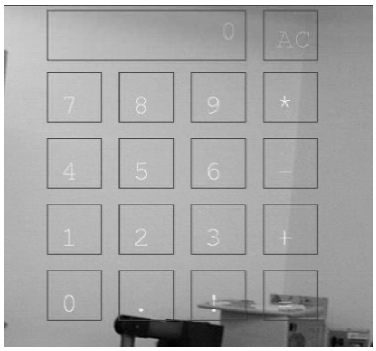
VICs: A Modular Vision-Based HCI Framework

The Visual Interaction Cues Project

Guangqi Ye, Jason Corso
Darius Burschka, & Greg Hager
CIRL, The Johns Hopkins University

Today, I'll be presenting work that is part of an ongoing project in the Computational Interaction and Robotics Lab at the Johns Hopkins University. The Visual Interaction Cues project is focused on vision-based interaction. This talk will introduce a framework for solving the interaction problem and discuss an example implementation that incorporates motion dynamics into the activity recognition process.

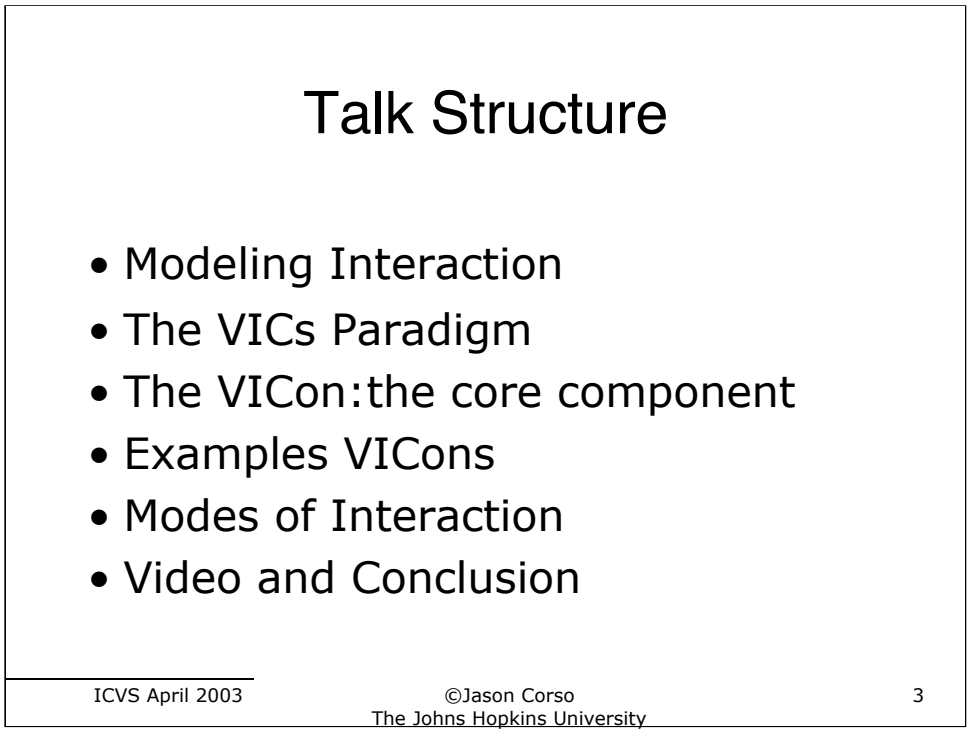
Visual Interaction Cues (VICs)



With this first slide, I will motivate the general, vision-based interaction problem. Here, you see two examples of VICs-based interfaces. On the left is a simple gesture based interface where the user can grab the icon and drag it across the display. On the right is a calculator program using vision as input.

As I mentioned on the title slide, the VICs project aims at using video as input for human computer interaction. This yields a fairly complex problem that must be solved; first, if you think about current interfaces for a moment, they are inherently one-dimensional in nature. They are dominated by the mouse and the input vocabulary is extremely small. However, when incorporating one or more video streams as the interaction medium, the dimensionality of the input increases along both spatial and temporal axes.

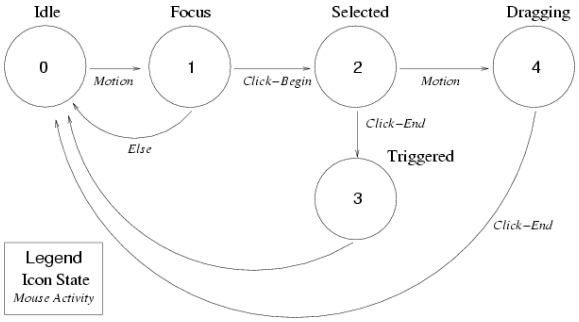
Thus, we are trying to make efficient use of this higher dimensional data in a way that will maximize action detection capability while minimizing computation.



