Shape Representations

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

Carnegie Mellon
Spring 2011
Types of Shapes

- Basic:
  - PointData, LineData, EllipseData

- Complex:
  - PolygonData, BlobData, MarkerData

- 3-D:
  - SphereData, BrickData

- Robot shape:
  - AgentData
Shapes Live in a ShapeSpace

- SketchSpace and ShapeSpace are duals:

- We'll be using camSkS and camShS: the camera spaces.
SHAPEVEC and SHAPEROOTVEC

- Often we want to work with collections of shapes.

- A “SHAPEVEC” is a vector of shapes of a specific type:

  ```
  std::vector<Shape<BlobData>>
  ```

- A “SHAPEROOTVEC” is a vector of generic shapes, useful when we mix shapes of different types:

  ```
  std::vector<ShapeRoot>
  ```

- There are macros for creating and iterating over these vectors:
  - NEW_SHAPEVEC, NEW_SHAPEROOTVEC
  - SHAPEVEC_ITERATE, SHAPEROOTVEC_ITERATE

This space is required
Vectors of Shapes

$nodeclass ShapeExample : VisualRoutinesStateNode : doStart {
    NEW_SKETCH(camFrame, uchar, sketchFromFile());

    NEW_SHAPEVEC(blob_shapes, BlobData,
                 BlobData::extractBlobs(camFrame, 100));

    if (blob_shapes.size() > 0) {
        NEW_SKETCH(blob0, bool, blob_shapes[0]->getRendering());
    }

    SHAPEVEC_ITERATE(blob_shapes, BlobData, myblob)
        cout << "Id: " << myblob->getId()
             << "  Color: " << myblob->getColor()
             << "  Area: " << myblob->getArea()
             << endl;
    END_ITERATE;
}

01/26/11 15-494 Cognitive Robotics
Some Orange and Yellow Blobs
Extracted Blob Shapes

Id: 10001  Color: [253,119,15]  Area: 2351
Id: 10002  Color: [253,119,15]  Area: 1256
Id: 10003  Color: [193,177,9]  Area: 1378
Id: 10004  Color: [193,177,9]  Area: 1065
Id: 10005  Color: [193,177,9]  Area: 705
Line Shapes

• A line has two endpoints, which can be
  – Valid or invalid (e.g., line runs out of the camera frame)
  – Active or inactive
    If both endpoints are inactive, line extends to infinity.

• Lines have several derived properties that are maintained automatically:
  – Length
  – Orientation (0 to $\pi$)
  – Normal vector ($\rho$, $\theta$)
Extracting the Lines

$nodeclass LineExample : VisualRoutinesStateNode : doStart {
  NEW_SKETCH(camFrame, uchar, sketchFromSeg());

  NEW_SKETCH(pink_stuff, bool,
              visops::colormask(camFrame,"pink"));

  NEW_SHAPEVEC(lines, LineData,
                LineData::extractLines(pink_stuff));
}

Extracted Line Shapes

- “Select All Shapes” displays everything.
- “ID” checkbox displays shape IDs.
Line EndPoints

- Lines have two endpoints: end1Pt and end2Pt
- Order is arbitrary

- Extracting endpoints:
  - `end1Pt()`, `end2Pt()` -- simple accessor functions
  - `leftPt()`, `rightPt()` -- compare X coords.
  - `topPt()`, `bottomPt()` -- compare Y coords.

- Orientation predicates:
  - `IsHorizontal` -- true if slope is < 60 degrees
  - `IsVertical` -- true if slope is > 30 degrees
  - These thresholds are user-adjustable
Extracting the Leftmost Point

```cpp
$nodeclass LineExample : VisualRoutinesStateNode : doStart {
    NEW_SKETCH(camFrame, uchar, sketchFromSeg());
    NEW_SKETCH(orange_stuff, bool,
               visops::colormask(camFrame, "orange");
    NEW_SHAPE(line, LineData,
               LineData::extractLine(orange_stuff));
    NEW_SHAPE(leftpt, PointData, line->leftPtShape());
    leftpt->setColor(rgb(0,255,0));
}
```
Extracted Point Shape

- leftpt's parent is line
- line's parent is orange_stuff: a shape whose parent is a sketch
Logical EndPoint Descriptions

- firstPt() -- if line is horizontal, returns leftPt(), else returns topPt()
- secondPt() -- similar: returns rightPt() or bottomPt()
- How do we compare two lines? Example:
  - Two lines are “close” if their first endpoints are close, and their second endpoints are also close.
  - But what about lines whose orientations straddle the critical horizontal/vertical threshold of 60 degrees?

- line1->firstPt(line2) -- returns first point of line2 based on line1's decision about horizontal/vertical
Constructing New Lines

• Use a LineData(camShS, ...) constructor to make new lines in camera space.

• Since we want to use smart pointers for shapes, the result should be fed to a Shape<LineData> constructor.
  – The NEW_SHAPES macro does this for us:
    
    ```
    NEW_SHAPES(myline, LineData, new LineData(camShS, ...));
    ```

• Can define a new line by specifying:
  – two points
  – a point plus an orientation (0 to $\pi$)
NEW_SHAPE

- NEW_SHAPE is a bit of syntactic sugar:

  ```
  NEW_SHAPE(myline, LineData,
             new LineData(camShS,pt1,pt2))
  ```

- Expands into:

  ```
  Shape<LineData> myline(new LineData(camShS,pt1,pt2));
  if ( myline.isValid() )
      myline->V("myline");      // make viewable
  ```

- Use NEW_SHAPE_N for shapes not to be viewable.
Parents and Viewable IDs

On the Robot

- **foo**
  - id: 11
  - parentId: 0

- **bar**
  - id: 17
  - parentId: 11

- **baz**
  - id: 19
  - parentId: 17

- **xam**
  - id: 23
  - parentId: 19

SketchGUI Display

- **foo**
  - id: 11

- **xam**
  - id: 23

**Not viewable**

01/26/11 15-494 Cognitive Robotics
Mixing Sketches and Shapes

• Problem: which side of an orange line has more yellow blobs?

