Ullman's Visual Routines and Tekkotsu Sketches

15-494 Cognitive Robotics
David S. Touretzky & Ethan Tira-Thompson
Carnegie Mellon
Spring 2011
Parsing the Visual World

• How does intermediate level vision work?
  - How do we parse a scene?

• Is the x inside or outside the closed curve?
Ullman: Visual Routines

- Fixed set of composable operators.
- Wired into our brains.
- Operate on “base representations”, produce “incremental representations”.
- Can also operate on incremental representations.
- Examples:
  - shift of processing focus
  - indexing (odd-man-out)
  - boundary tracing
  - marking
  - bounded activation (coloring)
## Base Representations

- Derived automatically; no decisions to make.
- Derivation is fully parallel.
  - Multiple parallel streams in the visual hierarchy.
- Describe local image properties such as color, orientation, texture, depth, motion.
- Marr's "primal sketch" and "2 ½-D Sketch"

<table>
<thead>
<tr>
<th>Input Image</th>
<th>Viewer centred</th>
<th>Object centred</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceived intensities</td>
<td>Primal Sketch</td>
<td>2 1/2-D Sketch</td>
</tr>
<tr>
<td>Zero crossings, blobs, edges, bars, ends, virtual lines, groups, curves boundaries.</td>
<td>Local surface orientation and discontinuities in depth and in surface orientation</td>
<td>3-D models hierarchically organised in terms of surface and volumetric primitives</td>
</tr>
</tbody>
</table>
Primal Sketch

(a) input image  
(b) sketch graph — configuration  
(c) pixels covered by primitives  
(d) remaining texture pixels  
(e) texture pixels clustered  
(f) reconstructed image
Incremental Representations

• Constructed by visual routines.
• Describe relationships between objects in the scene.
• Construction may be inherently sequential:
  – tracing and scanning take time
  – the output of one visual routine may be input to another
  – pipelining may speed things up
• Can't compute everything; too many combinations.
• The choice of which operations to apply will depend on the task being performed.
Dual-Coding Representation

- Paivio's “dual-coding theory”:

  People use both iconic and symbolic mental representations. They can convert between them when necessary, but at a cost of increased processing time.

  [Diagram of Paivio's dual-coding theory]

  Alan Paivio
Dual-Coding In Tekkotsu

- Tekkotsu implements Paivio's idea:

  - Sketch space = iconic representation
  - Shape space = lexical representation

- What would Ullman say? Visual routines mostly operate on sketches, but not exclusively.
Sketches in Tekkotsu

- A sketch is a 2-D iconic (pixel) representation.
- Templated class:
  - Sketch<uchar>  
    - unsigned char: can hold a color index
  - Sketch<bool>  
    - true if a property holds at image loc.
  - Sketch<uint>  
    - unsigned int: pixel index; distance; area
  - Sketch<usint>  
    - unsigned short int
  - Sketch<float>
- Sketches are smart pointers.
- Sketches live in a SketchSpace: fixed width and height.
- A built-in sketch space: camSkS.
Making New Sketches

• We can use a macro to create new sketches:

\[
\text{NEW\_SKETCH}(name, type, value)
\]

• The \textit{name} will be used as a variable name.

• The \textit{type} should be one of bool, uchar, uint, etc.

\[
\text{NEW\_SKETCH}(\text{camFrame}, \text{uchar}, \text{sketchFromSeg}())
\]
VisualRoutinesStateNode

- Subclass of StateNode

- Provides several SketchSpace / ShapeSpace pairs.

- Allows you to view the SketchSpace remotely, using the SketchGUI tool.

- Let's try a sample image:
First Visual Routines Example

#include "Behaviors/StateMachine.h"
using namespace DualCoding;

$nodeclass DstBehavior : VisualRoutinesStateNode : doStart {
    camSkS.clear();
    NEW_SKETCH(camFrame, uchar, sketchFromSeg());
    NEW_SKETCH(orange_stuff, bool,
                visops::colormask(camFrame,"orange"));
    NEW_SKETCH(o_edge, bool, visops::edge(orange_stuff));
    NEW_SKETCH(o_skel, bool, visops::skel(orange_stuff));
    NEW_SKETCH(o_neighbs, uchar,
                visops::neighborSum(orange_stuff));
}
Color-Segmented Image
visops::colormask(“orange”)
visops::edge(orange_stuff)
visops::skel(orange_stuff)
visops::neighborSum(orange_stuff)
Second Example

- Find the largest blue region in the image:
Second Example

$nodeclass DstBehavior : VisualRoutinesStateNode : doStart {
    camSkS.clear();
    NEW_SKETCH(camFrame, uchar, sketchFromSeg());