The talk is structured in the following fashion. First, I will discuss how we model interaction. Then, I will introduce the VICs paradigm and discuss its core component. After presenting two example VICons, I will enumerate the various modes of interaction in which VICs can exist followed by a video demo and a short conclusion.

Modeling Interaction

- Mainstream Interface Technology:
  WIMP - Windows, Icons, Menus, and Pointers. [van Dam 97]



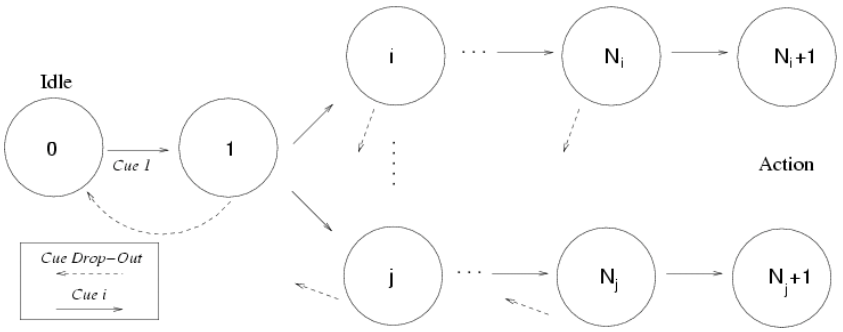
ICVS April 2003 ©Jason Corso The Johns Hopkins University 4

If you recall my earlier discussion about current interface technology, we see that such interfaces can be modeled with a simple state machine as shown in the diagram on the slide. Idle-Focus-Selected is the sequence prior to any action taken by the icon. This simplicity is due to the nature of the input device.

These sequential interfaces are governed by devices that set the focus on the user: the mouse yields the user's current location and the state of one or buttons. Usually, based on where the user clicks a button, an interface component responds accordingly with its one associated action. Thus, the number of outcomes of a single user action sequence is fairly small at best.

Modeling Interaction

- A more general model:



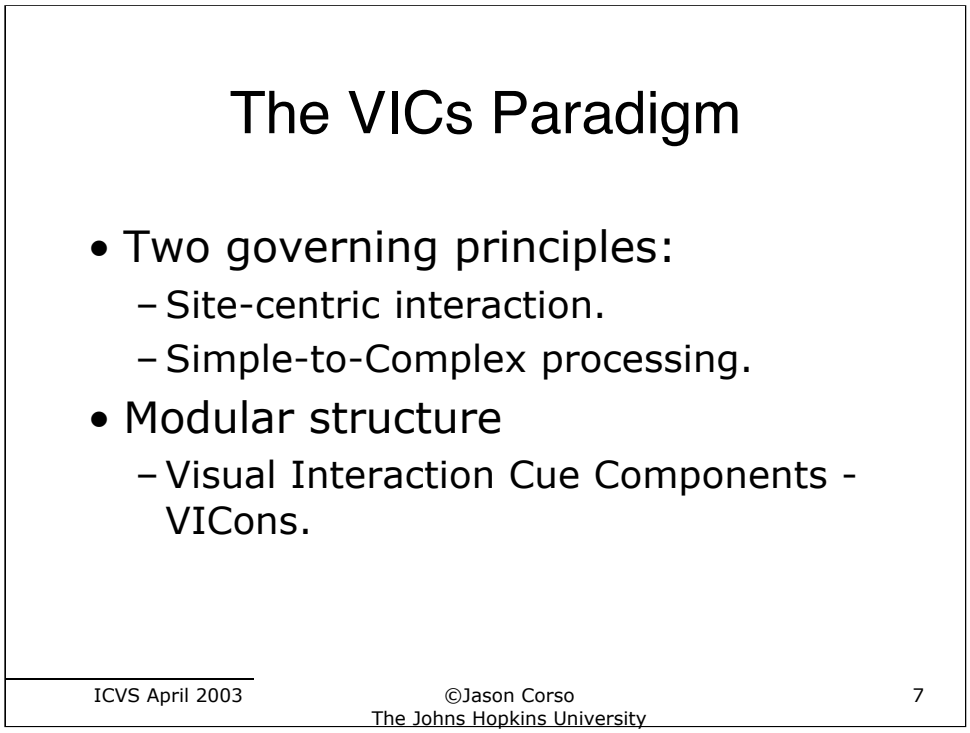
However, in next generation interfaces we will begin to see a more general model that has a parallel nature and a higher magnitude of outputs per action sequence. Obviously, video input streams offers one such approach at expanding the dimensionality of human-machine interfacing.

