Dynamic Vision Sensor Camera Based Bare Hand Gesture Recognition

Eun Yeong Ahn[1], Jun Haeng Lee[2], Tracy Mullen[3], John Yen[4]

Information Sciences & Technology
The Pennsylvania State University
University Park, PA, USA
{eua121[1], tmullen[3], jyen[4]}@ist.psu.edu

Advanced Samsung Institute of Technology
Samsung Electronics
Yongin-si, Gyeonggi-do, Republic of Korea
junhaeng2.lee[2]@samsung.com

***Abstract*— This paper proposes a method to recognize bare hand gestures using a dynamic vision sensor (DVS) camera. Different from conventional cameras, DVS cameras only respond to pixels with temporal luminance differences, which can greatly reduce the computational cost of comparing consecutive frames to track moving objects. Due to differences in available information, conventional vision techniques for gesture recognition may not be directly applicable in DVS based applications. This paper attempts to classify three different hand gestures made by a player during rock-paper-scissors game. We propose novel methods to detect the point where the player delivers a throw, to extract hand regions, and to extract useful features for machine learning based classification. Preliminary results show that our method produces enhanced accuracy of hand gesture recognition.**

Keywords-Dynamic Vision Sensor Camera; Bare Hand Gesutre Recognition; Feature Extraction

I. INTRODUCTION

Tracking moving objects and recognizing gestures using cameras becomes increasingly possible with the growth in computing power. Conventional cameras have a frame-based architecture where a series of snapshots is taken at a constant rate. To track a moving object, these consecutive frames are compared to find temporal changes, which is a computationally expensive task.

Recently, Delbruck's group at ETH in Zurich developed bio-inspired camera called Dynamic Vision Sensor (DVS) that responds asynchronously to pixels that have temporal changes in intensity [6, 8]. As we will describe in detail in the next section, figure 1 shows those pixels with changes in luminance intensity during 20ms. Since temporal differences are detected at hardware level, it becomes easy to tract moving objects with less computational cost. However, due to different architectures and data representation within the DVS camera, it is not straightforward to apply conventional algorithms for tracking or recognizing objects to applications based on DVS camera. Some features such as the hue of the pixel that can be easily obtained from the conventional camera are not available in DVS cameras. In contrast, new features that were not available or hard to obtain in the conventional camera become available or easily accessible in DVS cameras. The purpose of this paper is to explore the properties of DVS cameras to find useful features for machine-learning based classification.

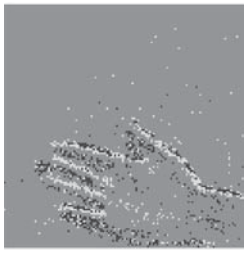


Figure 1. Output of a DVS camera

Our problem domain in this paper is the rock-paper-scissors game. Rock-paper-scissors (RPS) is a hand game in which two or more players can participate. Players start with their fist back towards their shoulder and then move their forearm down to deliver a throw. A throw consists of one of the hand poses: rock, paper, or scissors as shown in figure 2. Our goal is to classify the different hand poses made by a player. Classifying throws involves several subtasks such as locating the point where the player delivers a throw, recognizing hand regions, and extracting features to be used for machine-learning based classification. Thus gesture recognition for the rock-paper-scissors game must consider both static and dynamic hand postures. We suspect that rock-paper-scissors game is a good initial application domain since the method for recognition of throws can be generalized to real world problems such as reading sign languages or gesture input systems [9].

Figure 2. Three Possible Throws: Rock, Paper, and Scissors

This paper is organized as follows. Next section describes the properties of the DVS camera, and then section 3 explains the rock-paper-scissors game. Section 4 describes a novel method for detecting a point where a throw is delivered, and section 5 describes a machine learning based method to classify throws, which includes algorithms to extract hand region, and features from the hand. In section 6, preliminary results are presented, and our methods are discussed in section 7. Finally, section 8 concludes the paper with introducing possible extensions of this study.

978-1-4244-9915-1/11/$26.00 ©2011 IEEE

II. Dynamic Vision Sensor

Most of video data we use nowadays has a frame-based architecture that consists of a sequence of snapshots. Recently, Delbruck's group at ETH in Zurich developed bio-inspired dynamic vision sensor (DVS) camera whose pixels respond (spike) asynchronously to relative changes in luminance intensity [6, 8]. They also provided a java-based package called jAER which includes a function of displaying of the output of DVS camera and enables researchers to develop real time processing such as noise filtering. They say that "*the camera is bio-inspired in a sense that reflects three properties of biological vision: its sparse, event-based output, its representation of relative luminance change and its rectification of positive and negative signals into separate output channels*." Since discussion on the camera at the hardware level is out of our scope, this section mainly focuses on describing the representation of the data and the properties of DVS cameras.

