The Map Builder

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

Carnegie Mellon
Spring 2010

Horizontal Field of View

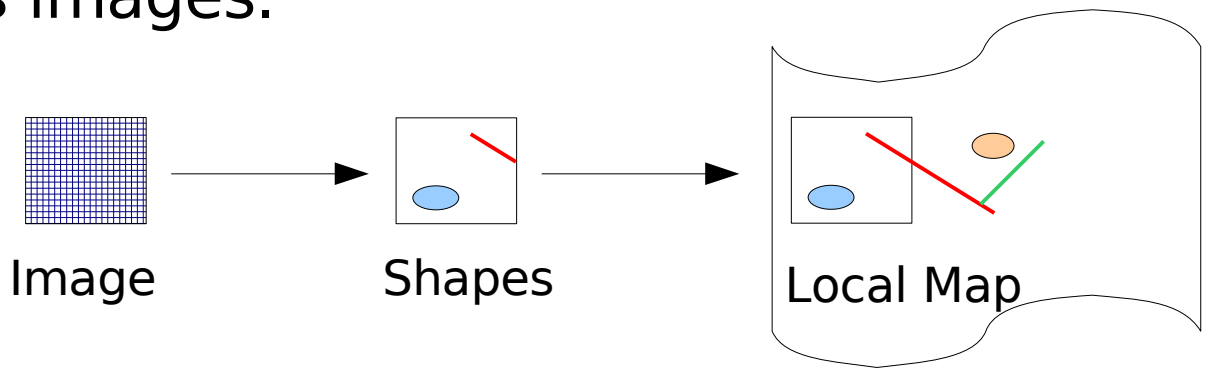
Rat: 300 deg.

Human: 200 deg.

Typical robot: 60 deg.

Seeing A Bigger Picture

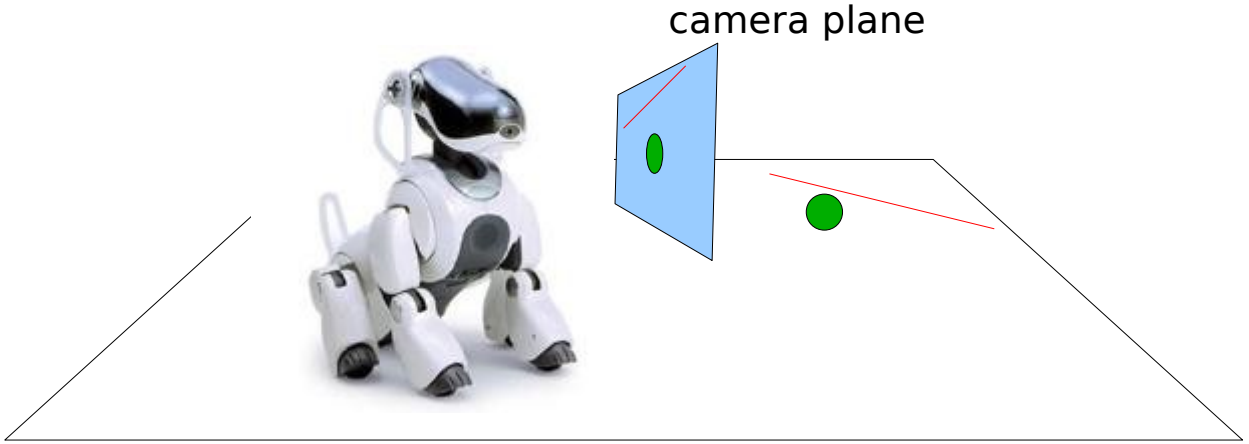
- How can we assemble an accurate view of the robot's surroundings from a series of narrow camera frames?
- First, convert each image to symbolic form: shapes.
- Then, match the shapes in one image against the shapes in previous images.



