As the robot moves, extend the world map.
  - Transformation matrix maps local to world coordinates.
  - SLAM: Simultaneous Localization and Matching
    Not currentlly implemented; will get there some day.
- Updating the world map when objects move?