The output of the DVS camera is a stream of *address events* each of which includes pixel address (x and y) within 128 by 128 pixels, and the changes of luminance intensity (on or off). This representation is called *address-event representation* (AER). The pixel has off-event if the intensity of the pixel reduces, and on-event if the intensity of the pixel increases. In other words, if the object is lighter than the background, the pixels where the object appears have on-events and the pixels where the object disappears have off-events. If the scene remains static, no event occurs. In a stream, events are ordered in time. Figure 1 shows the events that occurred during 20 ms of the downward hand movement, where black and white points represent either on- or off- events. It is displayed by using jAER which segments a recorded stream data into *frames* of a constant size, e.g., 20ms. In other words, events within 20ms are augmented to produce a frame, and the size of the frame is user-defined.

For the same data, if the size of the frame increases, the number of events in a frame will also increase, and consequently the computational cost will increase. If the frame size is too big, the frame is likely to contain old events, and the boundary of the moving object becomes blunt. If the size of the frame is gets smaller, on the other hand, the number of events within a frame will decrease and less computational power will be required. However, too small size for the frame will make it hard to recognize the shape of objects.

Compared to conventional cameras that produce information for both moving objects and backgrounds, DVS camera only response to pixels with temporal changes in luminance intensity. This novel camera can reduce the amount of redundant data, and lessen the computational cost of comparing different frames. Moreover, by using the temporal contrast rather than absolute luminance, the process can be less sensitive to the luminance conditions. Since the events are sent asynchronously, in addition, DVS data preserves accurate timing information. These properties of DVS camera will be particularly useful in applications such as tracking fast moving objects, traffic data acquisition, moving object in surveillance camera, etc [6, 8].

Although new data representations and properties of DVS data can ease some tasks such as tracking moving objects, it can also bring challenges to other tasks such as recognition of still objects. The reasons are that no event will occur if there is no moving object, and that hue information is not available in DVS cameras. Some features used by conventional methods can be hard to obtain or cannot be obtained from DVS cameras. On the other hand, there might be new features that were non-trivial to obtain in the frame-based camera but easy to obtain in DVS cameras. This paper explores properties of DVS camera to find new possible features that can be used for the recognition of different hand gestures.

III. Problem Definition

In each round of the rock-paper-scissors (RPS) game, each player delivers one of the three possible throws: rock, paper, and scissors as shown in figure 2. The goal of our paper is to classify these three different hand poses. This section briefly describes the terminology and rules of the game according to the world RPS society (http://www.worldrps.com/), and derives subtasks require to recognize the player's hand gesture.

According to the official rules of the RPS game, there are four phases in each round of the game: beginning, priming, approach, and delivery. In the beginning phase, all players put their fist back toward their shoulder, and a player indicates the start of the game by audible or visual signal. In the priming phase, players have their hand closed in a fist form, and then shake their hands vertically. In the approach phase, the hand is moving downward from the highest position of the last prime, and the player delivers the throw (rock, paper, or scissors) prior to the completion of the approach phase. The approach phase ends when the forearm is at a 90-degree angle at the upper body. Delivery refers to the point where player stop moving until an agreement about who has won is made.

Since the focus of this paper is to recognize the hand gestures of a single player, we simplify the original game, so that a round consists only of the following phases: (1) Starting: The player starts with a fist back toward his/her shoulder (figure 3(a)). (2) Approaching: The hand moves down to deliver a throw (figure 3(b-d)) (3) Delivering: A throw is delivered, and the hand stops moving for a second at a 90-degree angle at the upper body (figure 3(e)). (4) Returning: If there is successive round, the player goes back to the starting phase by moving his/her hand upward (figure 3(f-g)). Note that the hand stops moving at the starting and the delivering phases. Also note that the hand pose is changed from a fist to one of throws during the approaching phase. To make problem simpler, we also assume that the player's hand is located at the right side of the camera, and the palm faces toward the camera. Only some portion of the player's forearm is allowed to be presented in the screen. However, some of these restrictions such as the location of forearm can be easily relaxed by applying simple methods such as rotation of the frame.

To recognize a throw, the point where the player delivers a throw should be detected first. We call this point as *delivery point*. In terms of vision terminology, detecting the *delivery point* can be thought of as detecting a useful hand posture from hand gesture. Whereas hand posture refers to a static hand pose without involvement of movement, hand gesture refers to a

sequence of hand postures involving dynamic movement [5]. Detecting the *delivery point* is described in section 4.