- 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

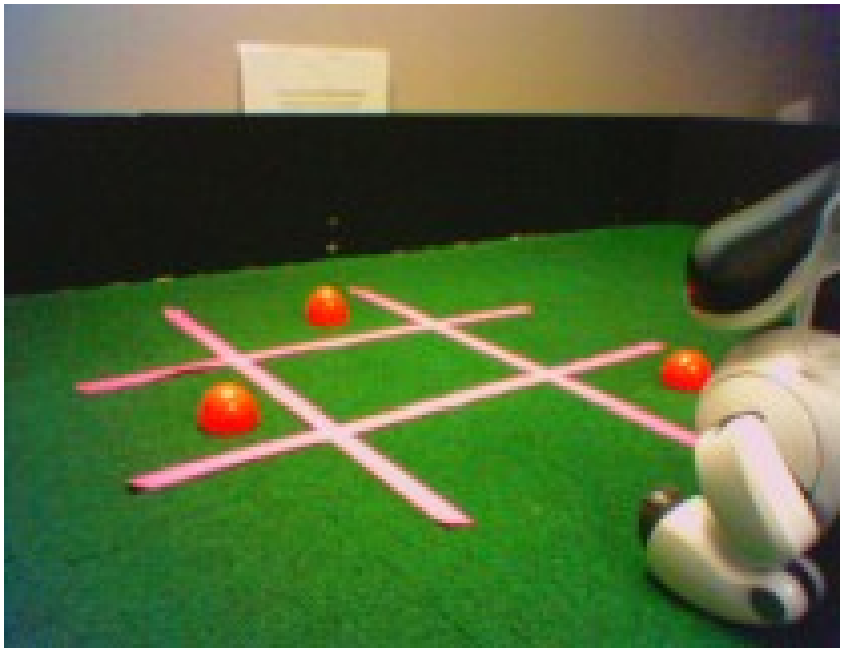
- How do we convert from camera-centered coordinates to body-centered coordinates?
- Need to know the camera pose: can get that from the kinematics system.
- Unfortunately, that's not enough.
- Add a planar world assumption: objects lie in the plane. The robot is standing on that plane.
- Now we can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

Shape Spaces

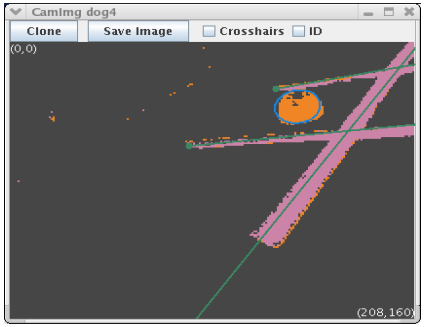
- camShS = camera space
- groundShS = camera shapes projected to ground plane
- localShS = body-centered (egocentric space);
  constructed by matching and importing shapes
  from groundShS across multiple images
- worldShS = world space (allocentric space);
  constructed by matching and importing shapes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

