Vision based gesture recognition with Kinect sensor

Salle Dhali

April 26, 2015

Abstract

Gesture recognition and its implementation that support Human Computer systems are becoming very popular mode of interaction now a days. It allows to interfacing the man machine commutative information flow naturally. Vision based gesture recognition has the potential that can provide intuitive and effective interaction between man and machine .However there are not adequate tools and techniques that support for developing, detecting or executing these tasks. In this paper we will implement a prototype that facilitates recording data during building some action based activities captured by the Kinect sensor. We analyze those recorded clips and visualize the user interactions by recognition the gestures objects based on depth, IR and skeletal data. Kinect tools include an analysis feature, a time-line-based approach that manually or automatically can mark the recording sequences of clips. We will implement both discrete and continuous gestures by using AdaBoast machine learning approach to detect hands activities. Our result suggest that the learning mechanism can achieve more than 98% of confidence level of given gestures.

Keywords: Gesture recognition, Kinect, HCI, Machine learning, AdaBoost, Computer vision

1 Introduction

Gesture are generally used to communicate between people as well as interaction with machine (Gandhi et al., 2014). It is vital to develop hand gestures interface naturally for human computer interaction (HCI) where a recognized gesture may trigger a robot to perform some meaningful task or simply controlling a machine (Garg et al., 2009). The successful development for any Kinect applications is that letting the users the ability to communicate in a very natural way. The importance of gesture recognition robustly allows persons that are the controller of that devices.

The rapid advancement of Information Technology (IT) at present day we can expect computer integrated systems will be embedded to a wider extend in human lives. These emerges a new type of interaction technique rather than the existing one, as for instance present days the keyboard and mice the most popular HCI based mode of interaction between man and machine (Murthy and Jadon, 2009). However this devices limits the speed and natural user interfaces and Interaction of humans with computers using mouse or a touch-pad is now obsolete (Gandhi et al., 2014).

The development of user interface requires a good understanding of the structure of human hands to specify the kinds of postures and gestures (Gandhi et al., 2014). Feelings and thoughts can also be expressed by the gesture. Users generally use hand gestures for expression of their feelings and notifications of their thoughts (Murthy and Jadon, 2009).

However, it is not a trivial task to detect gesture robustly in a natural way. There are several research for detecting and analyzing for gestures from simple approach such as image processing in 2D to 3D or advance method such as clustering, histogram variation also applied. Neither of these process are simple to detect gesture. This paper aims to highlighting a simple approach for detection or recognition of hand gestures by applying a machine learning algorithm.

1.1 Problem statement

Kinect sensors to detect gestures is neither a simple task nor a trivial task to resolve. As for instance a simple punch gesture detection can be implemented by writing some piece of code which might work in a optimal situation. However, even the much sample code has probably four thresholds that need to be manually found and maintained and tuned as well (MSTech, 2014). To test the reliability of detection a wide range of people with different environment includes much more coded need to be added results excessive complicating factors. The problems raises as different skeleton data will be generated as people are wearing different cloths also Kinect Sensors are placed in different heights and tilt angels. There are some joints that might be occluded as a result of objects such as tables. Much more coding has to be written for detecting gestures in that real world scenarios.

Some of the common challenges for implementation of gesture detection and examining those data collected from Kinect sensors are also it requires time and adequate engineering skills as well as demanding tons of code writing. Kinect data are complex in nature as it generates twenty-five 3D joint orientation Degrre of Freedom (DoF) (Zhu et al., 2013) positions also the precision level is very low. The determination of the best detection threshold is very difficult. To detect a gesture it always requires previous frames examination which added additional latency. Tagging is an important part of analyzing the frames captured during the recording as it will identify the gesture that will act naturally as possible.

Previously researcher such as (Worsley and Blikstein, 2013) has proposed gesture detection and its utilization to multimodal learning analysis. They used many tools and techniques such as histogram variation, clustering algorithm and Euclidian to measure the distance vectors. They manipulate the objects by coding the objects and measure how the expert and novice users move different objects. In this paper we also aim to focus some of their works and try to detect gestures by using gesture builder which is pretty straight forward and allow to detect even complex gestures by using machine learning algorithm.