Harnessing the Power of Vision

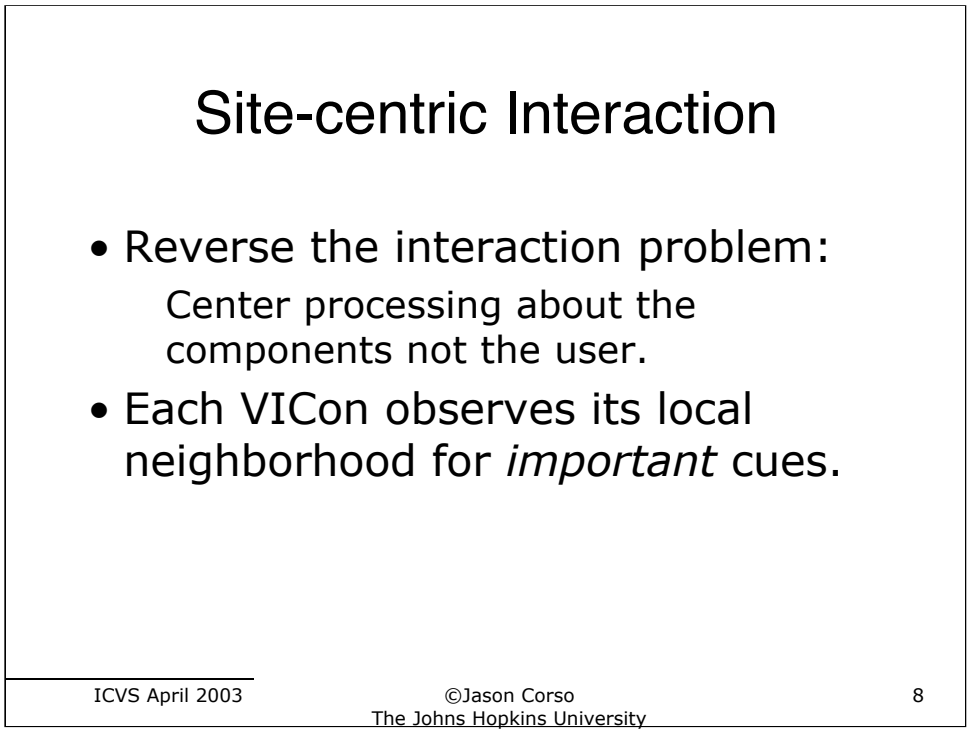
- Difficult
- Tracking-based approaches
  - Gaze, Head, Full-body tracking
- We differ by
  - Placing the focus on the interface.
  - Kjeldsen et al. (Session 5)

Harnessing the power offered by computer vision has proven to be a difficult task where we have seen the field dominated by approaches to directly expand current interfaces. That is to say, most approaches are based on tracking the user -- either gaze, hand, or full body tracking. For example, there was a recent paper that use nose-tracking to mimic the operation of a mouse.

Our work differs from these tracking based works on a fundamental level. We take the focus away from the user and place it on the interface modules. The interface does not need to know what the user is doing at all times. Instead, it is only concerned when the user is near a possible site of interaction; for instance, in the calculator example on the first slide, each button is idle until it notices some motion in its neighborhood.



This approach to the interaction problem is called the VICs paradigm. Approaching the problem in this manner yields a more computationally efficient and robust solution space. The VICs paradigm is based on two governing principles, namely site-centric interaction and simple-to-complex processing. We strive to maximize detection while minimizing computation. Thus, the paradigm is built with a modular structure facilitating the incorporation of VICs components into both current and future interfaces.



We base the framework on the notion of site-centric interaction. Instead of trying to solve the problem of tracking the user, we bring the user to the various interface components. Fundamentally, this is an equivalent problem; it's a more simple one to propose robust solutions.