• If all we have is a line segment, people can still interpret it as a “barrier”.

• How do we make the robot do this?
Lines as Barriers

$nodeclass LineExample : VisualRoutinesStateNode : doStart {
  NEW_SKETCH(camFrame, uchar, sketchFromSeg());
  NEW_SKETCH(orange_stuff, bool,
              visops::colormask(camFrame,"orange");)
  NEW_SKETCH(yellow_stuff, bool,
              visops::colormask(camFrame,"yellow");)

  NEW_SHAPE(boundary_line, LineData,
            LineData::extractLine(orange_stuff));

  NEW_SKETCH(topside, bool,
              visops::topHalfPlane(boundary_line));

  NEW_SKETCH(side1, bool, yellow_stuff & topside);
  NEW_SKETCH(side2, bool, yellow_stuff & ! topside);
Lines as Barriers (cont.)

NEW_SHAPEVEC(side1blobs, BlobData, 
    BlobData::extractBlobs(side1,50));
NEW_SHAPEVEC(side2blobs, BlobData, 
    BlobData::extractBlobs(side2,50));

vector<Shape<BlobData> > &winners = 
    sidelblobs.size() > side2blobs.size() ? 
    sidelblobs : side2blobs;

NEW_SKETCH(result, bool, visops::zeros(yellow_stuff));

SHAPEVEC_ITERATE(winners, BlobData, b) 
    result |= b->getRendering();
END_ITERATE;

boundary_line->setInfinite();  // for display purposes
Subtle point: bool overrides uchar in the SketchGUI, so selecting yellow_stuff allows the top yellow blob to display even though the inverted (orange) *topside* is covering its appearance in *camFrame*. (Competing bools are averaged.)
Lines As Barriers

[Image of two software interfaces]
Constructing a Perpendicular

```cpp
$nodeclass LineExample : VisualRoutinesStateNode: doStart {
  NEW_SKETCH(camFrame, uchar, sketchFromSeg());
  NEW_SKETCH(orange_stuff, bool,
              visops::colormask(camFrame,"orange"));

  NEW_SHAPE(line1, LineData,
            LineData::extractLine(orange_stuff));

  line1->leftPt().setActive(false);

  NEW_SHAPE(line2, LineData,
            new LineData(camShS,line1->rightPt(),
                         line1->getThetaNorm()));

  NEW_SKETCH(corner, bool,
              visops::seedfill(line1->getRendering() |
                              line2->getRendering(), 0));

  corner->setColor(rgb(0,255,0));
}
```
Constructing a Perpendicular

- Why isn't line2 shown as a child of line1?
Ellipses

• Used to describe circular or elliptical shapes.
• Different from blobs. Ellipse properties:
  – semi-major, semi-minor axis lengths
  – major axis orientation
• Ellipse extraction routine will ignore regions that aren't roughly elliptical in shape.
Extracting Ellipses

```
$nodeclass EllipseExample : VisualRoutinesStateNode : doStart {

    NEW_SKETCH(camFrame, uchar, sketchFromSeg());
    NEW_SKETCH(orange_stuff, bool,
                visops::colormask(camFrame,"orange");
    NEW_SKETCH(yellow_stuff, bool,
                visops::colormask(camFrame,"yellow");

    NEW_SHAPEVEC(ellipses, EllipseData,
                 EllipseData::extractEllipses(yellow_stuff));

    NEW_SHAPEVEC(ellipses2, EllipseData,
                 EllipseData::extractEllipses(orange_stuff));
}
```
Extracting Ellipses
Assignment and Copying

• Sketches: assignment is deep; copying is shallow.

  “A = 1” only makes sense for deep assignment.
  “A += B” only makes sense for deep assignment.
  So “A = B” should be deep as well.

  NEW_SKETCH(A, bool, B) does shallow copy. For deep copy, do:
  NEW_SKETCH(A, bool, visops::copy(B))

  For shallow assignment, do: A.bind(B)

• Shapes: assignment and copying are both shallow.

  Mostly we want to just pass shapes around, so shallow copy is all that's necessary.

  For deep copy, do: NEW_SHAPE(A, LineData, B->copy())

  Deep assignment is not supported.
Point vs. PointData

- Point(x,y,z) uses fmat::Column<4>.
- Operators +/-*/ == are defined on Point objects.
- EndPoint is a subclass of Point with a few extra properties: valid, active.
- LineData contains two EndPoints. EllipseData contains one Point defining its center.

- PointData is a shape representation with a Point inside.

- Why have both Point and PointData?
  - Shapes aren't allowed to nest, so you can't put a PointData inside a LineData or EllipseData.
Other Shape Types

- PolygonData can represent boundaries (like the edge of the robot's playpen) or containers.
- SphereData can be used to represent a ball in 3-D.
- BrickData will be used for blocks world tasks.
- AgentData represents the robot's position (as a Point) and orientation (as an AngTwoPi).
ShapeSpace:  
A Look Under the Hood