1.2 Aim and goals

The objective of this paper is to evaluate how the Kinect sensor can detect an implementation of gestures after being recorded some sample users through Kinect studio. The Ada boosting triggering algorithm will be used for machine learning and tagging the clips produces by the Kinect sensor. We will measure the confidence level of the gesture detection by utilizing the effective tools Visual gesture Builder (VGB) therefore the research questions will be:

Research Questions

- RQ1: How a data driven solution can be used to detect gestures through machine learning?
- RQ2: What are the confidence levels of those measured gestures?

2 Theoritical Background

2.1 Kinect sensors

Kinect by Microsoft is one of the most popular devices available on market among developers. This device was primarily intended for X-Box 360 video game console, few month after its release Microsoft announced Kinect for Windows (Jana, 2012). Kinect sensor is a physical device having depth sensing technology, a built-in color camera, an infrared (IR) emitter, and a microphone array. It enable us to sense the location and movements of people. The four

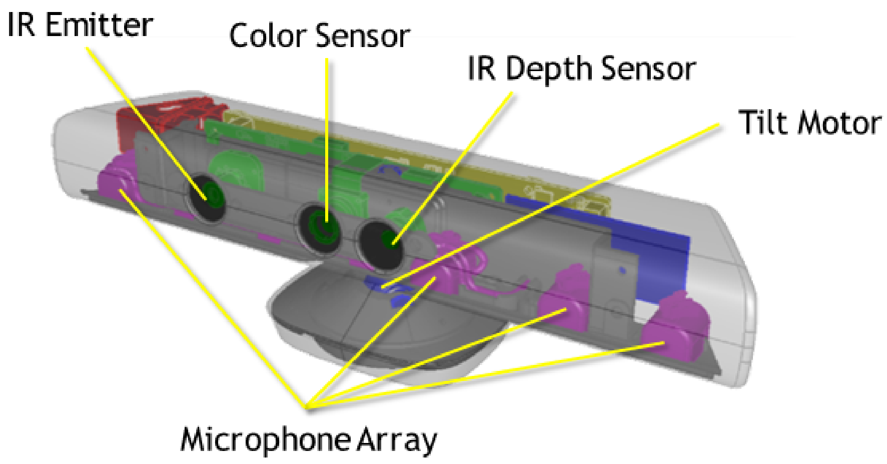


Figure 1: An overview of Kinect sensor(Courtesy: Microsoft)

microphone array allow us to capture recoring audio by 24-bit analog-to-digital converter (ADC) and signal processing including acoustic echo cancellation and noise suppression [1]. The newest Kinect sensor has three times higher depth fidelity and provides significant improvements in visualizing small to big objects more clearly which is shown in figure 1.

The Kinect for Window software development kit (SDK) 2.0 provides developers with necessary tools, drivers, APIs and device interfaces that allows the development of Kinect based applications easily. The body tracking features in SDK V.2.0, we can track up to six people and 25 joints person. The sensors together with the SDK, allow us to create and develop applications which are interactive and able to capture and as well as respond uses movement, gestures, speech commands results a naturall computer interaction [2].

2.2 Gestures and its detection

A gesture is an action or a motion that is intended to communicate feelings or intentions (Mazumdar et al., 2013). Gesture recognition aims to interpret human body motions via mathematical algorithms. The goal for recognition is that computer begin to understand human body signs as an standard input rather most commonly used primitive text and also GUI (Graphical user interfaces). The gesture recognition process must allow the computer system what should be performed by those specif movements of that gesture. Computer vision and image processing techniques can be used to accomplish those tasks. Gesture recognition allows humans to interact with machines in a natural way

[1]https://msdn.microsoft.com/en-us/library/jj131033.aspx
[2]http://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx

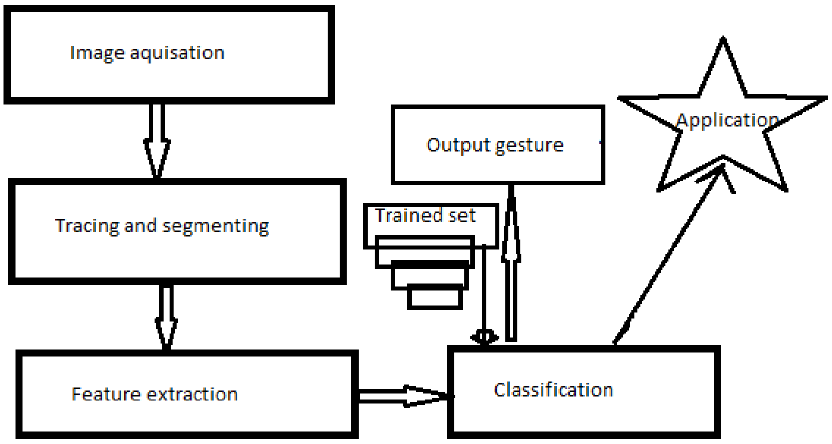


Figure 2: Block diagram of hand gesture process (Mazumdar et al., 2013)

without involving mechanical devices (Sánchez-Nielsen et al., 2004).