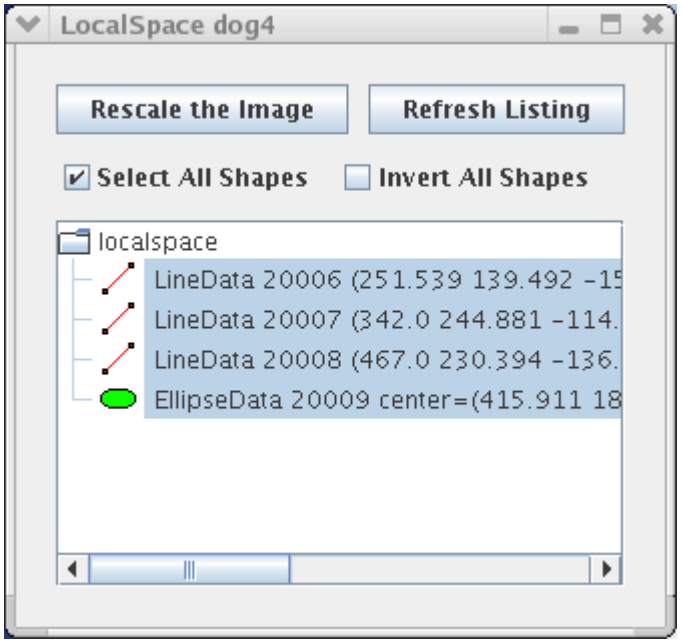
Invoking The Map Builder

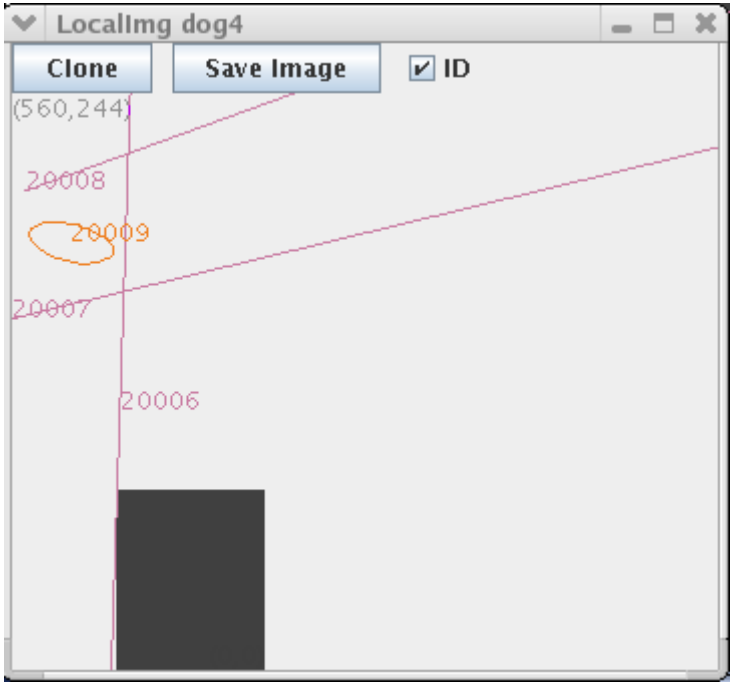
- Let's map the tic-tac-toe board:



Frame 1

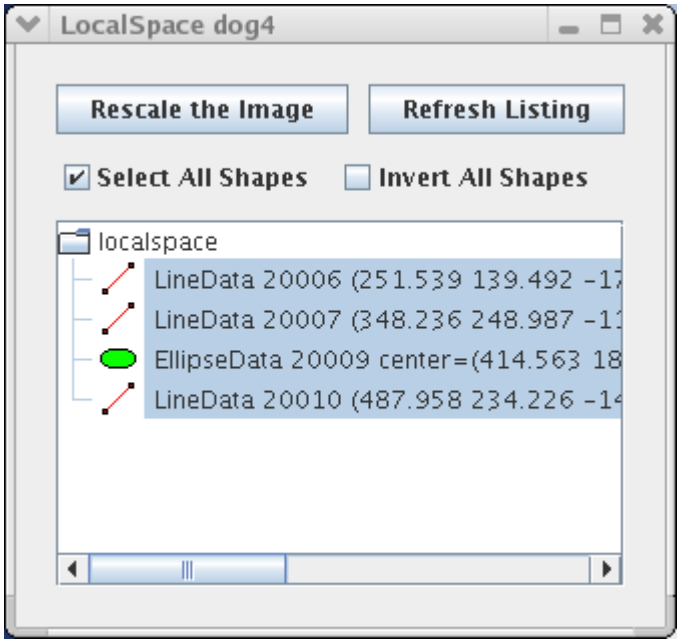

CamImg dog4
Clone
Save Image
Crosshairs
ID
(0,0)
(208,160)

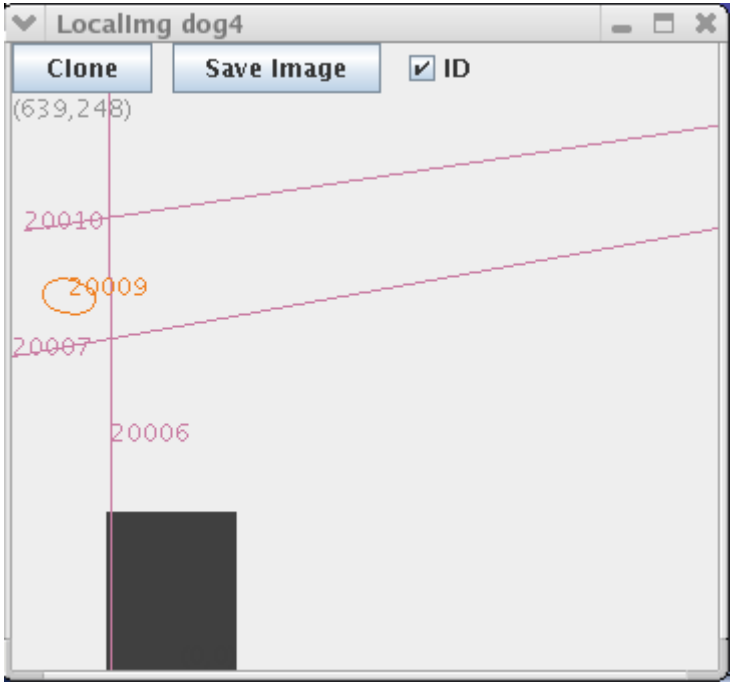

LocalSpace dog4
Rescale the Image
Refresh Listing
Select All Shapes
Invert All Shapes
localspace
LineData 20006 (251.539 139.492 -15
LineData 20007 (342.0 244.881 -114.
LineData 20008 (467.0 230.394 -136.
EllipseData 20009 center=(415.911 18


