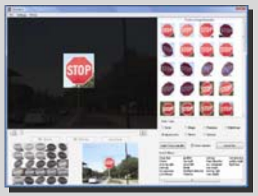
Eyepatch: Prototyping Camera-based Interaction through Examples

Dan Maynes-Aminzade, Terry Winograd, Takeo Igarashi
monzy@stanford.edu

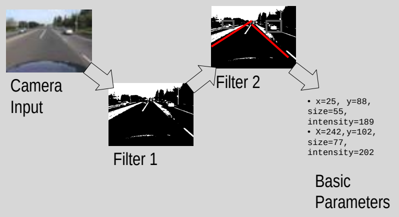
Introduction

Cameras are a useful source of input for many interactive applications, but computer vision programming is difficult and requires specialized knowledge that is out of reach for many HCI practitioners. In an effort to learn what makes a useful computer vision design tool, we created Eyepatch, a tool for designing camera-based interactions, and evaluated the Eyepatch prototype through deployment to students in an HCI course.

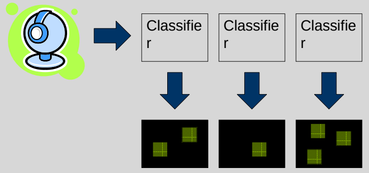


Approach

Eyepatch allows novice programmers to extract useful data from live video and stream that data to other rapid prototyping tools, such as Adobe Flash, d.tools, and Microsoft Visual Basic.


Camera Input
Filter 1
Filter 2
• x=25, y=88, size=55, intensity=189
• X=242,y=102, size=77, intensity=202
Basic Parameters

In our simple computer vision framework, input comes from a *camera or recorded video*, the sequence of frames is passed through one or more *binary classifiers*, and the output is the *set of image regions* yielded by each of the active classifiers and certain parameters associated with these regions, such as color, shape, size, and position.



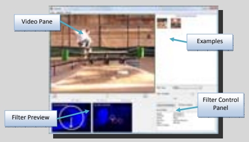
Tool Design

Eyepatch has two basic modes:

• A **training mode**, where users can train different types of classifiers to recognize the object, region, or parameter that they are interested in.
• A **composition mode**, where the classifiers are composed in various ways and users specify where the output of the classifiers should go.

Training Mode

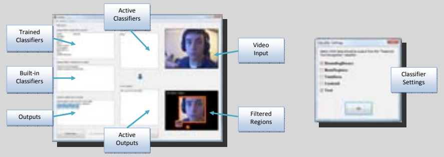
The Eyepatch **training mode** uses an interactive learning approach. The user loads a video in any standard format, or records a video from an attached webcam. The user can scroll through the video, much like in a video editor, and use the mouse to highlight regions of frames to use as examples for training a classifier.



After selecting some examples from the video feed, the user selects a classification method and clicks “Learn from Examples” to train a classifier. He can then check “Show Guesses” to observe the output of the trained classifier on the current frame. By scrolling through the frames in rapid succession, he can quickly judge the performance of the classifier, and if it is not satisfactory, he can provide more examples or try a different classification method.

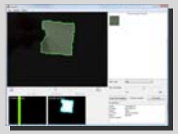
Composition Mode

The **composition mode** allows users to select among the classifiers you have trained, run them on live video, and stream their output data to different locations.

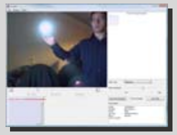


Classifier Types

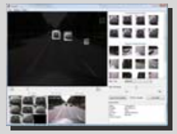
Eyepatch currently supports seven classifier types:



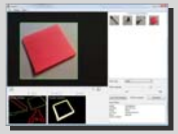
Color, based on hue histograms and backprojection, for identifying distinctively colored objects.



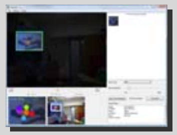
Brightness, for finding the brightest or darkest regions of an image, such as laser dots or shadows.



Adaboost, a machine learning technique that uses a boosted cascade of simple features, for recognizing general classes of objects like faces, animals, cars, or buildings.



Shape, based on Canny edge detection followed by contour matching using pair-wise geometrical histograms, for finding objects with distinctive outer contours.



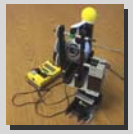
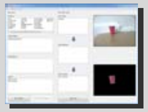
Scale-Invariant Feature Transforms, for recognizing specific objects with invariance to scale, pose, and illumination.

Motion, based on segmentation of a motion history image, for identifying the directions of moving objects in the scene.

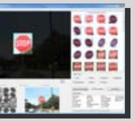
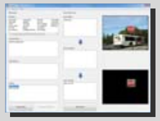
Gesture recognition, based on blob detection followed by motion trajectory matching using the Condensation algorithm, for recognizing particular patterns of motion.

Sample Projects

Describing a few projects we built with Eyepatch will provide a sense of its capabilities. We completed each of the following projects in a few hours, with the majority of this time devoted to components of the projects unrelated to computer vision.

BeiRobot is a Lego robot that uses computer vision to play Beirut. Beirut is a party game in which players attempt to throw ping-pong balls into cups. BeiRobot uses a color classifier to detect cup positions in the input image, and it rotates its swiveling base until one of the cups is centered in the image. It then launches a ping-pong ball at the cup, using the size of the detected cup region to estimate the cup’s distance from the camera.

The stop sign warning device warns drivers when they are about to run through a stop sign. It uses an Adaboost classifier to recognize stop signs, and a motion classifier to detect when the car is moving. If a stop sign is detected close to the camera, and the car continues forward without stopping, the device emits a warning tone.

The logo scoreboard uses SIFT classifiers to recognize an assortment of five different corporate logos. It watches live television and counts how many times each logo appears. A system like this would allow marketers to assess the brand penetration of their company or evaluate the success of their product placement campaigns.

Try It Yourself!

Eyepatch is open source software licensed under the GPL. A Windows installer for Eyepatch is available for download at http://eyepatch.stanford.edu/.

STANFORD HCI GROUP