Gesture detection is the ability of a computer to understand human gestures as input. Gesture detection is still used in many technologies today, such as touch screens, computer mice, handwriting recognition and Kinect sensors. A gesture recognition process system diagram shows in figure 2.

Image frame acquisition is the early process technique for gesture recognition. A number of image frames are captured by camera in our case a Kinect sensor. Hand segmentation is to track the movement of the hand where each frame is processed separately before its analysis. Segmentation procedures partition an image into its constituent parts or objects. In image processing hand tracking is a high-resolution technique that is employed to know the consecutive position of the hands of the user and hence represent objects in 3D (Mazumdar et al., 2013).

After tracking it is important to extract the meaningful features from the collected data. The feature extraction used for instance image processing and pattern recognition is a form of dimensionality reduction process. When the input data is too large then the algorithm can be used to track the other part of the body. The input data and transform it into a set of features is called feature extraction (Phung et al., 2005). A classifier plays a significant role in gesture recognition process after the extraction of features. It is a statistical method that takes feature set 'as input and gives a class labeled output, which are required output gestures' (Gu and Bone, 1999).

2.3 Machine learning

Machine learning is the data processing ability for a computer that automatically recognize a complex pattern of data for instance face detection, speech recognition and AI games etc (MSTech, 2014). A computer can perform this sort of job by learning from the empirical data collection and the result it can generate by classifying the data it has not been observed yet. There exist many machine learning algorithms such as decision trees, neural networks, support vector machine (SVM), clustering, Bayesian networks and boosting. In this paper we will use AdaBoosting algorithm to detect gesture recognition.

3 Literature Review

The literature review provides in depth the numerous methods that many researchers have proposed in order to recognition of detection of gestures. Here we are going to provide some as follows.

For handling gesture recognition different types of technique and tools are used.Some are based on mathematical models such as Hidden Markov Model (HMM) and Finite State Machine (FSM) (Ibraheem and Khan, 2012). Some approaches are based on software computational methods such as fuzzy logic clustering, generic algorithms and Artificial neural Networks (ANN) (Stergiopoulou and Papamarkos, 2009). Hand gesture is an open research area to explore as human hand is very complex having different joints and links that forms the 27 degree of freedom for the particular hand(Zhu et al., 2013).

Many researches applied histogram where the orientation histogram is used as a feature vector (Freeman and Roth, 1995).The method for recognizing gestures are based on pattern recognition using orientation histogram. The system consists of two phases, the training phase and running phase. 'In training phase, for different input gestures the training set is stored with their histograms. In running phase an input image is presented to the computer and the feature vector for the new image is formed. Then comparison performed between the feature vector of the input image with the feature vector (oriented histogram) of all images of the training phase, using Euclidean distance metric' (Ibraheem and Khan, 2012).

Clustering algorithms is a common term comprises all methods that partitioning the given set of sample data into subsets or clusters (Bezdek et al., 1984). It is based on some measures between grouped elements (Li, 2003). Clustering algorithms have been widely spread because of their ability of grouping complicated data collections into regularly clusters (Bezdek et al., 1984).

Gestures are divided into two sections, static gesture and dynamic gesture. The first one represents hand shape or hand posture requires less computational complexities whereas dynamic gesture are complex in nature and describe the movements of hands. Gesture are useful to interpret the movement of fingers, hands, arms as well as other parts of the body for convey information in a meaningful way for environmental interaction (Murthy and Jadon, 2009).

To interpret gesture recognition in HCI system there are two approaches that commonly used one is Data Gloves that uses optical sensors (Murthy and Jadon, 2009). This system is very sensitive for illumination, the position of the hands and orientation as well of the hands. The vision base approach uses cameras for capturing an input image. Vision based approaches allows easy, natural and less cost comparing to glove based approaches as it require the users to put/attach special instrumental devices and be connected to the computer which persist the natural interfaces between the users and computes (Pavlovic et al., 1997).