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

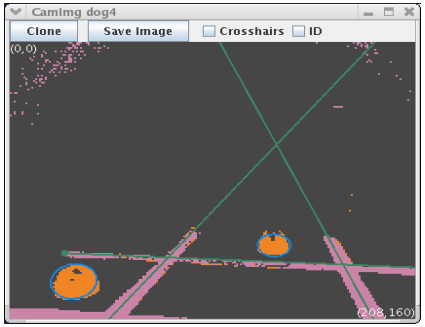
Frame 2

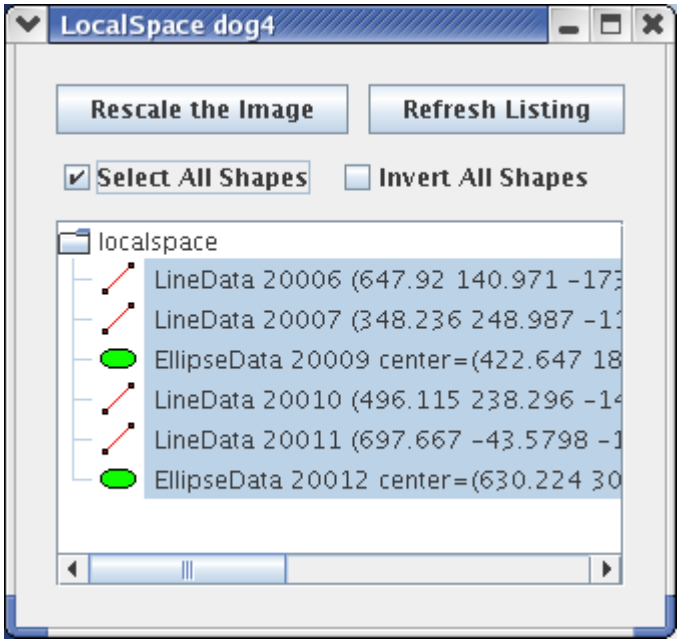

CamImg dog4
Clone
Save Image
Crosshairs
ID
(0,0)
(208,160)

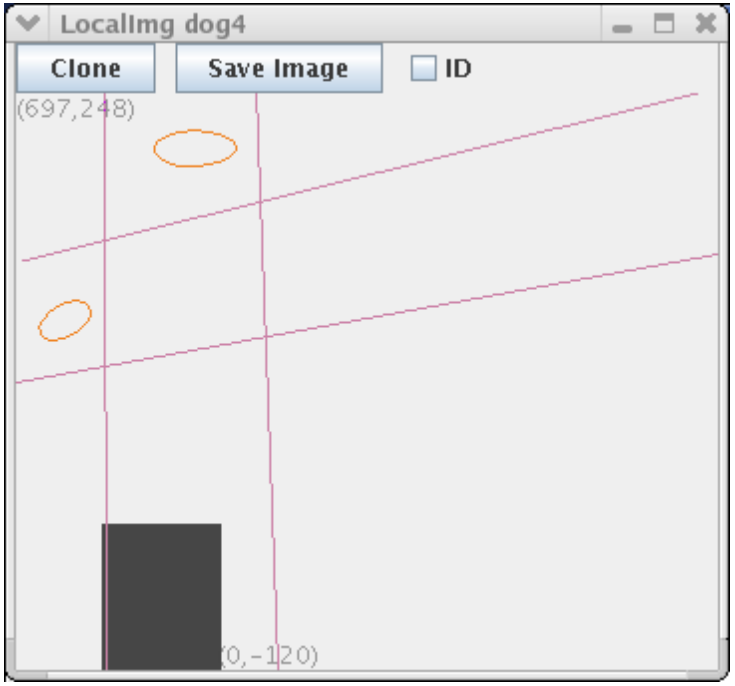

LocalSpace dog4
Rescale the Image
Refresh Listing
Select All Shapes
Invert All Shapes
localspace
LineData 20006 (251.539 139.492 -17
LineData 20007 (348.236 248.987 -11
EllipseData 20009 center=(414.563 18
LineData 20010 (487.958 234.226 -14


