UNIVERSITY OF NEBRASKA-LINCOLN

>HACK::Home Automation Controlled by Kinesics

CSCE489 Progress Report II

Author: Eric Grubaugh; Team Members: Joe Smith, Casey Dusseau

3/26/2009

This document contains a detailed description of the HACK project. Background information is provided to provide the reader a firm understanding of the basic principles applied. Design concepts, system architecture, system implementation, and future expansions are all discussed herein.

Introduction

Project Overview

The purpose of the HACK project is to combine rigid object detection with articulated object tracking to create a human gesture-controlled system, that is, a system controlled entirely by specific motions of the human body—specifically the hands. While the area of rigid object detection has been thoroughly researched and documented, it has yet to be paired with articulated object tracking to provide a seamless and robust control interface.

The HACK project can provide this interface, as the following sections show. Such an interface has enormous potential in almost any field of research and development, and the application of the concept shown in this document is only one of the myriad possibilities for this technology. This gives rise to another key element of the HACK system: extendibility. One goal for this system is for it to be easily adaptable to any number of gesture-controlled applications.

Document Overview

The remainder of this document begins by discussing relevant core concepts in human-computer interaction and then applies these concepts to exhibit the design of the HACK project. The Architecture section describes the overall structure of the HACK system for both software and hardware. The Implementation section follows to explain in detail the physical realization of the system. The final two sections discuss possible feature additions for the future and a general summary of the entire project.

Background

In this section, the necessary background concepts and information are presented to give the reader a firm understanding of the HACK project's foundation. The gesture-controlled system is constructed upon the fundamental framework of human-computer interaction (HCI) and gesture recognition.

Terminology

Here we discuss some general terminology that will be helpful to the reader throughout the remainder of the report.

Kinesics

Kinesics is defined as the study of communication through body movements (Colorado State University, 2005). The HACK project will perform one sort of kinesic analysis to determine what is communicated through human body movements by mapping them to natural, intuitive control gestures for various

devices. These control gestures will then be used to manipulate a system of devices that will respond accordingly to specific gesture input.

HCI

Human-computer interaction is a discipline concerned with the design, evaluation, and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them (Hewett, et al., 2008). This is a broad subject that encompasses the core concepts that HACK is built upon and much, much more. As the name implies, at the heart of this discipline is human interaction with computers. HACK is one application of this interaction.

AdaBoost

AdaBoost is the most popular boosting algorithm because of its ease of both use and implementation (Hertzmann, 2008). Research has also proven that it can yield effective results. A boosting method, in general, is a classifier learning strategy that builds strong classifiers out of average, weak classifiers. A classifier is an object that is able to recognize an image as a specific item; a weak classifier is so-named because its ability to correctly identify an image is only slightly better than random chance. AdaBoost averages the outputs of a collection of these weak classifiers repeatedly to strengthen them and then reiterates, continually refining the classifiers.

Viola-Jones Object Detection

Rigid object detection is accomplished using Viola-Jones Haar classifiers. This object detection technique uses a cascade of boosted weak classifiers organized as a rejection cascade. Each weak classifier is composed of one or more Haar-like features. Weak classifiers are “boosted” into strong classifiers using a variation of the AdaBoost algorithm, and these strong classifiers make up nodes in the rejection cascade. At each stage in the cascade, the majority of positive objects —preferably better than 99%, at the expense of false-positives—are correctly identified, and a small percentage of non-objects are rejected. Each stage removes more false-positives until an acceptable positive-to-false-positive detection ratio has been reached. This allows the software to recognize images it has never seen before as being “similar” to the positive images it has been provided in the past.

Flock of Features Articulated Object Tracking

Articulated object tracking is accomplished using a Flock of Features approach where objects are identified by a constantly updated group of features. This allows the object to be detected and tracked even as it changes shape. This is accomplished first by initializing a collection of traceable features on a region of interest—in this case the location of a detected rigid shape—and as the object moves, the features are “tracked” into subsequent frames. Features that stray from the least-squares-ellipse center of this “flock” of features are discarded, and new features are found within a least-squares-ellipse-derived bounding box. The theory is that stray features do not belong to the object that is being tracked and are most likely a part of the surrounding environment. The features used are strong corner features defined as high eigenvalue edges in two dimensions. These corner features are supplemented with

surrounding image profile information, such as color and texture. By requiring new features to exhibit similar image profile information as pre-existing flock features, this multi-cue approach makes the feature finding and tracking process more robust.