Other approaches such as appearance based where the input image is modeled by using the feature extracted from an image which will later be compared with the feature extracted from stored image. The appearance bases approached are much simpler and easier to handle in compare with the 3D based model (Pavlovic et al., 1997; Garg et al., 2009). The method used in this approaches is use of skin colored region of that image however changing the lighting condition and background change the effect of this approaches.

3D based approach are using for modeling and analyzing hand shape (Kainz et al., 2014). The kinematic parameters are required for establishing 2D projection from a 3D based model of that hand to correspond the edges of that image hand (Khan and Ibraheem, 2012). This model is classified in volumetric and skeleton models.Volumetric model use 3D visual appearance of hand and in real time application it uses all the hand parameters that has enormous dimentionalities. Skeleton models are used to limits that problems of that set of parameters of hands model in 3D that are used for volumetric models (Pavlovic et al., 1997).

AdaBoost learning algorithm is used that scale invariant feature transforming a histogram which represent 'a gradient orientation and magnitude information within a small image patch' (Wang and Wang, 2008; Garg et al., 2009). Using the sharing feature concept, efficiency can be achieved nearly 98 % (Wang and Wang, 2008). AdaBoost learning algorithm that can adaptively select the best features in each step and combine them into a strong classifier can be used. The training algorithm based on AdaBoost learning algorithm takes a set of **positive** samples, which contain the object of interest and a set of **negative**

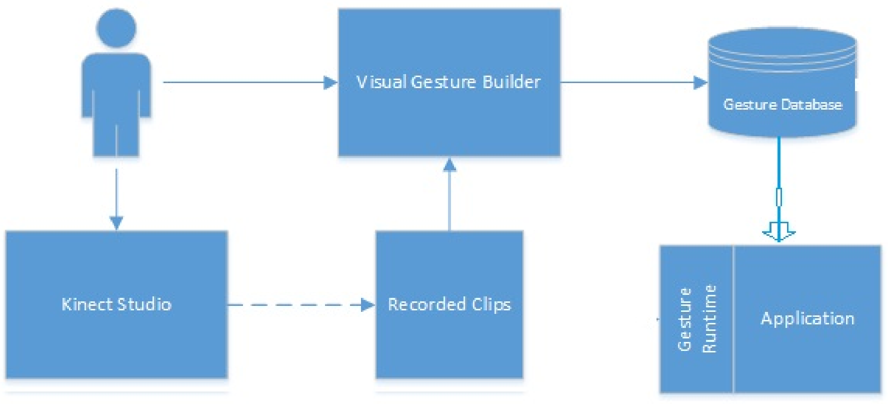


Figure 3: Gesture recognition process diagram by VGB
source: https://msdn.microsoft.com/en-us/library/dn785529.aspx

samples that is images that do not contain objects of interest. During the training process, distinctive Haar-like features are selected to classify the images containing the object of interest at each stage (Wang and Wang, 2008).

4 Research methods

4.1 Collecting data

This project uses Microsoft Kinect Studio for recording IR 11 Bit data. The newer version SDK allows to capture recorded data in a processed event file format (XEF) allows to record processed IR data as well as depth and skeleton data. I f data is recorded in XRF format then it could be converted by using the KSConvert in a command line prompt available in the Kinect files folders. To get a satisfactory result out of the recording we will capture recording by allowing each members of the target audience individually when they are interacting with the prototyping of our application. The different tilt angels of the sensor and having various outfits specially upper parts of wearing cloths will be considered during our recording settings.

4.2 Recording data

In our project we recorded all the files in a raw format collected by Kinect studio. The benefit is that any future change of depth data or skeleton data we are able to build the solution and add those sequence to the database (VGBOverview, 2014). The newly created converted file will generate both IR, depth, opaque and skeleton data to work with the gesture database. In our recording we simply not relying the depth and skeleton data as this can't disclose the whole intention of any user so by observing the IR data it is easier to determine the user intention. As this project dealing with data driven solution so it is important to tag frames and by looking the IR data it is easier to interpret recording data

4.3 Sample audience

We include a variety of users for generalization the research criteria and for selecting the training session. We choose to have 5 participants where two are minor aged and rest of the 3 are adults have knowledge about creating and building Lego bricks. Moreover it is important to have different body construction for building the training set as all the users definitively perform a gesture distinctively as due to their weight and height.

We will record each video clip separately or individually with the test users because performing a gesture by one user might influence other users to do the same way. A wide variety of relevant or specific gestures capturing by different users are important in real world application.