To reiterate and make this more concrete: in a conventional interface setting with the user pointing their finger instead of using a mouse to point and click. It is unnecessary to know where the user's finger is at all times. Instead the sites of interaction, I.e. the icon, menus, and buttons, only need to watch for when the finger encroaches into their neighborhood. Processing in this fashions removes the need to perform costly, global tracking procedures.

Simple-to-Complex Processing

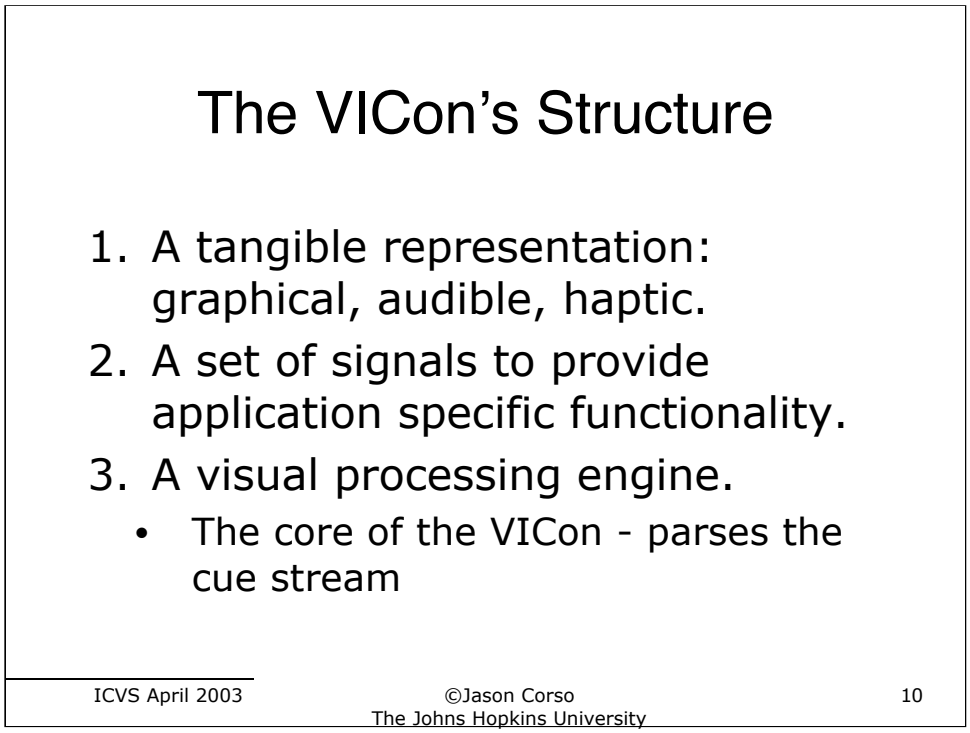
- Maximize detection vs. minimize computation
- Typical approach - template matching
  - Prone to false-positives
  - Potentially wasteful
- Watch for a stream of cues structured from simple-to-complex
  - E.g.. Motion detection : Hue Blob : Shape Test : …

The second basic principle on which the VICs paradigm is based is structured processing. We model interaction with the general state machine I showed earlier.

Given site-centric processing, the immediate solution one of template matching. However, such an approach is prone to false positives and can be potentially wasteful. For instance, if the interface is covered with components, each doing a template matching solution on their neighborhood in the current video frame the system's computation will be wasted in regions where nothing is happened.

Instead we structure the processing in a simple-to-complex manner in an effort to minimize wasted computation and maximize correct detection rates. One example of a simple routine is motion detection.

As you will see in the second example, using this general state model, we are able to incorporate varying temporal aspects into the components of our interface.