Cascade Training

In order for a cascade to learn to correctly identify new images, a set of classifiers must be trained to recognize specific features that are common to all of the images, such as eyes on the human face. The images are preprocessed into sets of such features, and then a classifier algorithm runs over the features repeatedly. As the algorithm runs, it refines various process parameters, dropping classifiers that do not accurately represent data and reinforcing classifiers that adequately represent a given feature. This process of adaptively adjusting parameters is the genesis of the term “machine learning.”

OpenCV

OpenCV is an open source computer vision library designed for computational efficiency in real-time applications (Bradski & Kaehler, 2008). Computer vision is the transformation of data from a still or video camera into either a decision or a new representation. The OpenCV project strives to streamline the development of complex computer vision applications by abstracting the more difficult low-level details of computer vision. The library is built on a C/C++ framework and can be run on Linux, Mac OS X, and Windows operating systems.

Qt4

Qt4 is a cross-platform application and user interface framework built on C++ (Nokia Corporation, 2008). It is designed to deploy across multiple desktop and embedded operating system environments without rewriting any source code. There is also a cross-platform integrated development environment to streamline application development with this framework.

Bluetooth

Bluetooth is a small-area radio-frequency (RF) protocol used for streamlined wireless, low-power data transfer (Franklin & Layton, 2000). It is a means for devices to have electronic conversations over short distances. Up to eight devices can be connected simultaneously on one Bluetooth network up to a range of 10 meters (32 feet). Each Bluetooth data transfer uses approximately one milliwatt of power, a trivial amount when compared to the three watts that typical mobile phones can use.

Bluetooth operates in the RF band between 2.402 gigahertz (GHz) and 2.480 GHz. This bandwidth is shared by industrial, scientific, and medical devices. To avoid interfering with these devices and other Bluetooth devices, the Bluetooth protocol employs *spread-spectrum frequency hopping*. This technique allows the Bluetooth device to choose 79 discrete and randomly-chosen frequencies within the designated bandwidth and then rapidly switch among these 1600 times every second. Devices that are connected on a network will change frequencies in unison so that they stay in constant communication.

This makes it unlikely that one network will interfere with another, and even if multiple networks do land on the same frequency at one time, the interference will only last $^1/_{1600}$ of a second.

Now that we have the groundwork laid for the system, we can expand these core concepts and begin to form a base set of components and operations that can help realize the project goals.

Architecture

In the Architecture section, we take the design constructs discussed previously, and we form them together to describe an abstract system that is able to effectively implement the gesture-controlled system. This system is categorized by three main modules: the API, the Core, and the Kinesics Interface.

The API is the software backend that provides functionality that obscures the low-level details of the system, thus making the programmatic interface much cleaner and simpler on the developer. The Core module is composed of all the system hardware. This component is the underlying engine of the system, driving and receiving both internal and external communications. Finally, the Gesture Interface comprises the link between the two previous modules. It is here that the results from the API are translated into the commands and operations that manipulate the Core components.

Overview

The system is broken down in to these three separate components so that the major areas of functionality are kept separate. This modular design allows for streamlined development and testing, as well as a simple division of labor. **Figure 1** demonstrates the typical workflow of the system. Also, if any feature additions are made in the future to any one module, the designer of these additions will not have to worry about any part of the other modules.

