Human & Modern Machine Interactions

Jeffrey Minton

Topics

- Hardware/Software Platform
- Image Processing
- Natural Language Processing
- Using NLP and Image Processing to allow robotic operation

Hardware/Software Platform

- Two iterations

- First iteration
  - created to compete in Trinity College Fire Fighting Home Robot Competition

- Second Iteration
  - created to accomodate the needs for a class in inteligent robotics

First Iteration Hardware

- VEX robotics kit for chassis
- VEX Included motors for locomotion
- Sonar Rangefinders
- Web-cam
- Arduino to control motors and sensors
- Netbook to control arduino and process images

First Iteration Software

- Windows XP on netbook
- OpenCV for image processing
- C++ used for writing all robot operation software

Problems with First Iteration

- Inaccurate motor control
- Sonar signals bounce inside corners
  - provide inaccurate measurements
- Web-cam had too small of a viewing angle

Second Iteration Hardware

- Create by iRobot
- Platform built onto Create cargo bay to accommodate equipment
- Web-cam with increased viewing angle
- IR Rangefinder
  - IR light doesn't bounce like sound
- Arduino
- Net-book

Second Iteration Software

- Windows XP on net-book
- MATLAB
  - Image Acquisition Toolbox
  - Image Processing Toolbox
  - iRobot Create Toolbox
  - Arduino Toolbox
- C++ interfacing with MATLAB functions

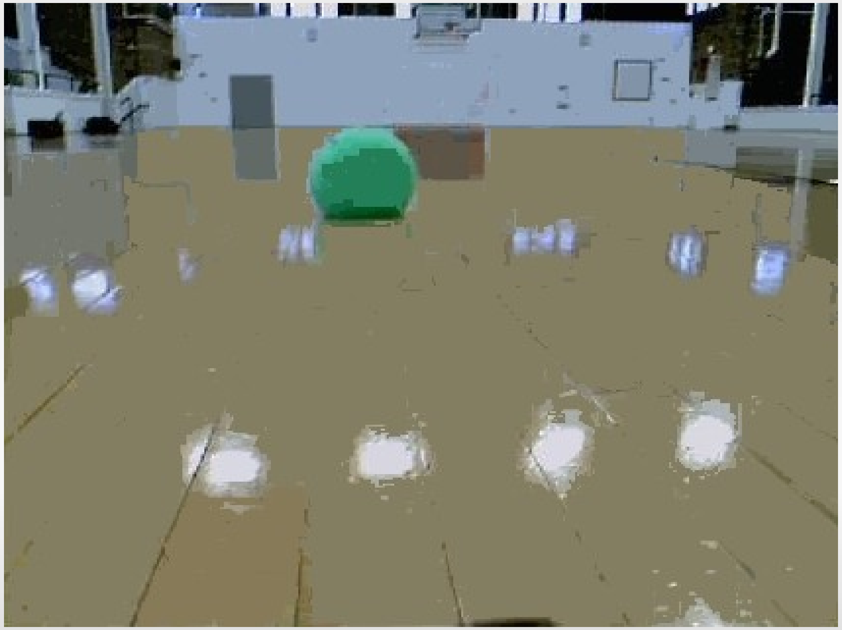
Image Processing

- “A picture is worth a thousand words”
- Need to extract discrete objects from images to identify them
- Blobbing
- K-Means Clustering

Blobbing

- Connected Component Labeling, Blob Labeling
- Compare pixels to neighbors
- Use threshold to determine if pixels are part of same blob
- Often fails to retrieve blobs that are confined to one object, or that contain an entire object