    NEW_SKETCH(blue_stuff, bool, 
                visops::colormask(camFrame,"blue");
    NEW_SKETCH(b_cc, uint, visops::labelcc(blue_stuff));
    NEW_SKETCH(b_area, uint, visops::areacc(b_cc));
    NEW_SKETCH(b_max, bool, b_area == b_area->max());
}
camFrame
visops::colormask
Components labeled starting from 1 in upper left; max label in lower right.
visops::areacc
b_area == b_area->max()
Third Example

- Find the orange region closest to the largest blue one; ignore any orange noise (blobs smaller than 10 pixels).
Third Example

```cpp
NEW_SKETCH(b_dist, uint, visops::edist(b_max));

NEW_SKETCH(orange_stuff, bool,
           visops::colormask(camFrame,"orange")[needs more context here]);
NEW_SKETCH(o_cc, uint, visops::labelcc(orange_stuff));
NEW_SKETCH(o_area, uint, visops::areacc(o_cc));
NEW_SKETCH(o_blobs, bool, o_area > 10);

NEW_SKETCH(bo_dist, uint, b_dist\*o_blobs);
int const min_index = bo_dist->findMinPlus();
int const min_label = o_cc[min_index];
NEW_SKETCH(bo_win, bool, o_cc == min_label);

NEW_SKETCH(rawY, uchar, sketchFromRawY());
```
visops::edist(b_max)
\texttt{o\_area > 10}

\texttt{NEW\_SKETCH(o\_blobs, bool, o\_area > 10);}
bo_dist

NEW_SKETCH(bo_dist, uint, b_dist*o_blobs);
bo_win

NEW_SKETCH(bo_win, bool, o_cc == min_label);
Sketch Properties

• Every sketch has a color, and a colormap.
• Sketch<bool> is rendered in that color.
• Sketch properties are inherited from the first argument of any visual routine or sketch operator.
• Example:

    NEW_SKETCH(result, bool, blue_stuff | orange_stuff);

    The result will have color blue.

• Colormaps: segMap, grayMap, jetMap, jetMapScaled
Sketch Constructor #1

- Specify a sketch space and a name:

```cpp
Sketch<bool> foo(camSkS, "foo");
foo = false;
for ( int i=50; i<90; i++ )
    foo(i,i) = true;
foo->V();
```
Sketch Constructor #2

• Specify a name and a parent sketch to inherit from.

```cpp
Sketch<uchar> bar("bar", foo);
bar = (Sketch<uchar>)foo + 5;
bar->V();  // make viewable in SketchGUI
```

• Sketch bar's parent is foo.

• We can use type coercion to convert Sketch<bool> to Sketch<uchar> in order to do arithmetic.
Result of Second Constructor:
Sketch bar

![Sketch bar](image)
NEW_SKETCH Macro

- **NEW_SKETCH** is just syntactic sugar:

  ```
  NEW_SKETCH(orange_stuff, bool,
              visops::colormask(camFrame,"orange");
  ```

- This expands into a **copy constructor** call:

  ```
  Sketch<bool> orange_stuff(visops::colormask(...),
                           "orange_stuff",
                           true);
  ```

  Indicates sketch should be visible in the SketchGUI
SketchSpaces:
A Look Under the Hood
Do Tekkotsu's Representations Fit Ullman's Theory?

- What are the base representations?
  - color segmented image: `sketchFromSeg()`
  - intensity image: `sketchFromRawY()`
  - extracted blobs

- What are the incremental representations?
  - Sketches
  - Shapes

- What's missing?
  - Attentional focus; boundary completion; lots more.
Triesman's Visual Search Expt.

Find the green letter:

<table>
<thead>
<tr>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Triesman's Visual Search Expt.

Find the O:
Triesman's Visual Search Expt.

Find the green O:
What Do Human Limitations Tell Us About Cognition?

- Subjects can't do parallel visual search based on the intersection of two properties.
- This tells us something about the architecture of the visual system, and the capacity limitations of the Visual Routines Processor.
  - Base can't do intersection.
  - VRP can't process whole image at once.
  - There must be a limited channel between base and VRP.
- But in Tekkotsu, we can easily compute intersections of properties.
  - Is that a problem?
Science vs. Engineering

- Science: figure out how nature works.
  - Limitations of a model are good if they suggest that the model's structure reflects reality.
  - Limitations should lead to nontrivial predictions about comparable effects in humans or animals.

- Engineering: figure out how to make useful stuff.
  - Limitations aren't desirable.
  - Making a system “more like the brain” doesn't in itself make it better.

- What is Tekkotsu trying to do?
  - Find good ways to program robots, drawing inspiration from ideas in cognitive science.