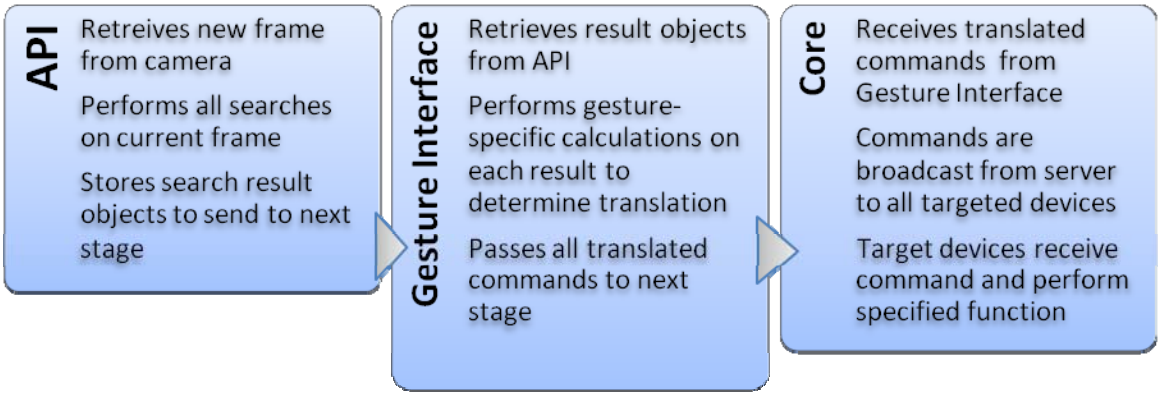

Figure 1 - **A workflow diagram for the HACK project showing the three components and their major roles within the system.**

Each component in the HACK system is a standalone unit. The results from one phase are not specifically packaged for the next phase. The receiving stage simply picks and chooses which of the results are of

interest and ignores the rest. In this way, each component becomes both adaptable and extendable for any system that future designers may conceive.

API

The Kinesics API encapsulates the functionality to detect rigid shapes and track non-rigid, articulated, objects by way of the previously discussed Viola-Jones and Flock of Features algorithms, respectively. It is an object-oriented API designed to ease the detection and interpretation of gestures formed from the combination of detected shapes that are in motion. The Kinesics API consists of several classes that collectively handle the process of detection and tracking. Figure 2 shows the class structure diagram.

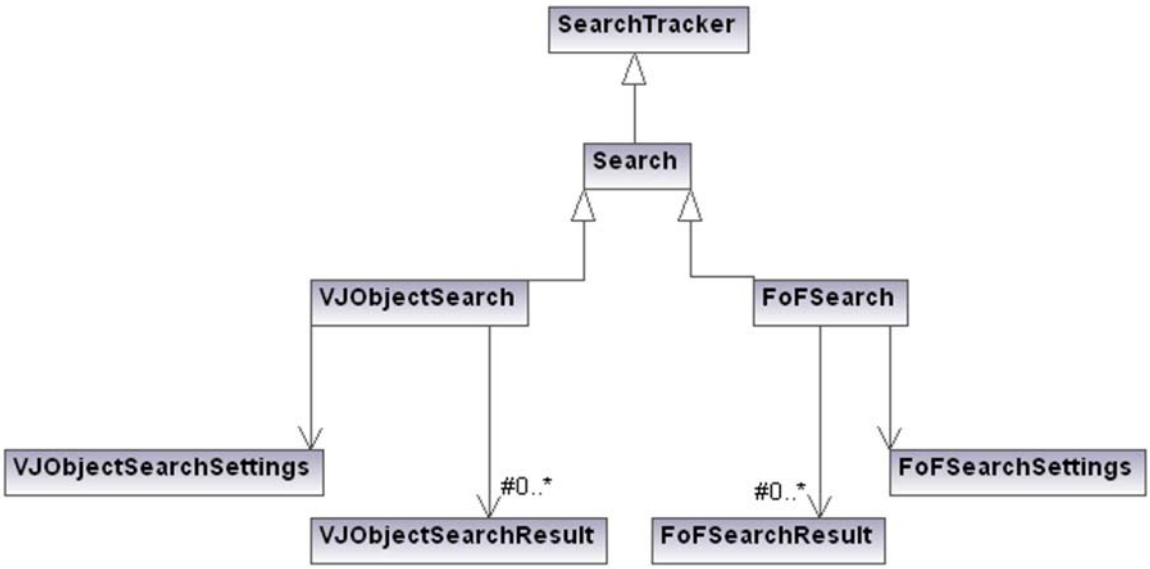

Figure 2 - **The class structure for the Kinesics API shows the major components of object searching, detection, and tracking.**