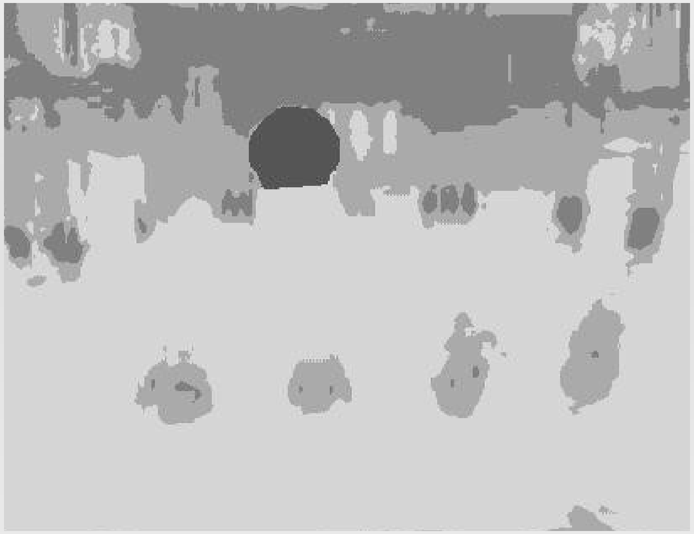
Blobbing Example



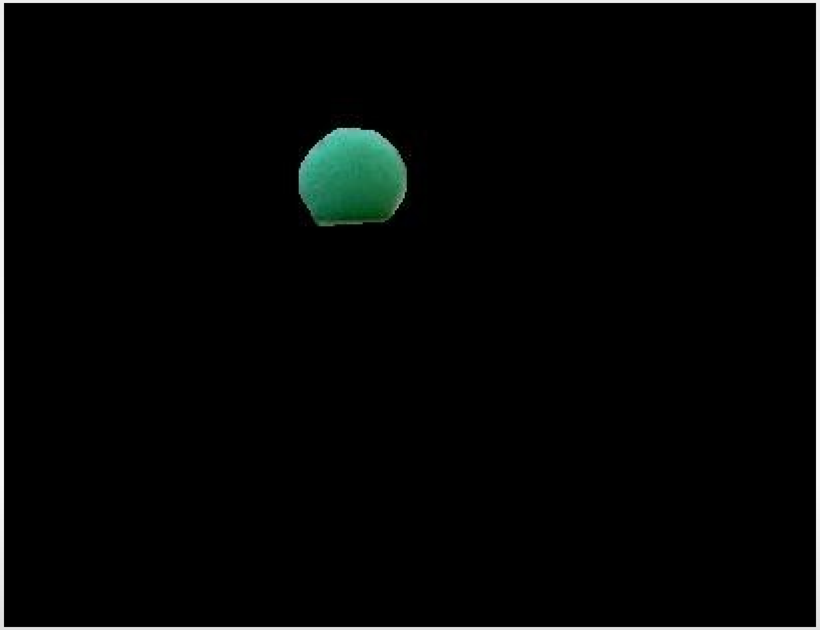
K-Means Clustering

- Cluster sets of data
  - Into user defined number of segments
  - Number of segments referred to as *K*
  - Clusters defined by MEAN of all values in cluster
- More accurate than blobbing
  - Likeness compared to entire cluster not simply adjacent pixels
  - Able to reliably retrieve clusters of discrete objects