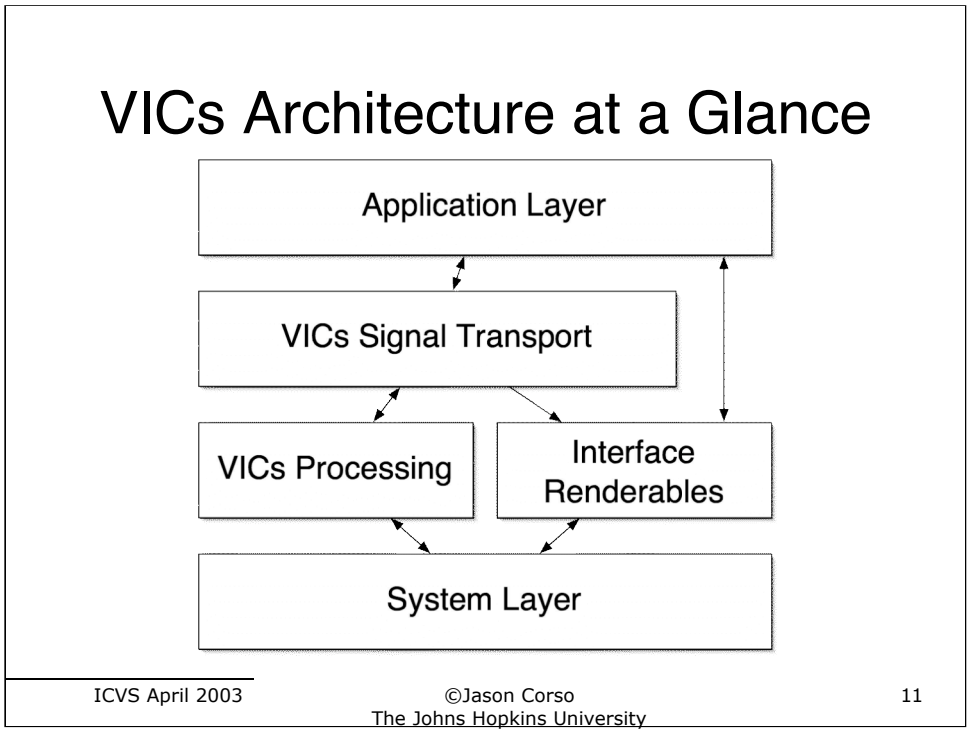


At the core of our framework is the VICon; any vision enabled interface component operating under the VICs paradigm is loosely termed a VICon. It has three parts:

One, a tangible by which it can render itself to the user, these include graphical, audible, and haptics-based.

Two, a set of application specific signals that triggered by pre-defined action sequences like a button-push.

And at its core, a visual processing engine, or parser. This parser sits atop a state machine that is modeled for a given set of action sequences. It is in this underlying vision processing that temporal aspects and high-dimensional spatial interaction is modeled.



On this slide is a figure that gives a simple introduction to the architecture in which the VICs framework is implemented. I can provide further reading if anyone is interested.

The VICs framework operates as a substrate beneath any applications. Like most event-driven application programming interfaces, it communicates with the application via a set of signals and directly communicates with the system to handle such tasks as video acquisition and interface rendering.

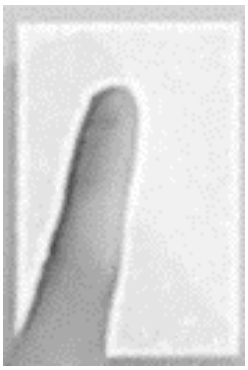
An Example VICon - A Button

- The order of the cue-parser
  - Motion
  - Hue Blob
  - Shape

Now, I will present two example VICons. The first is a simple spatially activated push-button modeled with a 3-state parser.

An Example VICon - A Button

- The order of the cue-parser
  - Motion
  - Hue Blob
  - Shape



An Example VICon - A Button

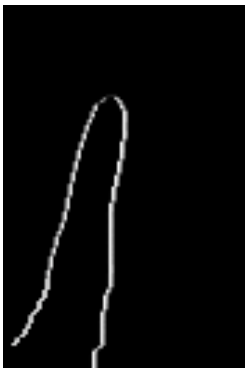
- The order of the cue-parser
  - Motion
  - Hue Blob
  - Shape
- Background Subtraction

An Example VICon - A Button

- The order of the cue-parser
  - Motion
  - Hue Blob
  - Shape

An Example VICon - A Button

- The order of the cue-parser
  - Motion
  - Hue Blob
  - Shape



Computation Minimized

- Constant computations per-pixel.
  - In this case, a difference and a threshold.
- With action, increased computation only occurs near the action.
- Unnecessary computation removed.

Thus picture an interface completely covered with VICons similar in design to the first example. If no action is occuring in the video frame, then the system will perform a constant amount of computation per video frame. In this case, a difference and a threshold per-pixel. If an action is occuring, more complex processing will only occur in regions near in the action. Thus, we have designed a framework that make a best effort to minimize unnecessary computation.

Example using Motion Dynamics

- A Stochastic VICon via Hidden Markov Model.
  - Commonly used in Speech Recognition.