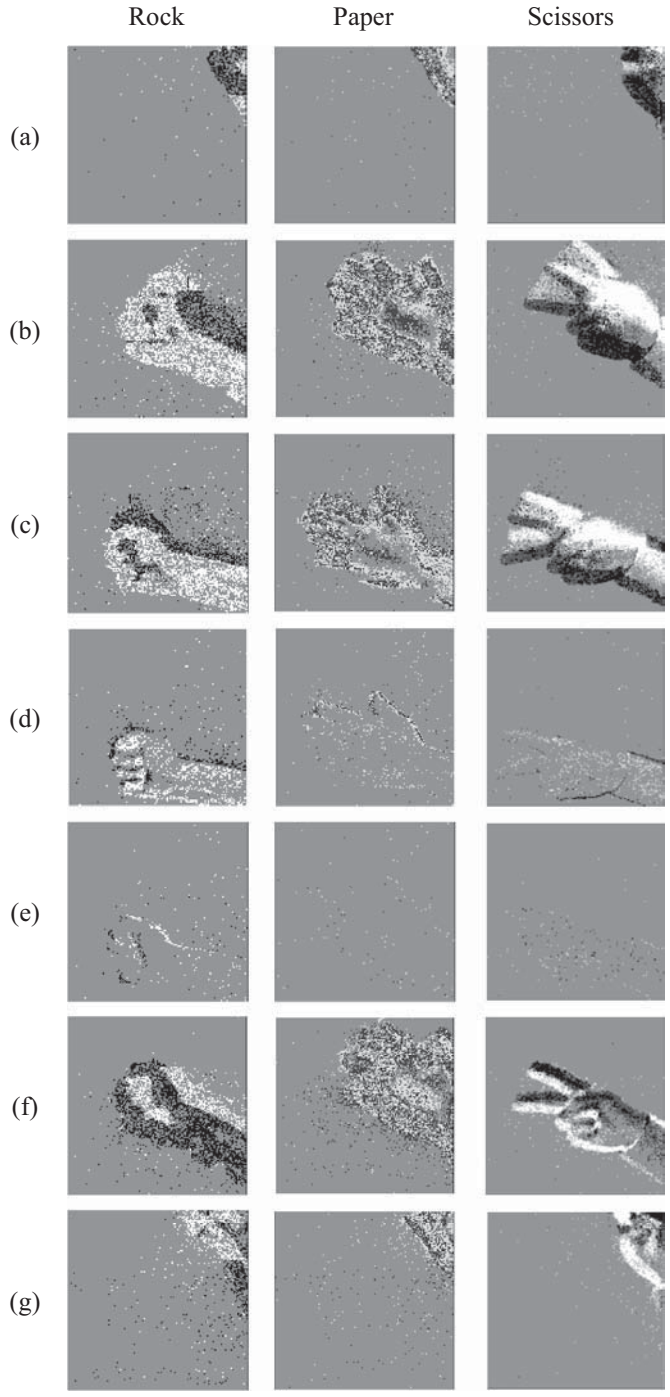


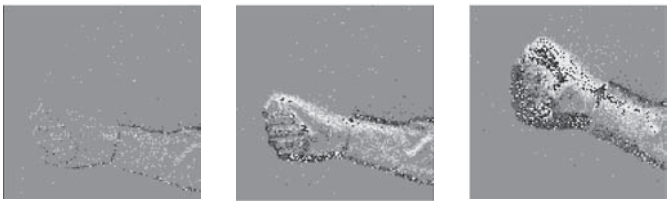
Figure 3. Snapshots of rock-paper-scissors

Once the *delivery point* is detected, we can classify different throws using only a single posture captured at the *delivery point*. Previous literatures on using frame-based cameras discuss two main approaches for bare hand gesture recognition: model-based and appearance-based [7]. In model-based research, people try to model hands by using 27 degree of freedoms [3], while the appearance-based approach attempts to classify images based on shapes inferred from skin colored regions. Our approach is similar to the appearance-based approach in a sense that a hand posture is classified based on the hand shapes. Since DVS cameras do not provide hue information, however, the hand shape is inferred from different methods rather than using skin colored region, which is discussed in section 5.

IV. Detecting Delivery Point

The *delivery point* refers to a point where the player delivers a throw. As we discussed in the previous section, the movement of the hand is slower as the hand reaches the end of the approaching phase in order to stop moving at the delivering phase. Recall that DVS cameras only response to the pixels with luminance intensity, and send events of only those pixels. Since there is little movement of the hand at the *delivery point*, the number of events within a frame will be dramatically reduced. This can be seen by comparing figure 3(e) with figures 3(b, c, d). To find the *delivery point*, we track the number of events for each frame, and if the number of events in the frame is less than the given threshold N_{DLVR}, the frame is regarded as a *delivery point*.

If N_{DLVR} is too large, the frame detected as a *delivery point* can be located at the middle of the approach phase. In this case, the hand pose in the detected frame can be different from the actual hand throw. Even if the hand pose in the detected frame is the same as the actual throw, the frame may not be the appropriate one for detecting hand pose. The reasons are that the frame is likely to contain too many events, which will increase the computational cost, and the shape of the hand can be not clear by thickening the boundary as shown in figure 4(c). In contrast, if the size of N_{DLVR} is too small, as in figure 4(a), the frame detected as a *delivery point* may contain too few events to recognize the shape of hand. Therefore, N_{DLVR} needs to be carefully chosen. Figure 4 (a), (b) and (c) show the frames detected as a *delivery point* when N_{DLVR} is set to 1,000, 5,000, and 10,000 respectively.