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

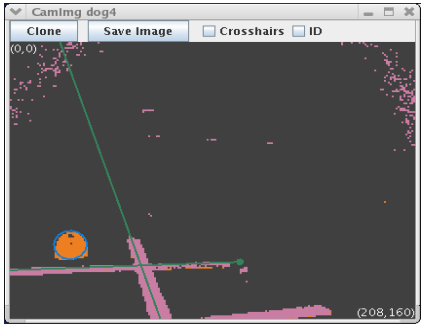
Frame 3

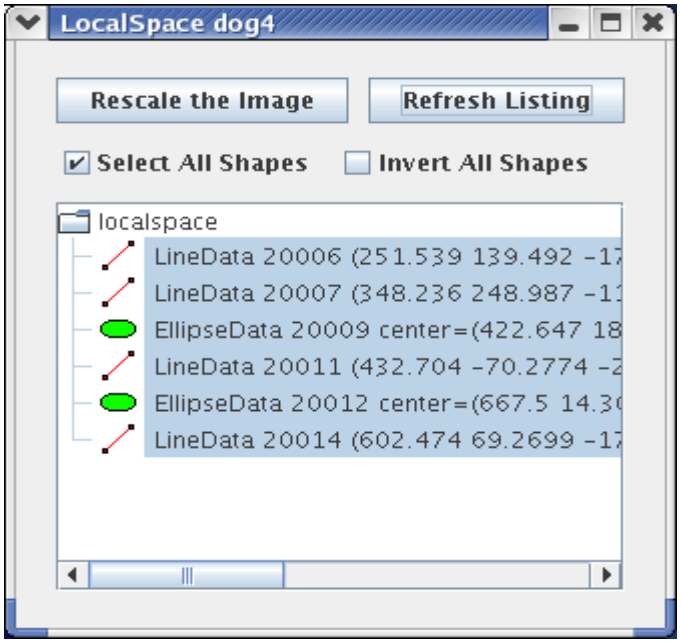

CamImg dog4
Clone
Save Image
Crosshairs
ID
(0,0)
(208,160)

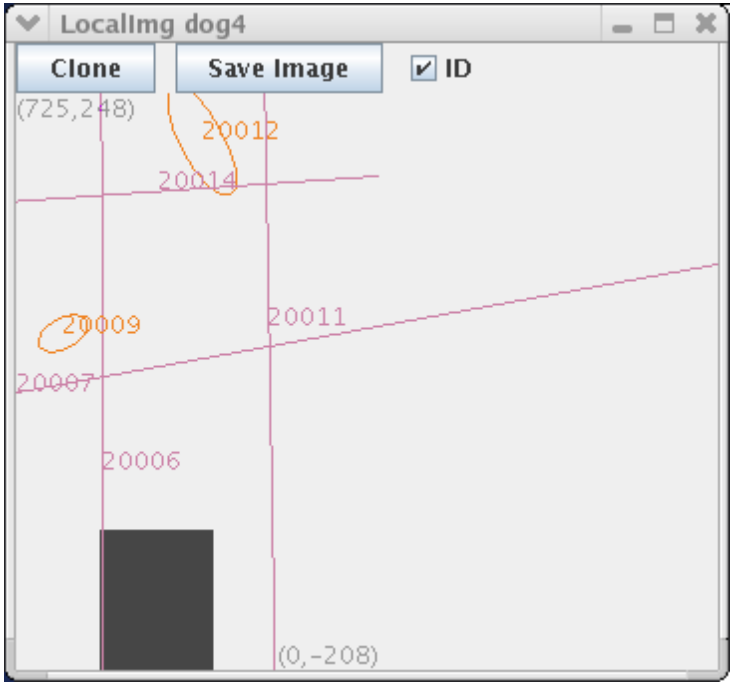

LocalSpace dog4
Rescale the Image
Refresh Listing
Select All Shapes
Invert All Shapes
localspace
LineData 20006 (647.92 140.971 -173
LineData 20007 (348.236 248.987 -1
EllipseData 20009 center=(422.647 18
LineData 20010 (496.115 238.296 -1
LineData 20011 (697.667 -43.5798 -1
EllipseData 20012 center=(630.224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