The top-level entity is an object tracking management class called **`SearchTracker`**. Instances of this class are responsible for providing frames of video to associated detection objects, providing a consistent detection rate by enforcing a specific frame-rate, and enabling or disabling search objects. Search objects are derived from the **`Search`** abstract class. The HACK project employs two search object classes—**`VJObjectSearch`** and **`FoFSearch`**—which perform Viola-Jones object detection and Flock of Features object tracking, respectively. Each search class type has a unique settings class and result class, derived from the **`SearchResults`** abstract class. The Viola-Jones search class contains **`VJObjectSearchSettings`** and **`VJObjectSearchResult`**, and the Flock of Features search class contains **`FoFSearchSettings`** and **`FoFSearchResult`**. These settings classes provide methods to adjust certain parameters and thresholds for each specific search. The parameters can be altered to adjust many aspects of each search, including sensitivity and scale. The results classes house vectors of objects that have been detected in the frame by the containing search classes. These result objects are then passed to their respective handler methods for interpretation.

Core

The Core is the device control subsystem of the HACK project. It consists of the central server, the remote microcontrollers, the peripheral devices that the system is controlling and the means by which to control these devices. Figure 3 visualizes the general workflow of the Core unit.

The server will house the API and the Gesture Interface applications. The server is responsible for broadcasting the control signals to the respective microcontroller units for further processing.

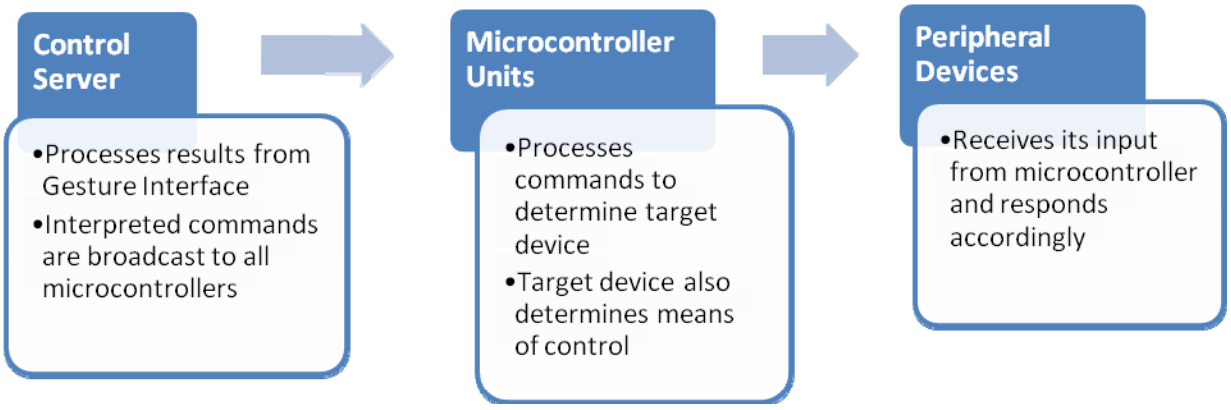

Figure 3 - The workflow for the Core component shows how the interpreted commands are passed on to eventually control the peripheral devices.

Once the control signals have been interpreted and processed by the server, they are sent out to the remote microcontroller units. These units interpret the commands from the server and determine for which peripheral device they are intended.

The device chosen will also determine the method by which this information can be sent. For example, if the command is for a television, an IR transmitter will be necessary to communicate the command to the television. Other peripheral devices may require various forms of communication, such as another Bluetooth connection or perhaps serial communication via USB.

The modularization of the hardware allows for a flexible system that decouples the Kinesics Interface from the device control implementation. This gives the project the extendibility mentioned earlier. With this in mind we can see that if a different hardware implementation is desired, the only modifications need be made to the Core and the API and Gesture Interface can remain nearly untouched.

Gesture Interface

The purpose of the Gesture Interface layer is to employ the API and interpret human-hand gestures to control a variety of peripheral devices. Gesture Interface consists of handler methods that monitor the Kinesics API results for object and movement combinations that constitute a given gesture. These

gestures and motions must have an intuitive semblance to the command that they perform in order to provide the user with a more pleasant experience.

The Gesture Interface is responsible for the interpretation of objects and motions detected by the API. When a specific gesture has been detected, the Gesture Interface will pass control to the handler method that has been implemented distinctively for that gesture. The handler method then performs any calculations or information forwarding that is deemed appropriate by the handler designer.

