Finger Gesture Recognition with Microsoft Kinect and CogniMem CM1K Pattern Matching Chip

Chris McCormick, Bill Nagel, Avinash Pandey
CogniMem Technologies Inc., Folsom, CA USA
http://www.cognimem.com
{chris.mccormick, bill.nagel, avinash.pandey}@cognimem.com

Abstract

The release of the Microsoft Kinect™ and Microsoft SDK for Kinect™ have made gesture recognition accessible by providing accurate skeleton tracking information down to the location of a person's hands. Notably missing from the skeleton tracking data, however, are the positions of the person's fingers. Recognizing the arrangement of the fingers on a person's hand has applications in recognizing hand gestures such as sign language, as well as simulating user inputs that are normally done with a mouse or a button on a controller. Tracking individual finger joints with the Kinect™ poses many challenges, including the limited resolution of the depth camera, and the possibility for fingers to occlude each other, or be occluded by the hand. By using pattern matching to recognize the overall shape of the hand, rather than attempting to identify the individual fingers, we can recognize finger gestures robustly using the existing Kinect hardware. In this paper, we present an approach for recognizing pre-trained finger gestures with a level of accuracy suitable for commercialization, and describe our demonstration of this algorithm.

1. Introduction

The lack of finger recognition with the Kinect™ creates a notable gap in the abilities of the system as compared to other motion devices which incorporate buttons such as on the Nintendo Wii™ and PlayStation® Move. Currently, on the Microsoft Xbox360™ with Kinect™, the on screen cursor tracks the position of the player's hand effectively, but selecting an object is done by positioning the hand over an object and waiting for a short timeout. There is no option for quickly selecting an item, or for doing drag-and-drop operations.

Game developers have designed games for the Kinect around this omission by focusing on titles which recognize overall body gestures, such as dancing and sports games. As a result, there exists an untapped market of popular games which lend themselves to motion control but require the ability to quickly select objects or grab, reposition, and release them. These titles are instead released on competing platforms such as the PC with mouse input, or on the Wii by using the buttons on the Wii remote.

The ability to click on objects as well as to grab, re-position, and release objects is also fundamental to the user-interface of a PC. Performing drag-and-drop on files, dragging scrollbars or sliders, panning document or map viewers, and highlighting groups of items are all based on the ability to click, hold, and release the mouse.

Presumably these features have and continue to be omitted on the Kinect because the depth camera is not deemed to have sufficient resolution to perform joint tracking of the fingers using the skeleton tracking algorithm.

Skeleton tracking of the overall body with the Kinect has been implemented successfully by Microsoft and others. One open source implementation (https://github.com/joaquimrocha/Skeltrack) identifies the joints by converting the depth camera data into a 3D point cloud, and connecting adjacent points within a threshold distance of each other into coherent objects. The human body is then represented as a collection of 3D points, and appendages such as the head and hands can be found as extremities on that surface. To match the extremities to their body parts, the proportions of the human body are used to determine which arrangement of the extremities best matches the expected proportions of the human body. A similar approach could theoretically be applied to the hand to identify the location of the fingers and their joints; however, the depth camera lacks the resolution and precision to do this accurately.

To overcome the coarseness of the fingers in the depth view, we will use pattern matching to recognize the overall shape of the hand and fingers. The silhouette of the hand will be matched against previously trained examples in order to identify the gesture being made.

The use of pattern matching and example databases is common in machine vision. An important challenge to the

Figure 1: The hand is isolated from its surroundings using the depth data.

Figure 2: A small subset of examples used to train the chip on an open hand. During learning, only examples which the chip doesn't already recognize will be stored as new neurons.

approach, however, is that accurate pattern recognition can require a very large database of examples. The von Neumann architecture is not well suited to real-time, low-power pattern matching; the examples must be checked in serial, and the processing time scales linearly with the number of examples to check. To overcome this, we will demonstrate pattern matching with the CogniMem CM1K (http://www.cognimem.com/products/chips-and-modules/CM1K-Chip/index.html) pattern matching chip. The CM1K is designed to perform pattern matching in full parallel, and simultaneously compares the input pattern to every example in its memory with a response time of 10 microseconds. Each CM1K stores 1024 examples and multiple CM1Ks can be used in parallel to increase the database size without affecting response time. Using the CM1K, the silhouette of the hand can be compared to a large database of examples in real-time and low-power.[1]

1.1. Hand Extraction

The Microsoft Kinect SDK provides skeleton tracking information which identifies the coordinate of the hand joint within the depth frame. Within a small square region of the depth frame around the hand, we exclude any pixels which are outside of a threshold radius from the hand joint in real space. This allows us to isolate the silhouette of the hand against a white background, even when the hand is in front of the person's body (provided the hand is at least a minimum distance from the body). See Figure 1.

1.2. Training the CM1K

Samples of the extracted hand are recorded in different orientations and distances from the camera (Figure 2). Recorded examples are categorized by the engineer, and shown to the chip to train it. As we repeatedly train and test the system, more examples are gathered to improve its accuracy.

The chip uses a Radial Basis Function algorithm to learn examples. For each example input, if the chip does not yet recognize the input, the example is added to the chip's memory (a new "neuron" is committed) and a similarity threshold (referred to as the neuron's "influence field") is set. The example stored by a neuron is referred to as the neuron's model.

Inputs are compared to all of the neuron models in parallel. An input is compared to a neuron's model by taking the Manhattan (L1) distance between the input and the neuron model. If the distance reported by a neuron is less than that neuron's influence field, then the input is recognized as belonging to that neuron's category.

If the chip is shown an image which it recognizes as the wrong category during learning, then the influence field of the neuron which recognized it is reduced so that it no longer recognizes that input.

1.3. Demonstration

The demonstration will consist of a Microsoft Kinect sensor, a television or monitor, and a CogniMem hardware evaluation board, all connected to a single PC. Software on the PC will extract the silhouette of the hand from the Kinect's depth frames using the Microsoft Kinect SDK, and will communicate with the CogniMem board to identify the hand gesture.

The mouse pointer on the PC will be controlled by the user's hand, with clicking operations implemented by finger gestures. Several gestures will be demonstrated as methods of user input, including the ability to click on objects, grab and reposition objects, and pan the screen. The user will be able to use these gestures to interact with various software applications, including both video games

[1] The CM1K can also obtain depth information using only CMOS sensors, pattern matching, and stereoscopic triangulation.

and productivity software. The demonstration will be open for anyone to try (not just the developers) in order to demonstrate the robustness of the system.