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

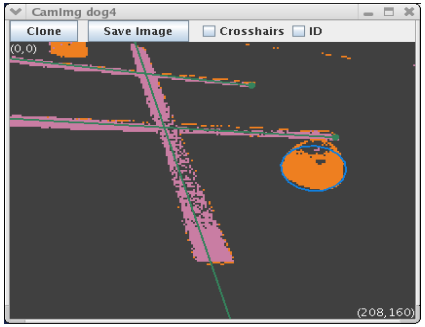
Frame 4

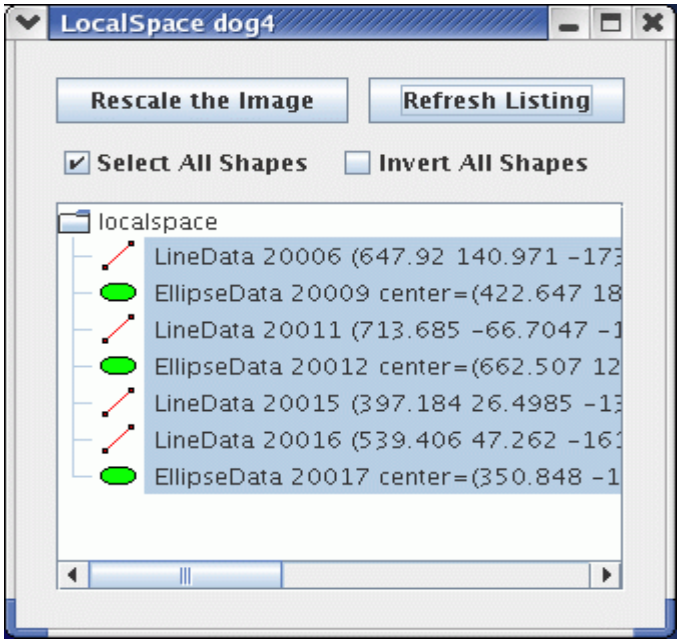

CamImg dog4
Clone
Save Image
Crosshairs
ID
(0,0)
(208,160)

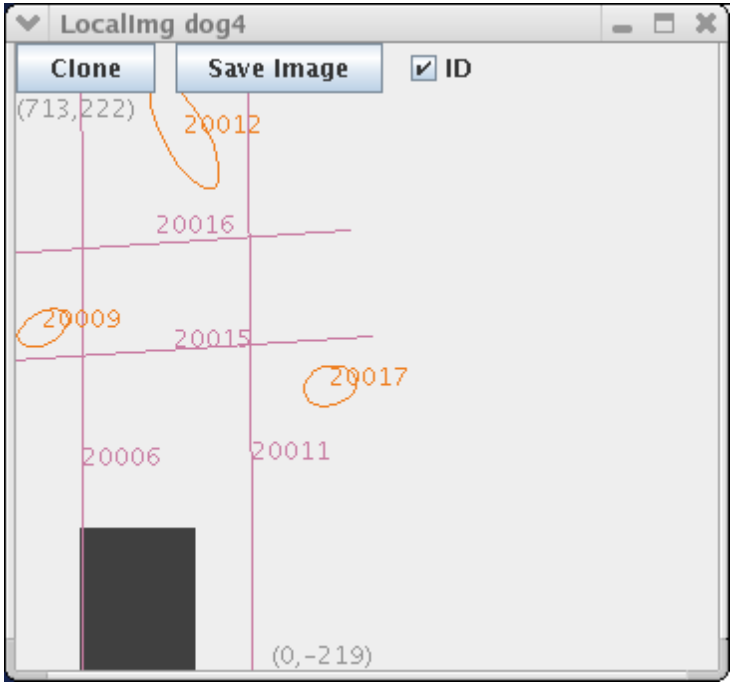

LocalSpace dog4
Rescale the Image
Refresh Listing
Select All Shapes
Invert All Shapes
localspace
LineData 20006 (251.539 139.492 -17
LineData 20007 (348.236 248.987 -11
EllipseData 20009 center=(422.647 18
LineData 20011 (432.704 -70.2774 -2
EllipseData 20012 center=(667.5 14.3
LineData 20014 (602.474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