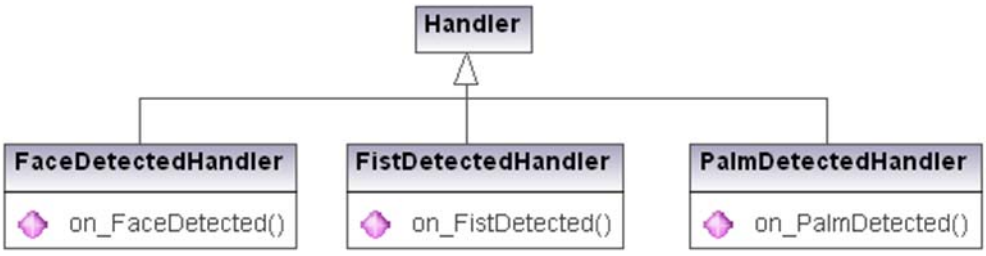

Figure 4 - An example implementation of the gesture interface showing handlers for face detection, fist detection, and palm detection. Each derived handler class implements an operation to be executed whenever that event occurs.

This component consists of a base **`Handler`** class that implements the activities that are common to any handler that a designer may implement. Gesture-specific handlers may then be extended from this base class to process that gesture as seen fit by the designer. A sample implementation is shown in Figure 4.

Architecture Summary

With the abstract system in place, we can now move forward and put real-world devices to work in this system. The next section gives specific devices and protocols used in the HACK system for its development.

Implementation

Now that the underlying architectural concepts have been discussed, we take the next logical step and discuss the physical implementation of these concepts. Again the system is decomposed into its three major modules and each is discussed in detail.

Overview

In order to achieve the goals set out for the HACK system, there are a few criteria that must be met by the practical implementation. First, as this system is to be run on a computer environment, it must be as platform-independent as possible so as not to shut out any potential user or developer. Second, the end-user should not have to be concerned about the complex underlying structure of both the software

and hardware. The system should provide a smooth and intuitive experience for the user. This places a red flag on system performance. Any delay from real-time must be imperceptible or trivial to the user.

This also places a burden on the designer to provide simple and natural gesture-to-command mapping. For example, the user raising his or her hand high in the air to turn the volume down on a stereo system is entirely counter-intuitive and is thus the antithesis of the HACK concept.

API

In order to provide the cross-platform functionality mentioned previously, the API is built on the OpenCV and QT4 frameworks. Since Qt4 provides the designer with both a programmable interface and an integrated development environment, it is an excellent choice to provide the portability required by the HACK system.

OpenCV is a sophisticated yet simple programmable interface for many computer vision applications. It masks the underlying detail from the designer to present them with an intuitive and straightforward interface to get started immediately with their vision project. This ease of use and the level of maturity of the framework make OpenCV an ideal choice for the HACK system. The OpenCV library not only facilitates gesture recognition and tracking but also the training of new gestures, giving it one more reason to be chosen for this application.

Core

The Core hardware devices must be portable and efficient. They must also be easily adaptable to a variety of configurations. For the microcontroller units, the Arduino family has been chosen for its reasonably low cost and simplicity in modification and adaptation.

The Arduino family has a large variety of microcontrollers that run the gamut of functionality, but the most basic of them can also be modified with modular functional components. In the case of the HACK system, Bluetooth 2.0 will be the primary means of server-controller communication. There exists both an Arduino processor with built-in Bluetooth support and a modular Bluetooth add-on. The basic Arduino and Bluetooth add-on have been chosen for the HACK system.

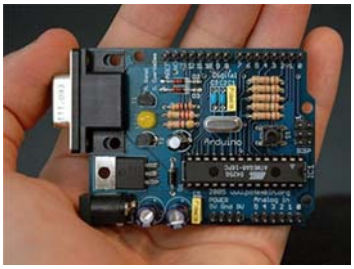

Figure 5 - The basic Arduino microcontroller used in the middle tier of the HACK Core.

Finally, the central server for the HACK system is the Asus EEEPC 1000H. This small, lightweight laptop computer has been chosen for an array of reasons, including its low cost, its lightweight and compact frame, and the capable processing power it contains.