4.4 Kinect placement

It is important to make recording with users in different environments with different heights and tilt angels of Kinect sensors. The reasons for doing so are that when we tracking the skeleton it produces different skeleton data at varied tilt angels and heights because the body joints are logged differently by the Kinect sensor. For instance one joint may be occluded from one angel but it can be observed from another angel. The table or any furniture can also contribute occlusions for tracking skeleton. Moving kinect sensor a bit from its previous location might improve the recording sequence for proper skeleton tracking.

The prebuild algorithm in VGB works well on tilt-corrected skeleton data which runs locally for the users environment. However it is also important to re create the user environment as well. The best placement for Kinect sensor for recording is the centered above or below the screen. In our case we had only possibilities to capture the video clips below the screen. We record the users from different distance from the Kinect sensor. As the gesture we will capture will be performed in a seated position we do introduce variety of seating options and postures. As our application relies on multi users perspective so we not only relies on centered position of the sensor rather we do positioned the sensor in both left and right angels as well. In the test scenario we had Kinect placed about 0.5 to 0.7 meter from the Lego bricks objects.

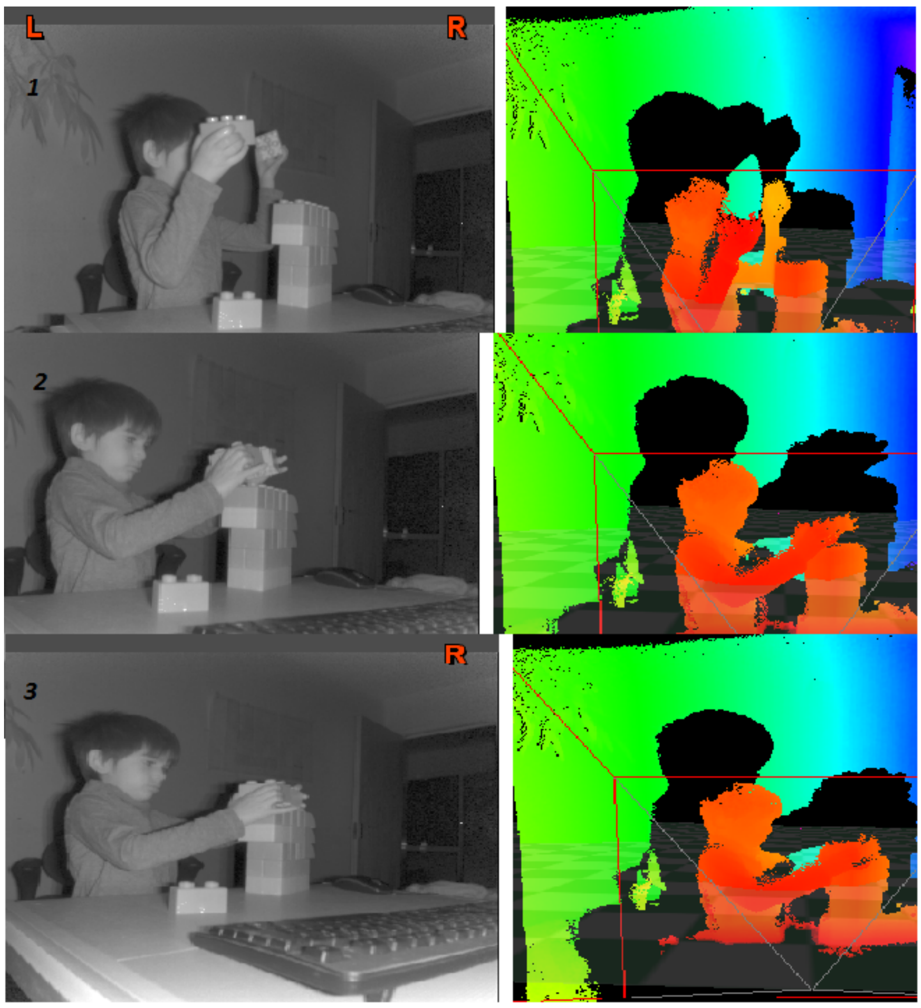


Figure 4: Tagging the gestures frame by frame

4.5 Tagging the recorded data

Tagging the data plays a vital role for the result. it is the most time consuming section for detecting gestures. The most important step is that sorting the garbage in and garbage out as VGB uses the data driven method. Our approach for tagging the gestures are that we break the gestures into several sub gestures for building multiple detectors. Then we tagged all the frames that create the gesture however we took only the core portions of that gesture. The action we captured for tagging is that the canonical motions of that gesture. We avoid tagging the actions such as when the user prepare to pick a brick or the recovery actions such as failed to set up the brick and or leave the hands for taking another brick etc.