(a) N_{DLVR} =1,000 (b) N_{DLVR} =5,000 (c) N_{DLVR} =10,000

Figure 4. The Frames at the Delivery Points

However, stopping the hand movement does not necessarily mean that the player is delivering a throw. The player can also stop moving their hand between the two consecutive rounds of the game. The main difference between stopping movement at the delivering phase and the starting phase in consecutive rounds involves the direction of hand movement before and after the phase. The hand moves down before the delivering phase, and moves up after the delivering phase. In contrast, the hand moves up before the starting phase, and moves down after the beginning phase. Therefore, we track the direction of movement for each frame, and the first frame with number of

events less than N_{DLVR} during the downward movement is considered to be a *delivery point*.

Compared to the conventional frame-based camera where the direction of the movement of objects can be found by comparisons of consecutive frames, it can be easily found in DVS camera by considering events only in a single frame. We discuss one possible way of finding the direction of movement for the RPS problem by comparing the *y*-addresses of events that occur at the beginning of the frame with those that occur at the end of the frame. Recall that an event stream is ordered in time, so that we can we average the *y*-addresses of events in the first and the last half of the frame separately, and then compare them. If the average of *y*-addresses in the last half of the frame is smaller than that of first half, the hand is regarded as moving down; otherwise, it is regarded as moving up. Another approach to detecting the direction of movement might be using the type of events (on/off). Recall that if the object is lighter than the background, the pixels ahead of the moving direction might have on-events while those behind the moving direction will have off-events. If we could know whether or not the object is lighter than the background, the direction of the movement can be obtained by comparing the average *y*-addresses of on- and off-events. Since the second method can be sensitive to the conditions of luminance requiring a way to recognize the relative brightness of the object, we use the first method to recognize the moving direction of a hand within a frame.

Although events in a frame are sufficient to recognize the moving direction of a hand during the frame size, e.g. 20ms, it might not be sufficient to discern the moving direction of a hand across several frames. If there is a little shake during the downward movement, for example, the hand might briefly move upward. One way to cope with this problem would be to increase the size of the frame, but that will cause other problems such as thickening the boundary of the object and increasing the computational cost as discussed in section 2. Another approach, which is used in this paper, is to consider the moving direction of the hand in a few prior frames, e.g. 5 frames, and determine the majority moving direction as the current moving direction of hand.

To summarize, the frame is regarded as a *delivery point* when it has the number of events less than the given threshold (N_{DLVR}) and the hand is moving downward, where the moving direction of the hand is determined by selecting the majority of moving directions of the hand in a few prior frames and the current frame.

V. HAND POSTURE RECOGNITION

Once the *delivery point* is detected, the frame at the *delivery point* is used to recognize hand pose. Since there are some noisy events, a noise filtering process is first conducted using connected component analysis [2, 4]. To apply connected component analysis, the stream of events of a frame is represented by a 128 by 128 matrix. Recall that there are two types of events: on- and off- events. Although a single matrix can be generated by ignoring the event types, instead two 128 by 128 matrixes each of which represents on- and off- events are used, and connected component analysis is applied to each matrix separately. Since events of the same type are likely to occur closely in space, we expect that the two matrix representation can filter out noisy events that are surrounded by events of a different type.

A. Hand Extraction

After the filtering process, the hand region is first extracted. Notice that, in our problem, some portion of the forearm is presented in the frame. Since the various lengths of forearm presented in the frame do not provide useful information to distinguish different hand postures, only the hand region is used for classification. To extract the hand, first the wrist point is found. To do this, we estimate the width of the hand along the horizontal axis, and then record changes in width from right to left (from forearm area to hand) to locate the point with the largest width increase. Specifically, it works as follows: (1) the frame is segmented into b_H bins along the horizontal axis as shown in figure 5, and the segmentation point which lies in between the *i*th bin and the *i*+1th bin is denoted as δ_i. The size of the bin is $|max(x) - min(x)|/ b_H$, where $max(x)$ and $min(x)$ represent the largest and the smallest *x*-address of events, respectively, and b_H is the number of bins. (2) Estimate the width of the hand for each bin, which is denoted by d_i for *i*th bin. (3) Calculate the width change for each segmentation point δ_i by subtracting the width of the hand in the left bin of δ_i (i.e.,by the width of the hand in the right bin of δ_i, i.e., $d_i - d_{i+1}$ for $0 \le i \le b_F - 1$. (4) From forearm area to hand, find the segmentation point p with the maximum changes. (5) All the events whose *x*-address is smaller than p are regarded as a hand region.

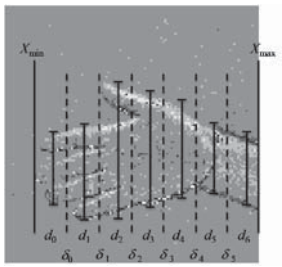


Figure 5. Hand Extraction

The simplest way to estimate the width of the hand for each bin might be to use the difference between the maximum and the minimum *y*-addresses of the events within the bin. We can also use standard deviation of *y*-addresses within the bin or apply *K*-means algorithm where $K = 2$ to find the *y* differences between the two centroids. It is expected that the latter two methods will perform better than the first method when there are many noisy events. Since significant noisy events are filtered out before the hand extraction process, we choose the computationally less demanding way of using the difference between the maximum and the minimum y-addresses. Notice that since the events are likely to be distributed at the boundary of objects, the process of estimating the width of hand does not involve any edge detection algorithm such as canny edge detection [4].

