Angry Birds Project
-- vision component

By: Andrew Wang
Supervisor: Dr. Stephen Gould

COMP4560
Advanced Computing Project
Outline

• Background
• Existing system
• Proposed method
  – Connected-component extraction
    • Pixel labelling
    • Edge detection
  – Shape deduction
• Results
• Conclusion & future work
Fig 1. Screen-shot from Chrome version of the game
The Angry Birds project

• An agent that automatically plays the game
  – Input: screen-shot of the game
  – Output: sequence of shots which maximise the score

• Component breakdown
  – Computer vision
  – Knowledge representation & reasoning
  – Planning
  – Machine learning
Existing System

- Compress each pixel to 9 bits
- Extract connected-component by matching each pixel against possible colours for each class
- Bounding box determined by minimum/maximum x, y values
- Time: < 0.2 s
- Issues:
  - Actual shape and orientation not detected
  - Stacked objects
  - Inaccurate bounding box
Existing System

Fig 2. Example where shape and orientation of objects are not detected
Existing System

Fig 3. Stacked ice blocks treated as a single object
Existing System

Fig 4. Inaccurate bounding boxes
Existing System

Fig 4. Inaccurate bounding boxes
Existing System

Fig 5. Failure to describe the objects accurately making it almost impossible to reason
Proposed Method

- Assign a class label to every pixel in the image (e.g. red bird, wood, ice etc.)
- Determine object boundaries using an edge detector
- Extract objects as connected-components of the same class
- Deduct shape of the component from its boundary points
Pixel Labelling

• Manually labelled training data
  – Set of points in RGB space and their corresponding class
  – e.g. {160, 0, 0, RED_BIRD}

• Classify using Nearest-Neighbour rule
  – Each point is assigned the same label as its closest neighbour in training data

• Issue: NN is slow

• Solution:
  – Compress each pixel to 15-bits and calculate the class label for all colours during initialisation phase
  – Label each pixel via table lookup
Pixel Labelling

Fig 6. left: original image; right: labelled image, 1 colour for each class label
Edge Detection

- Existing approach: Canny edge detector
Edge Detection

• Issue with intensity based
• Colour space: \{hue, saturation, intensity\}
• Define similarity measure as weighted Euclidean distance

\[ D(\vec{a}, \vec{b}) = \sqrt{w_h(a_{\text{hue}} - b_{\text{hue}})^2 + w_s(a_{\text{sat}} - b_{\text{sat}})^2} \]
Edge Detection

- Consider 3x3 neighbourhood of a point
- Edge strength in each direction $i \in \{1, 2, 3, 4\}$
  
  $G_i = D(\vec{v}_0, \vec{v}_i) + D(\vec{v}_0, \vec{v}_{i+1})$

- Non-maximum suppression and double threshold applied for each direction
- Combine edges in all directions
Edge Detection

Fig 7. left: original image; right: edge image
Connected Components

- BFS until edge / different class label encountered

Fig 8. left: original image; right: all connected components
Shape deduction

- Required shapes: rectangles, circles and polygons
- Draw a bounding box around the component for angles from 0 to 90 degrees
  - Rotate boundary points by $\theta$
  - Size of bounding box determined by min/max x, y coordinates
- Reduce computation complexity
  - Only use points where edge orientation changes (corners)

\[
x' = x \cos \theta - y \sin \theta \\
y' = x \sin \theta + y \cos \theta
\]
Shape deduction

- Record the maximum and minimum area
- Circles: minArea * t1 > maxArea
- Polygon: minArea > component size * t2
- Otherwise shape is given by the minimum area rectangle
Results

Fig 9. Vision output on level 19
Results

Fig 10. Vision output on level 25
Conclusion & Future work

- Improvement over the existing system
  - Time: < 0.2 seconds $\rightarrow$ < 0.4 seconds
- Generally reliable for known objects
- Improve result by reasoning
- Extend to detect unknown objects
Questions