K-Means Example



K-Means using k = 6



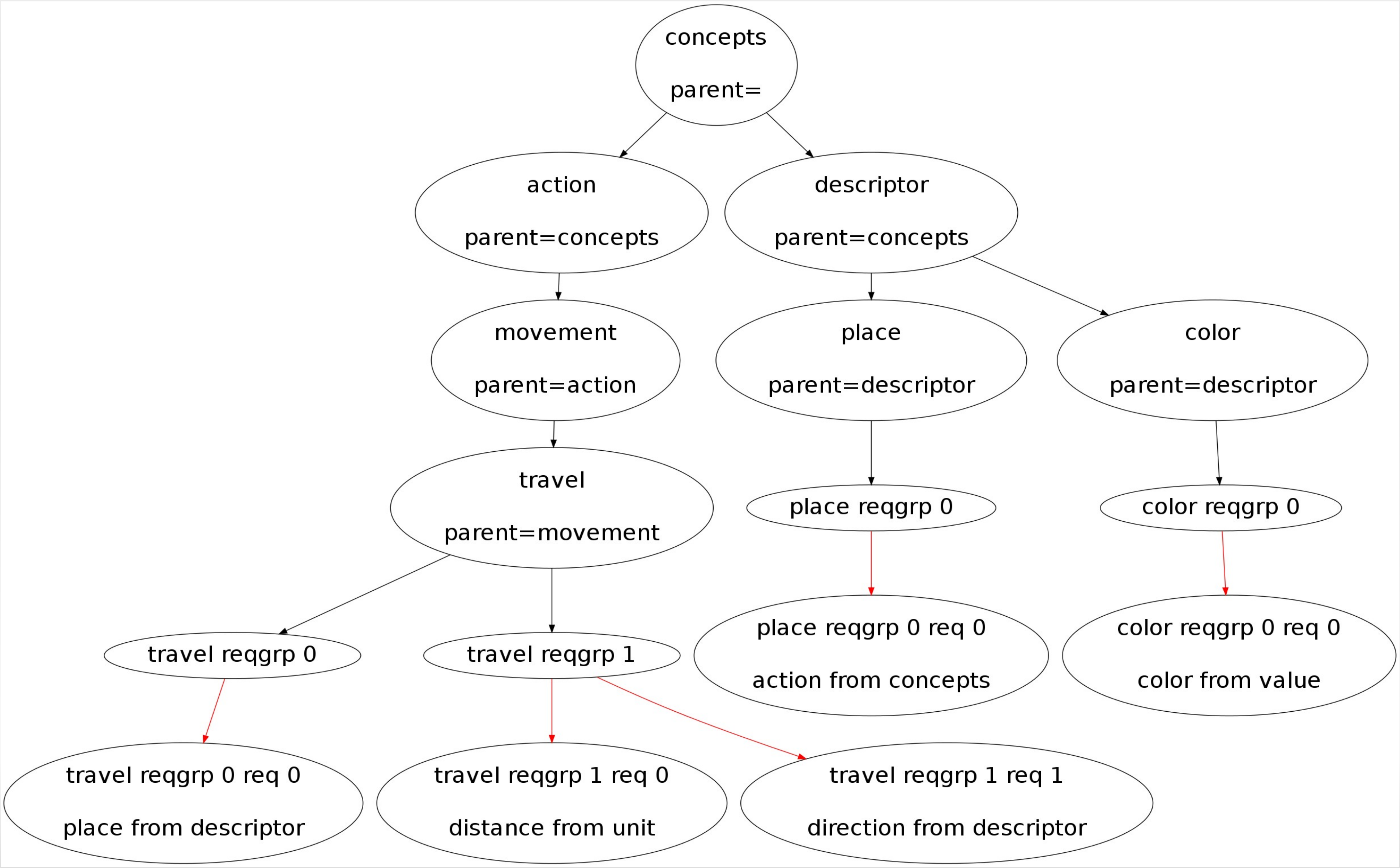
Natural Language Processing

- "Go to the green ball"
- The meanings behind words must be inferred
- "Go," conceptually can represent many things
  - Take a turn in a game
  - The Chinese strategy game
  - Travel to a location
- Determine the concept being referred to
- Conceptual Parsing

Conceptual Parsing

- Words are mapped to concepts
- Concepts
  - Rules define set of related concepts
  - One concept may have many separate rule sets

Concept Tree



References

- iRobotCreate Toolbox - Vader Laboratory at Lehigh University
- K-Means algorithm - KMeans Segmentation – MEX, http://www.mathworks.com/matlabcentral/fileexchange/27969-kmeans-segmentation-mex
- Arduino Toolbox - MATLAB Support for Arduino http://www.mathworks.com/academia/arduino-software/arduino-matlab.html

Questions?

K-Means Algorithm

```
function KMEANS(data_vecs, numSegments) returns data_idxs, centroids:
        inputs: data_vecs – vector of data to be segmented < 3 dimensions
                numSegments – the number of segments the data stored in data_vecs
                        should be separated into

        centroids ← numSegments evenly spaced values inclusivly between min of
                        data_vecs and max of data_vecs
        prev_centroids ← centroids offset by 1 /*guarantess that prev_centroids
                and centroids are not equal at start of first pass over data_vecs*/
        data_idxs ← empty 1 dimensional vector of length LENGTH(data_vecs)
        while prev_centroids ≠ centroids do:
                prev_centroids ← centroids
                dists ← empty vector with dimensions
                                LENGTH(data_vecs) by numSegments
                for j = 1 to numSegments do:
                        for k = 1 to LENGTH(data_vecs) do:
                                temp ← data_vecs[k] – centroids[j]
                                dists[j, k] ← SUM_OF_SQUARES(temp)
                for j = 1 to LENGTH(data_idxs) do:
                        data_idxs[j] ← IDX_OF_MIN(dists[j])
                for j = 1 to numSegments do:
                        centroids[j] ←
                                MEAN(data_vecs[x] for all x where data_idxs[x] == j)
        return data_idxs, centroids

function SUM_OF_SQUARES(vals) returns sum_sq:
        inputs: vals – collection of values to get sum of squares for
        returns: sum_sq – the sum of the squares of values in vals
        for j = 1 to LENGTH(vals) do:
                sq ← vals[j] ^ 2
                sum_sq ← sum_sq + sq
        return sum_sq
```