In addition, the number of bins, b_H, should be appropriately set. If b_H is too small, the actual width changes of the hand cannot be captured during the hand extraction process, and the algorithm is likely to detect a point which is not even close to

the actual wrist as a wrist point. Even if the detected wrist point is near the actual wrist, the extracted hand region can contain a large portion of forearm or exclude a large part of the actual hand region due to a large bin size. In contrast, if b_H is too big, the changes of width can be captured at the fine-grained level, but it will increase the computational cost. Also, by focusing on changes of width too locally, the algorithm can be prune to noise and fail to recognize the overall changes of width.

B. Feature Extraction

After extracting hand regions, we extract features for machine-learning based classification. In the machine-learning based methods, selecting good features is critical to the performance of learning algorithms [1]. In this paper, we attempt to extract features from which the shape of the hand can be inferred.

One simple feature which can represent the shape of a hand might be the width distribution across the hand. Similar to the method to extract the wrist point, we segment the extracted hand into b_F bins along the horizontal axis as shown in the figure 6(a), and then estimate the hand width for each bin. A sequence of hand widths shows how the size of width changes along the horizontal axis. Since the absolute value of width can be different depending on person and the distance between the hand and the camera, we use relative width, which can be obtained by dividing the absolute width of the bin by the sum of absolute widths over all bins. It is shown in equation 1, where relWidth(i) and absWidth(i) represent the relative and the absolute size of width at the ith bin, respectively. As we discussed previously, the size of width can be estimated by the maximum y-difference between the two data points, variance of y-addresses, or y-difference between the two centroids obtained by K-means where $K = 2$.

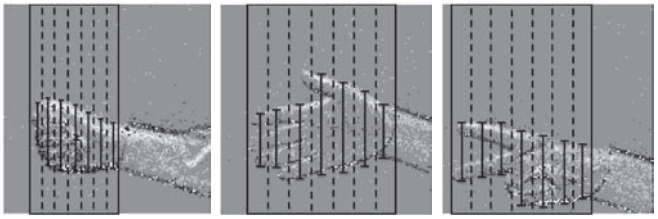
$$\text{relWidth(i)} = \text{absWidth(i)} / \sum_{\text{i}}^{b_F} \text{absWidth(i)} \qquad (1)$$

Three graphs in the figure 6(b) show the relative width of hand where horizontal and vertical axis represent the index of bin and the relative width, respectively. Absolute hand width within a bin is estimated by using the maximum y-difference, and each point in the graph is the average of 10 different hand postures with the same type. In rock pose data, the width increases at first and then gradually decreases, while in paper pose data the width gradually increases at the first few bins and then steeply increases toward the peak at the finger point. In scissors pose data, there is a concave point around at the middle.

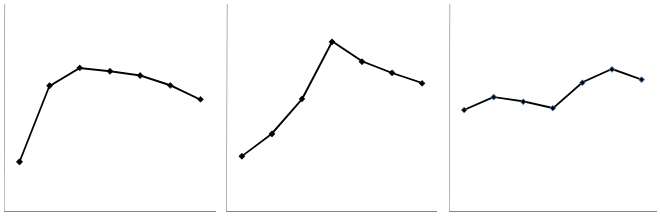
Similar to the process of hand extraction, the number of bins, b_F, should be appropriately set. If the number of bins is too small, key features cannot be captured. An extreme case is using one bin. Although computationally less expensive, the pattern of width and the hand shape cannot be inferred from only the average hand width. The opposite extreme case is using the number of column pixels of the abstracted hand data as b_F. Although the shape of hand can be represented at the fine-grained level, it is computationally expensive. More importantly, by focusing on too specific and too local patterns of data, we may fail to extract more general data patterns.

Although the changes of hand width can be used to classify the three different types of postures, this method may not be effective if there are more possible gestures such as stretching index, middle, and ring fingers. To extend the number of possible hand gestures in the future, more sophisticated method will be needed. One possibility is to model that problem such that each finger can have only one of two states: stretched or folded, so that most of the commonly used hand gestures are combinations of the states of all five fingers. As a first step toward the recognition of the states of fingers, we attempt to recognize the number of fingers in each bin without distinguishing whether a finger is stretched or folded or by specifically identifying the fingers in each bin.