- Emulates a simple button
  - 2 state VICon model

This second example is our first product that incorporates motion dynamics, I.e. temporal information fused with spatial information. It, too, models a simple button press. However, it is a stochastic VICon and uses a Hidden Markov Model to analyze motion dynamics. HMMs are commonly used in the speech recognition problem. The figure on the slide depicts the flow of such a system: a filterbank operates on discrete clips of speech. The output of the filterbank is passed to an HMM model for acoustic processing which yields symbolic output. There is a symbol per acoustic element: most commonly, these are phones. There is a phone for each sound, like a, aaaa.

We use HMMs in a similar fashion, given input from a filterbank that computes some measure on the input stream, the HMM outputs a symbol from its dictionary or null. In our case, the outputted symbol corresponds to activating the button from one of four directions.

For simplicity the VICon state machine in this example is a two state one with the HMM operating in the first state. However, it should be noted that the HMM can operate as a more complex state in a VICon similar to the first example.

The HMM State-Space

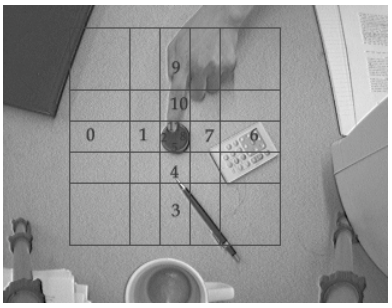
- Convert input image stream into a series of symbols that describes the system state.
- Discrete feature describing current position and orientation of the finger tip.

•3 Distances

•4 Directions

Up,left,etc

•Yields 12 states



As I just said, the HMM expects input from a discrete feature set. Thus, we create a feature-set that splits the region around the button into a 5 by 5 grid with the button in the center.

Since we are interested in position and orientation, we define 12 states over our feature space: 3 distances for each of 4 directions. A state is active when it's corresponding cell is determined to be in the foreground of the scene; our foreground segmentation algorithm is presented on the next slide.

From this state-space, we have four actions: triggering the button from each of the four directions.

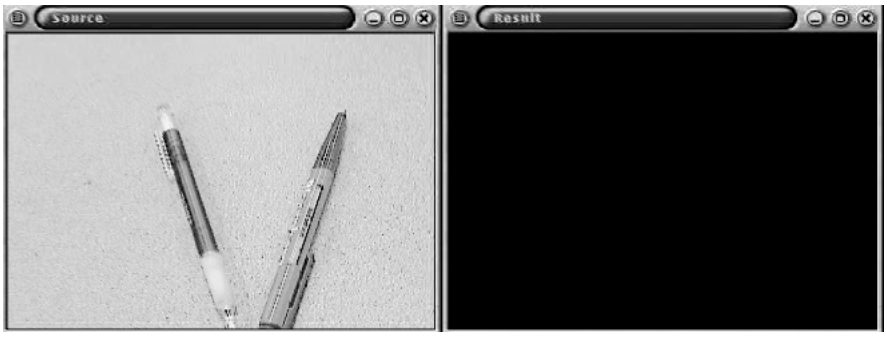
BG/FG Modeling & Segmentation

- Assume static camera.
- Hue Histogram to model appearance on-line.
- Segmentation based on histogram intersection.

$$HI(Measure, Model) = \frac{\sum_{i=1}^{n} \min(Measure_i, Model_i)}{\sum_{i=1}^{n} Model_i}$$

To segment the foreground from the background, in this vision module, we employ online histogram modeling and histogram intersection. This approach is robust to simple changes in lighting, like the dimming of office lights, and it is relatively invariant to translation and rotation about the viewing axis.

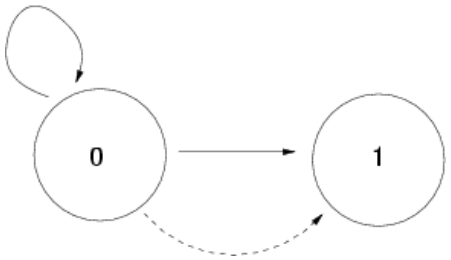
Foreground Segmentation : Example



Here is an example of the segmentation operating over an entire image.

The HMM Structure

- Building block: singleton HMM.

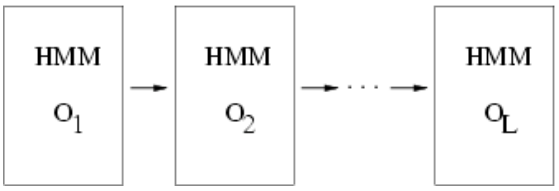


- For each of the 12 states, define basic HMM to represent it.