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

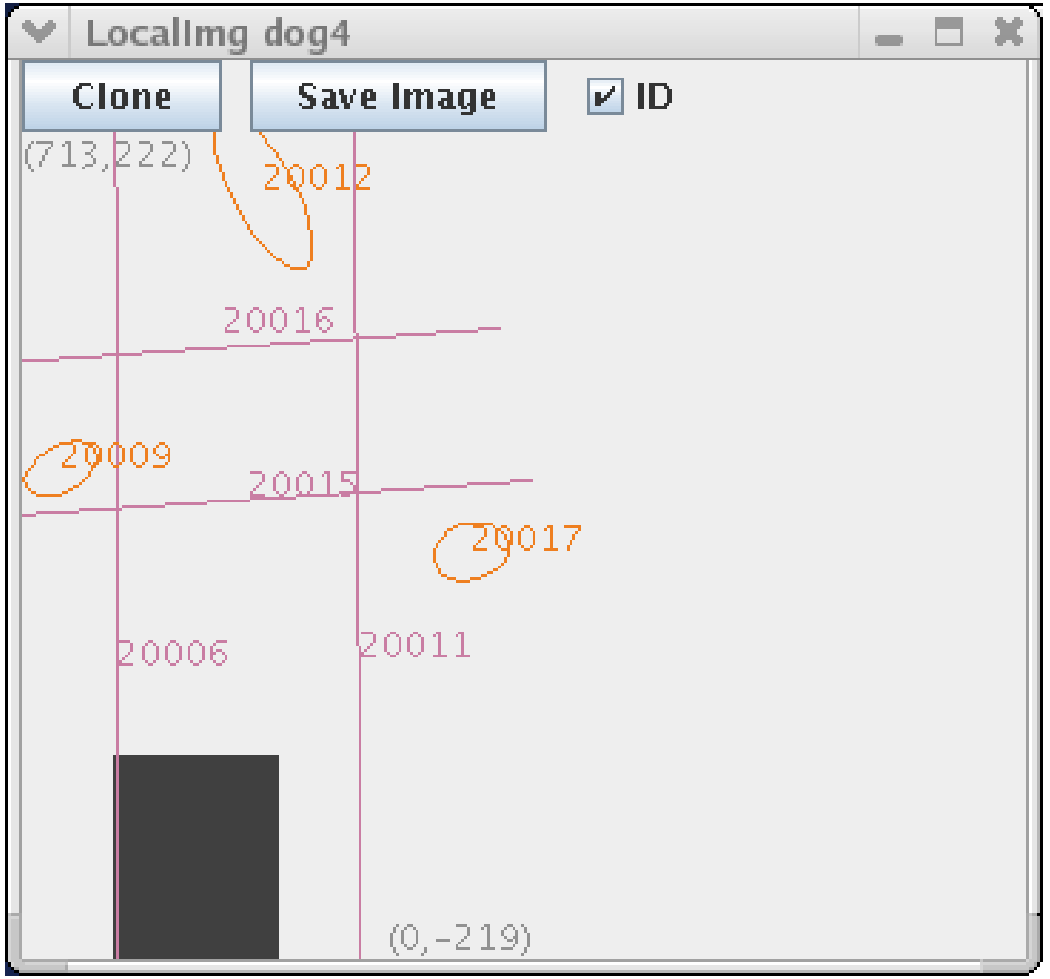
Frame 5

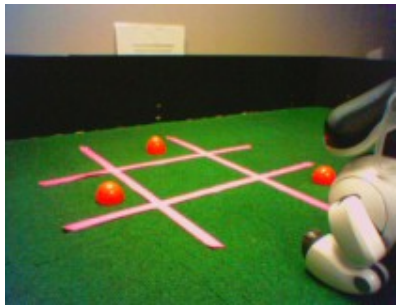

CamImg dog4
Clone
Save Image
Crosshairs
ID
(0,0)
(208,160)


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





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.

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.

MapBuilderRequest

- You invoke the Map Builder by creating an instance of a MapBuilderRequest object.
- You can use a MapBuilderNode to create the request and submit it to the Map Builder for you.
- Fill in the fields of the member variable *mapreq* to indicate what you want the MapBuilder to do, e.g., find pink lines and project them to local space.

```
#shortnodeclass MakeRequest : \
        MapBuilderNode($,MapBuilderRequest::localMap) : DoStart
    mapreq.addObjectColor(lineDataType,”pink”);
```

- At the completion of the DoStart, the request will be submitted to the MapBuilder.