Even though VGB can produce one side of for instance left hand data to use another side of hand as it uses mirroring that data for training purpose but we use both hands settings for the value of body side.

5 Implementation

5.1 Visual Gesture Builder

A data driven powerful solution tool Developed by Microsoft corporation which allows to detect or recognize gestures via machine learning. The basic idea for detection a gesture is to create content based tasks rather writing bunch of codes. It creates a small gesture database that has significantly very low run time cost efficiency in terms of memory overhead and CPU usage (MSTech, 2014). The whole process is very simple as it uses the data driven technology which refers that fist we record interesting people for instance prototyping an application. Kinect studio can be used for accomplishing this task. Next, we can convert this data to an XEF processed format for IR /depth and skeleton data. Then using VGB we can tagging /flagging or leveling the meaningful data that we are interesting for the particular frames which will define a gesture from the recording clips. The whole process from recording to database event handling driving process shows in figure 3.

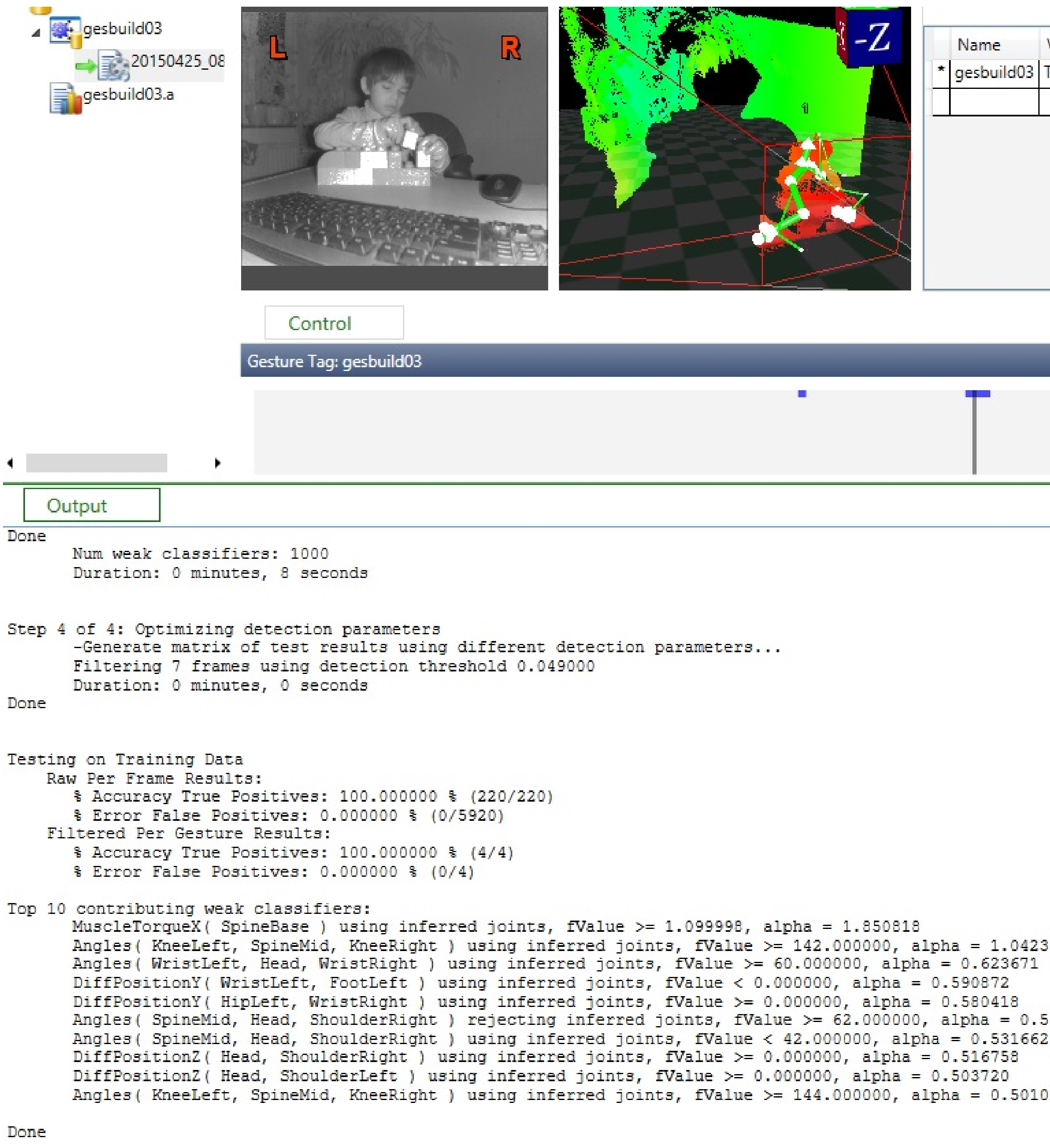
After the completion of tagging we can build the gesture detector. Visual gesture builder uses machine learning for building the database at the run time for the project solution. A handy tool, Visual gesture live view enable us to iterate all the previewed gestures for rapid prototyping.