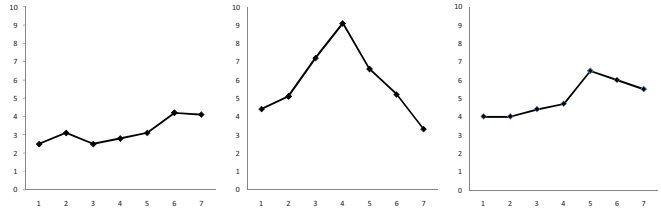
To find the number of fingers within a bin, connected component analysis is used [2, 4]. Instead of filtering out the component with the smallest number of events as we did in the filtering process, the number of connected components for each bin is counted. Recall that the pixels where object disappears and where object appears have different types of events. If the hand is lighter than the background and is moving down, the upper side of the hand or finger is likely to have off-events while the bottom side is likely to have on-events. Therefore, we expect that the number of connected component for each bin is ideally the twice of the number of fingers in the bin, since there will be two connected components for each finger.



(a) The Frames at the Delivery Point



(b) The Width Distribution



(c) The Number of Connected Components

Figure 6. The Width Distribution and the Number of Connected Components

Figure 6(c) shows the number of components for each bin when N_{DLVR} is set to 7,000. Each point is the average of 10 different hand postures of the same type. In the scissors pose data, there are about four connected components in the first few bins on average, which reflects the existence of two fingers on the left side of the frame. Although the exact number of fingers at the middle of palm where ring and little fingers are crossed

with thumb is not clearly shown, the important overall tendency of scissors is shown. The paper pose data shows that the number of fingers on the first bin is about two, or half of the number of connected components, which is four in this case, and the number of components increases along the horizontal axis until it meets the peak point where the number of connected components is about 9. Compared to the paper and scissors poses, the number of fingers is not clearly shown in the rock pose data. Although we expected that the number of connected components in the first few bins would be about 10, only two or three connected components were found. The reason is that the fingers are so closely attached or overlapped that the points at the boundary of the fingers are likely to be connected.

Next, as a step toward the recognition of the state of fingers, we attempt to recognize whether any of the fingers except the thumb is stretched or not. This could be easily recognized by comparing ratio of the length from the tip of fingers to the thumb to the length from the thumb to the wrist as shown in figure 7. We can roughly estimate the location of thumb by locating the column with the maximum width. We call this feature as *Horizontal Ratio* in the rest of the paper. Although simple, this method can be effective especially when there are different possible hand postures with similar silhouette. In figure 6(b), for example, rock and papers might have similar silhouette patterns since it only considers the relative hand width of along the horizontal axis. The graph in figure 7 shows that *Horizontal Ratio* for each type of data.

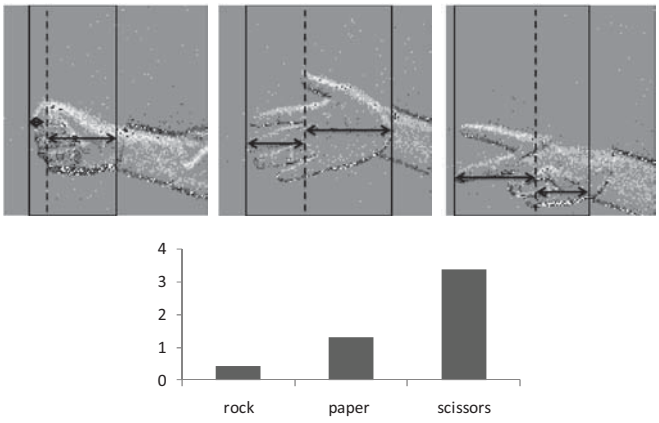


Figure 7. Horizontal Ratios

C. Classification

Once the features are extracted, a machine learning algorithm is applied to build a model for prediction. In this paper, we use naïve Bayes algorithm which is a simple version of Bayesian network with an assumption that all the features are independent given the class [1]. Despite of this strong assumption, naïve Bayes has proven to perform well in practice and requires less learning time than Bayesian network by using a fixed network structure. Although we can expect that there are dependencies between features extracted by the methods descried in the previous subsection, comparisons of our simple experimental results using Support Vector Machine, which is a state-of-art machine learning technique, has shown that the performance of gesture recognition in our application is more dependent on the features we use rather than the machine learning technique we employ.

VI. Experimental Results

We used previously recorded DVS camera video data with two types of data. In one type of video data, a player delivers the same throw 20 times consecutively, for each rock, paper and scissors hand gesture. In another type of video, a player delivers 40 random throws. In the random data, there are twelve rocks, fourteen papers and fourteen scissors, producing 40 rounds in total.

In the previous section, we discussed two possible schemes of extracting features: width-based and component-based. Also, we introduce a feature called *Horizontal Ratio* which estimates the ratio of the length of finger to the length of palm. In this section, we compare the accuracy of classifiers when width-based and component-based methods are used for extracting features, and analyze the effect of using *Horizontal Ratio* on the classification accuracy.