Figure 6 - The lightweight, compact EEE PC 1000H that is the central server for the HACK system.

Gesture Interface

The Gesture Interface needs to be cross-platform capable just as the API is. For this reason it is built on C++ using the same OpenCV and Qt4 libraries as the API. The Gesture Interface employs a Signal-and-Slot architecture. This is an event-driven architecture in which various signals are mapped to specific slots. When a slot receives a signal, that signal's specific event handler is executed. One signal can be assigned to multiple slots and one slot may be listening for several signals. This setup provides an excellent framework for the gesture recognition interface since it is a real-world analog to an event-driven system in which many separate inputs may need to be processed in the same manner

As a demonstration of this particular HACK system implementation, an iTunes gesture control system was developed. This system recognizes a closed-fist gesture from the user. The motion of this gesture is then tracked to determine the control signal to send to iTunes. Raising the fist upward increases the volume that iTunes is playing slowly, and, conversely, lowering the fist lowers the volume. When the user drags the gesture to the left or right, the current song is changed to the previous or next track, respectively.

Difficulties

Thus far, some shortcomings in the implementation have been uncovered. While the component designs are believed to be effective and adequate, there are more subtle problems that need to be addressed before continuing new feature development.

First, the Bluetooth communication between the Arduino and the EEE PC has proved to be difficult. The devices can detect each other, but they are unable to communicate for reasons yet to be found. If this issue cannot be resolved in a timely fashion, an alternative communication scheme must be in put in place.

Second, the Flock of Features algorithm is not executing correctly when it is placed in to the API framework. As a standalone program, the Flock of Features runs exactly as it should. There are some unseen errors when the Flock is incorporated into the HACK API, and these issues are being looked in to as earnestly as possible.

A third shortcoming of the application as a whole is the time constraint of training a new gesture. While it is almost trivial to add a new gesture to the HACK API, training the cascade for that gesture takes hours, possibly days, of processing time on even a high-performance computer. Until newer training techniques are developed, the solution to this problem is out of the scope of this project, so it must be handled as is.

Cost Analysis

The HACK system provides a low-cost application for the Kinesics API. The current implementation yields a system for less than $500, as shown in Table 1.

Item	Cost
EeePC Notebook	$ 350.00
BlueSmirf Bluetooth Module	$ 50.00
Arduino Duemilanove Microcontroller	$ 30.00
Assorted electronics	$ 10.00
Apple Remote	$ 20.00
Total	**$ 460.00**

Table 1 - Cost analysis for the HACK system.

Project Summary

As we have seen throughout this document, the HACK system provides a simple and extendible platform on which to develop gesture-controlled systems. The open source, cross-platform API abstracts many of the tougher details of computer vision and machine learning to present the designer with an effective interface to rapidly develop new applications. The modularity of the system as a whole provides easily extendible and pluggable interfaces for adding or removing any features potential designers may desire. The realm of gesture control is only beginning to grab a foothold in the computer vision industry; projects like HACK can provide a strong foundation on which to build new, creative computer vision solutions.

References

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV.* O'Reilly Media, Inc.

Colorado State University. (2005). *Writing Guides*. Retrieved March 26, 2009, from Writing@CSU: http://writing.colostate.edu/guides/research/observe/

Franklin, C., & Layton, J. (2000, June 28). *How Bluetooth Works*. Retrieved March 26, 2009, from http://www.howstuffworks.com: http://electronics.howstuffworks.com/bluetooth3.htm

Hertzmann, A. (2008). *CSC411 Machine Learning and Data Mining.* Retrieved March 26, 2009, from University of Toronto Dynamic Graphics Project: http://www.dgp.toronto.edu/~hertzman/courses/csc411/fall_2008/lectureNotesWeb/AdaBoost.pdf

Hewett, Baecker, Card, Carey, Gasen, Mantei, et al. (2008, April 11). *Curricula for Human-Computer Interaction*. Retrieved March 26, 2009, from ACM SIGCHI: http://sigchi.org/cdg/cdg2.html#2_1

Notes

1. UML diagramming was done with the trial version of Altova's UModel software, available at http://www.altova.com/.
2. The Qt4 application framework is open source and cross-platform. The framework and IDE are available at http://www.qtsoftware.com/products.