Similar to traditional acoustic processing, the basic structure of our HMM is the singleton model. For each of the 12 states we defined a singleton to represent it.

The HMM Structure

- Build an HMM for each action category (up,down,etc).
- Concatenate singletons based on a representative sequence and fix a length L.



- If likelihood for a sequence is too low, consider it an illegal sequence.

Then we build a larger HMM for each of the four action categories by concatenating a set of the singleton HMMs. To chose the exact structure of this larger HMM, for each action category, we choose a representative sequence and use its singleton flow as the representative one.

One important point to note here is that we also must define an action that corresponds to the null action; for instance, if the user's finger passes by the button without pressing it. However, unlike the speech problem where there is a single point in state space corresponding to silence, we have many possibles sequences of states that result in an invalid action. To solve this problem, instead of explicitly defining a null-action state, we choose a threshold on the likelihood of each of the other four action's occurring. If neither of them have high likelihood, then we consider the sequence a null-action.

HMM Training and Recognition

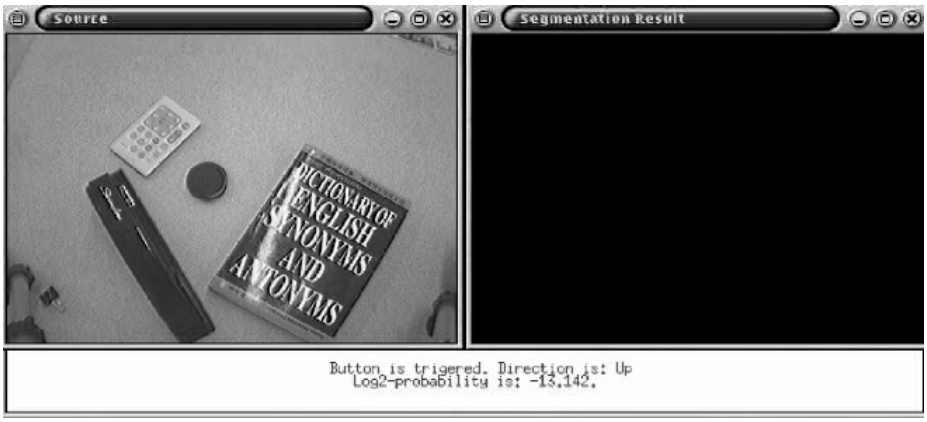
- Training set of valid actions.
- Select a characteristic sequence for each of the 4 directions.
- Run the Baum-Welch Algorithm.
- At run-time, for each length L sequence, attempt recognition.
  - If valid, trigger correct signal.

We train the system by recording a set of valid (I.e. non-null-actions) actions and use the Baum-Welch algorithm to calculate the state transition probabilities.

At run-time, we attempt recognition for each video sequence and trigger the correct signal if a valid action has occurred.

Experiment Results

- 76 sequences for training, over 300 for testing.
- 100% on training set; 96.8% on test set.



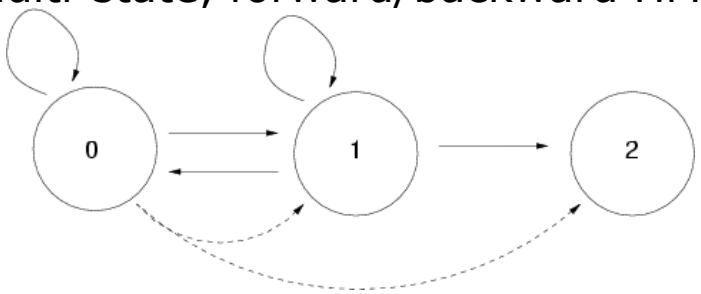
For image resolution of 320 x 240, system runs over 20 fps on Pentium III pc.

Foreground segmentation: 8 bins for hue histogram, sub-images of size 4x4, average correct ratio about 98%.

Robustness to modest illumination changes, e.g., turn on/off the office lights.

Improving the HMM Structure

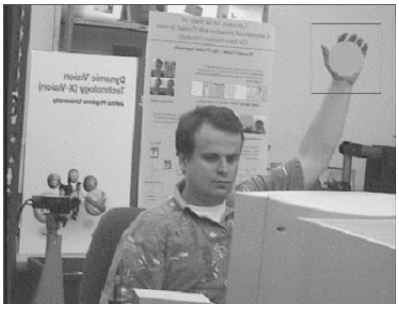
- Singleton-based HMM is rudimentary
- Incorporate time dynamics into 1 multi-state, forward/backward HMM.