The first ten rounds of each rock, paper, and scissors data are used for training with manually provided labels. To test our model, we use the whole rounds of rock, paper, and scissors data and random data. Note that testing data with 100 rounds also includes training data with 30 rounds. We experimentally set $N_{DLVR} = 7000$, $b_H = 15$, $b_F = 7$, and the frame size = 20 ms. Table 1 shows the accuracy of classifier for each feature extraction method in different data sets. The last column represents the average accuracy of the classifiers which is the ratio of total number rounds that gesture is correctly classified to a hundred rounds. Width-based+*Horizontal Ratio* and component-based+ *Horizontal Ratio* methods use *Horizontal Ratio* as a feature in addition to seven features obtained by width-based or component-based method. In width-based+component-based+ *Horizontal Ratio*, features extracted from both width-based and component-based are used with *Horizontal Ratio*, which produces 15 features in total.

TABLE I. Classification Accuracies

Methods	rock	paper	scissors	random	Avg
width-based	90	95	90	72.5	84.0
component-based	85	95	90	77.5	85.0
width-based + *Horizontal Ratio*	85	95	90.0	75.0	84.0
component-based + *Horizontal Ratio*	85	100	100	80.0	**89.0**
width-based +component-based + *Horizontal Ratio*	90	100	100	72.5	87.0

According to the results, the component-based feature extraction method slightly performed better than width-based methods, and *Horizontal Ratio* could contribute to the enhancement the classification accuracy. We expected that combining width-based and component-based method by simply concatenating the features from both methods can enhance the accuracy of classifier, but the results have shown it is not necessarily true. Although it performs similar or better in rock, paper, and scissors data, its performance in random data

was worse than that of component-based, component-based+ *Horizontal Ratio* or width-based+ *Horizontal Ratio*. The reason might be that the increase of the number of features leads the model to be overfitted to the training data.

Figure 8 shows the confusion matrixes for each feature extraction method in two types of data. The actual throws are shown on the first row, and the predicted throws are shown on the most left column. For all methods, rock was hard to be correctly classified. Component-based and component-based+ *Horizontal Ratio* distinguish between paper and scissors and between rock and scissors slightly better than width-based methods. The slight enhancement of the performance of component-based methods might be achieved by looking at the data at the more fine-grained level, i.e., the number of connected components within each bin.

rock/paper/scissors

	r	p	s
width-based			
r	18	0	0
p	2	19	2
s	0	1	18
component-based			
r	17	1	1
p	1	19	1
s	2	0	18
width-based + *Horizontal* Ratio			
r	17	0	0
p	2	19	2
s	1	1	18
component-based + *Horizontal Ratio*			
r	17	0	0
p	1	20	0
s	2	0	20
width-based +component-based + *Horizontal Ratio*			
r	18	0	0
p	1	20	0
s	1	0	20

random

	r	p	s
r	4	0	0
p	6	14	3
s	2	0	11
r	6	1	0
p	5	13	2
s	1	0	12
r	5	0	0
p	7	14	3
s	0	0	11
r	5	0	0
p	6	14	1
s	1	0	13
r	4	0	0
p	8	14	3
s	0	0	11

Figure 8. The Confusion Matrixes

Although we expected that rock and paper can be more accurately classified through *Horizontal Ratio* especially in width-based methods, the results have shown that it is not necessarily true. However, in component-based methods, *Horizontal Ratio* could help distinguish not only between rock and paper but also between other types of gestures.

Our investigation on the process of feature extraction has shown that some frames at the *delivery point* contain so small number of events in the frame or around the boundary of objects, which makes it hard to recognize the shape of objects. It might happen if the player suddenly stops moving the hand or the size of hand appeared in the frame is so large compared to other data. This will be discussed in detail in the next section. We could also see that some portion of actual hand region is excluded as a result of hand extraction process by incorrectly detecting a wrist point. Such a problem is mainly caused by the player not putting his hand at the 90-degree angle at the upper body. However, our preliminary results were promising in a sense that about 89% level of accuracy was achieved by using *Horizontal Ratio* and component-based feature extraction method

VII. DISCUSSIONS

Although the promising results are shown, there is much room for improvement. This section describes the limitations of our approach and proposes possible extensions of the research to improve the performance of our system and to have our system more applicable to general applications.

From the investigation of the process of gesture recognition, we found that some frames at the *delivery point* have much less number of events than N_{DLVR}, which makes it hard to recognize the shape of the object. It could happen when the object suddenly stops moving. We expect that this problem can be partially solvable when a few frames prior to the *delivery point* are also taken into account during classification.

We also found that even if the number of events in the frame at the *delivery point* is about N_{DLVR}, such points were not enough to cover the boundary of the object in some data. It might be caused by setting N_{DLVR} as a constant. The number of events in a frame is not only dependent on the speed of the object, i.e., whether the object is moving or not, but also the size of object appeared in the frame. The difference of the size of objects appeared in the frame can attribute to the actual difference of the size of objects or the distance between the camera and the object. If the hand appeared in the frame is so big, more than N_{DLVR} of events will occur even when the hand is about to stop moving. Since N_{DLVR} is set as a constant regardless of the size of object in the frame, the frame detected as a *delivery point* in such data can have not enough number of events around the boundary of the hand, which makes it difficult to recognize the shape of hand. One way to cope with this problem would be using dynamic N_{DLVR} which updates its value adaptively depending on the size of object.