Examine the Results with Another VisualRoutinesStateNode

- The results left by the MapBuilder in camShS or localShS can be examined by any subsequent state node that inherits from VisualRoutinesStateNode.
- Running the behavior again will clear out the shape spaces before processing the next request. (This can be overridden by setting mapreq.clearShapes=false.)

```
#nodeclass Report : VisualRoutinesStateNode : DoStart

    cout << "MapBuilder found " << localShS.allShapes().size()
         << " shapes." << endl;

#endnodeclass
```

Programming the MapBuilder

- Use a MapBuilderNode to submit a MapBuilder request.
- Use a =MAP=> transition to detect request completion.

```
#nodeclass LocalMapDemo : VisualRoutinesStateNode

  #shortnodeclass MakeRequest : \
        MapBuilderNode($,MapBuilderRequest::localMap) : DoStart
    mapreq.addObjectColor(lineDataType,"pink");

  #shortnodeclass Report : VisualRoutinesStateNode : DoStart
    cout << "Saw " << localShS.allShapes().size() << " shapes" << endl;

  #nodemethod setup
    #statemachine
      startnode: MakeRequest =MAP=> Report
    #endstatemachine
#endnodeclass

REGISTER_BEHAVIOR(LocalMapDemo);
```

Parent state machine must be a VisualRoutinesStateNode if any of its children are.

MapBuilderRequest Parameters

- RequestType
  - cameraMap
  - groundMap
  - localMap
  - worldMap
- Shape parameters:
  - objectColors
  - occluderColors
  - maxDist
  - minBlobArea
  - markerTypes
- Utility functions:
  - clearShapes
  - rawY
  - immediateRequest
- Lookout control:
  - motionSettleTime
  - numSamples
  - sampleInterval
  - pursueShapes
  - searchArea
  - doScan, dTheta
  - manualHeadMotion

Programming the MapBuilder

```
#nodeclass MakeRequest :      \
    MapBuilderNode($,MapBuilderRequest::localMap) : DoStart

  mapreq.numSamples = 5; // take mode of 5 images to filter out noise
  mapreq.maxDist = 1200;  // maximum shape distance 1200 mm
  mapreq.pursueShapes = true;

  mapreq.addObjectColor(lineDataType, “pink”);
  mapreq.addOccluderColor(lineDataType, “blue”);
  mapreq.addOccluderColor(lineDataType, “orange”);

  mapreq.addObjectColor(ellipseDataType, “blue)”;
  mapreq.addObjectColor(ellipseDataType, “orange”;

#endnodeclass
```

Sharing Among State Nodes

- NEW_SKETCH and NEW_SHAPE define local variables that go out of scope when the DoStart method returns.
- But the sketch or shape itself is still present.
- To access a sketch created by another state node, use GET_SKETCH to bind a local variable.
- Use GET_SHAPE to access shapes.

```
#shortnodeclass Step1 : VisualRoutinesStateNode : DoStart
  NEW_SKETCH(camFrame, uchar, sketchFromSeg());
  NEW_SKETCH(pink_stuff, bool, visops::colormask(camFrame, “pink”));
  NEW_SKETCH(pink_edges, bool, visops::edge(pink_stuff));

#shortnodeclass Step2 : VisualRoutinesStateNode : DoStart
  GET_SKETCH(pink_edges, bool, camSkS);
  NEW_SKETCH(outline, bool, visops::neighborSum(pink_edges) > 0);
```

MapBuilder Clears camSkS

- The MapBuilder clears camSkS before processing a new camera image.
- If your behavior involves repeated calls to the MapBuilder, you may want to prevent a particular sketch from being lost across calls.
- To retain a sketch, use:
  ```
  mySketch->retain();
  ```
- Use GET_SKETCH to recover access to the sketch.
- To release a sketch, use:
  ```
  mySketch->retain(false);
  ```

Qualitative Spatial Reasoning

- Reading for today:
  How qualitative spatial reasoning can improve strategy game AIs
  Ken Forbus, James Mahoney, and Kevin Dill (2002)
- Uses visual routines to "reason about" maps, e.g., compute reachability, calculate paths, etc.
- Possible research topic: applying these ideas to world maps in Tekkotsu.