Since the submission of this paper, we have changed the HMM structure to a more sophisticated one. In the new HMM, we incorporate the time dynamics into one multi-state forward/backward HMM instead of a concatenation of singletons. Thus new structure will be able to better capture actions of a more dynamic nature.

Interaction Modes 1

- 2D-2D Mirror
  - One camera observes user
  - Video stream displayed in interface.
  - VICons composited into video stream.



To finish the talk, I will enumerate some interaction modes and then present a short video of a working system.

The first of the 5 modes is 2D-2D Mirror. The two videos I showed at the beginning of the talk demonstrate this style of interaction wherein video of the user is rendered onto the screen and virtual objects are composited into the video stream. This is a good way to allow the user to employ the motor coordination skills from the real-world in the interface.

Interaction Modes 2 & 3

- 2.5D Augmented Reality
  - Video see-through
  - Constrain interface to a surface
- 3D Augmented Reality
  - Allow VICons to be fully 3D
- Examples
  - Surgery for 3D Microscopes; e.g. retinal
  - Motor-function training for young children.

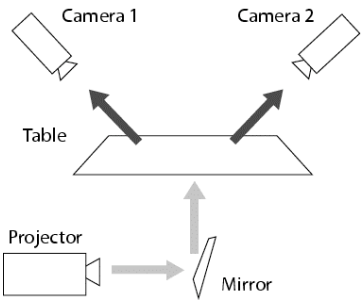
Interaction modes 2 and 3 are based on augmented reality. In this case, a user is wearing a head-mounted display and video of the world is being rendered into the helmet. Virtual objects are composited into the stream and the user is then allowed to interact with 2.5D and 3D virtual environments.

In the 2.5D case, the interface is pinned to a surface in the world and VICons operate in this subspace.

Applications of these modes are numerous. One example is augmented an eye-surgeon with a stereo microscope and using 3D VICons to allow the surgeon better control of his tools. We have one such microscope in the lab and are currently building such a demonstration interface. Many simulation-type applications will also benefit from employing VICs-based components.

Interaction Modes 4 & 5

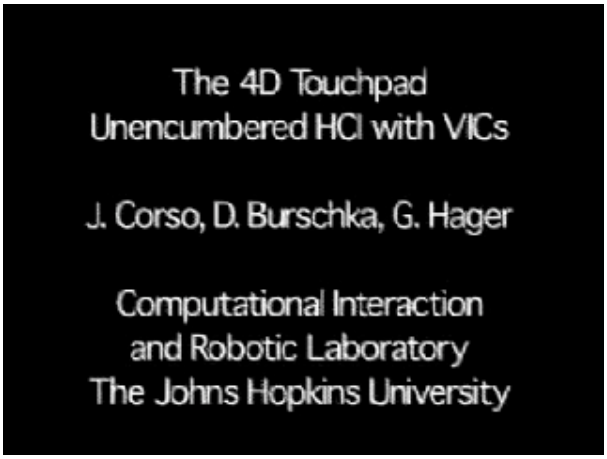
- 2D-2D & 3D-2D Projection
  - 1, 2 or more cameras
  - The 4D-Touchpad [CVPRHCI 2003]
    - Provisional Patent Filed.



The last two interaction modes are projection style modes. In these cases, the display is projected onto a surface and the user interacts with the interface as if it were in the real world. One, Two or more cameras are observing the user and the interface and feeding the information to the system.

We have a paper in CVPRHCI 2003 that demonstrates the 4D-Touchpad.

Video Example
3D-2D Projection - 4DT



Conclusions

- A new framework for transparently incorporating vision-based components into interface design.
- Our first system to incorporate motion dynamics in a formal manner.
- Can we fuse the higher spatial dimension and temporal nature of interaction in a structured way?
- A language of interaction?

To conclude, I have presented a new framework for vision-based interfaces that makes good use of the increased amount of information offered by using video as input. I have also talked about our first attempt at incorporating temporal dynamics into the visual stream parsing. At this point, we are trying to develop more complex interfaces wherein we will fuse higher dimensional spatial information with temporal dynamics which will lead us into making full use of the video input.

- Thank You.
- Questions/Comments?
- Acknowledgements:
  - This material is based upon work supported by the National Science Foundation under Grant No. 0112882. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.