Moreover, our method will fail to correctly extract hand regions if the hand is not at 90-degree angle at the upper body. Although including forearm in the hand region might not be a serious problem, excluding some part of hand could be detrimental. Whatever methods we use to extract features, the features cannot represent the shape of hand appropriately if some parts of hand are missing. Therefore, it will be needed to accurately detect wrist point in more general positions.

Finally, using component-based method and *Horizontal Ratio* was our first step toward the recognition of the state of fingers: stretched and fold. Although counting the number of connected component could provide a rough idea on the number of fingers in each bin, the results were highly dependent on how N_{DLVR} was set. In rock data, for example, if N_{DLVR} is set too large, the boundary of object becomes blunt, and the events from different fingers are likely to be considered as connected, which yields the small number of connected components in the first few bins. In contrast, if N_{DLVR} is set too small, not enough number of events is distributed around the boundary of the object, which yields the same types of events from the same finger to be considered as different components. Although we believe that adaptively changing the threshold can help detecting the number of fingers more accurately, more elaborate methods will be required in order to ultimately detect the state of each finger.

VIII. Conclusions

This paper proposes a method to classify bare hand gesture using dynamic vision sensor (DVS) camera. Specifically, we focused on classifying three different throws delivered by a player playing rock-paper-scissors game. We first described the properties of DVS camera which only responds to the pixels with luminance changes. Then, we proposed a method to detect a *delivery point* where the final throw is made. Once the *delivery point* is detected, only the still image of hand at the *delivery point* is used for classification. The region of hand is first extracted by locating the wrist point where the changes of the width of hand is the biggest, and then we extract the distribution of width within the hand or the number of connected components for each segment as features. The experimental results were promising; using component-based method and ratio of the length of finger to the length of palm results in 89% of the accuracy. Component-based feature extraction method performed slightly better than width-based method, and the ratio of the length of finger to the length of palm could contribute to the enhancement of classification accuracy.

However, there is a lot room for improvement. We discussed that the threshold for detecting *delivery point* should be adaptively changing depending on the size of actual object, and that the hand region should be more accurately detected in a situation where forearm is not at 90-degree angle at the upper body. It is expected that studies on detecting the finger state (i.e., open or folded) goes a step further. This work will be based on using the component-based method and is expected to help make our system applicable in more general situations with numerous possible hand gestures. Furthermore, we plan to compare the classification accuracy and computational cost of using a DVS camera with those using conventional frame-based camera.

Acknowledgment

This paper is an extension of research done during the internship at Advanced Samsung Institute of Technology (SAIT). We acknowledge Dr. Tobi Delbruck for providing dynamic vision sensor cameras and researchers, Dr. Hyun-Suk Ryu, Dr. Chang-Woo Shin, Dr. Keun-Ju Park and Dr. Byung-Chang Kang, at Frontier IT lab in SAIT for useful comments. We also thank Dr. James Wang for the fruitful advice.

References

[1] I. Witten and E. Frank. Data Mining: Practical machine learning tools with Java implementations. Morgan Kaufmann, 2000.

[2] K. Suzuki, I. Horiba, and N. Sugie, “Linear-time connected-component labeling based on sequential local operations,” Computer Vision and Image understanding, vol. 89, pp. 1–23, 2003.

[3] L. Bretzner, I. Laptev, and T. Lindeberg, “Hand gesture recognition using multiscale color features, hierchical models and partical filtering,” in Proceedings of Int. Conf. on Automatic face and Gesture recognition, Wahsington D.C., May 2002.

[4] L. G. Spiro and G. C. Stockman, Computer Vision, Pentice Hall, 2001.

[5] P. Garg, N. Aggarwal and S. Sofat, “Vision Based Hand Gesture Recognition,” World Academy of Sciences, Engineering and Technology, vol. 49, pp. 972–976, 2009.

[6] P. Lichtsteiner, C. Posch, T. Delbruck, “A 128× 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor,” IEEE Journal of Solid State Circuits, vol. 43(2), pp. 566-576, 2008.

[7] Q. Chen, N. D. Georganas, E. M. Petriu, “Real-time Vision-based Hand Gesture Recognition Using Harr-like Features,” in Proceedings of Instrumentation and Measurement Technology Conference, Poland, May 2007.

[8] T. Delbruck, “Frame-free dynamic digital vision.” Proceedings of Intl. Symp. On Secure-Life Electronics, Advanced Electronics for Quality Life and Society, pp. 21–26, March 2008.

[9] V. I. Pavlovic, R. Sharma, T. S. Huang, “Visual interpretation of hand gestures for human-computer interaction a review,” IEEE Transactions on Pattern Analysis and machine Intelligence, vol. 19, pp. 677–695, Jul 1997.

[10] Y. Xu, J. Gu, and D. Wu, “Bare Hand Gesture Recognition with a Single Color Camera,” in Proceedings of the International Congress on Image and Signal Processing, China, 2009.