5.2 Building gestures

when we are done with tagging the data we are able to build a solution or creating a gesture project. During the building the machine learning algorithm process the tagging data. This process create a gesture database file with *.gdb extension which later be attached to any application during the run time. Figure 5 shows how to build gestures. In this time we captured the recording from the right side of the monitor and we tagged 7 frames to train the data for gesture recognition.

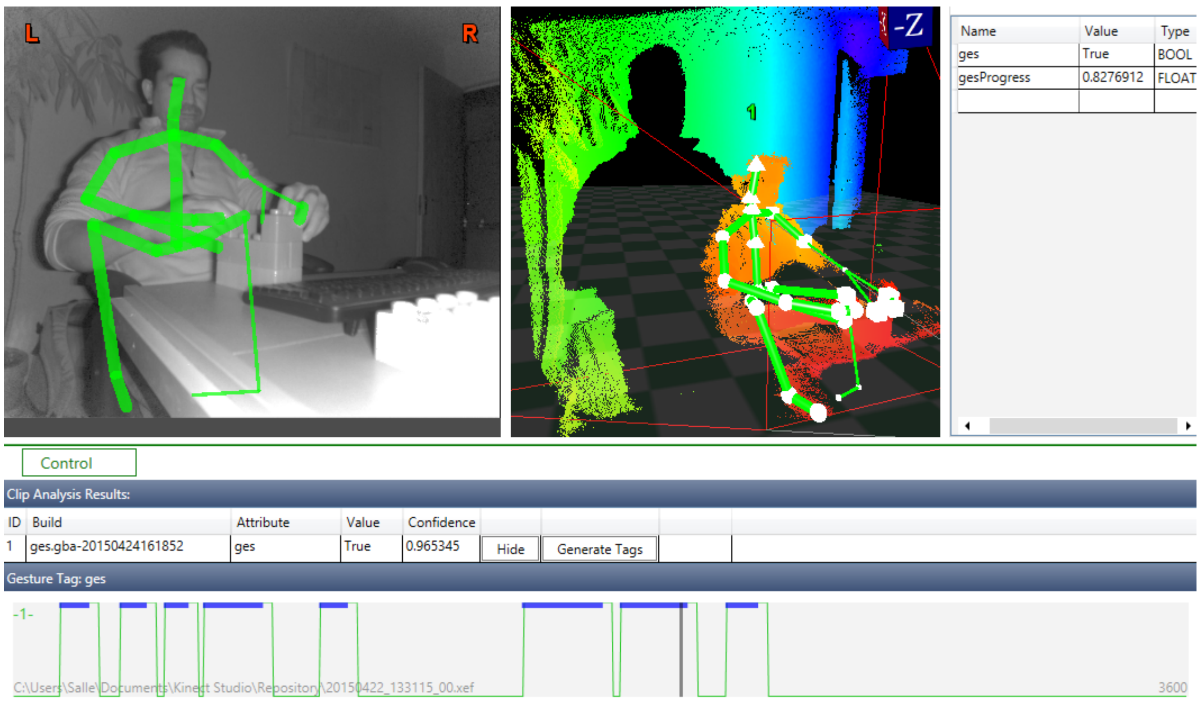
The log file also provide very important information. For instance, the log file in figure 5 we can see that the angle velocity between the WristLeft, Head, WristRight is actually a good indicator for a right gesture. If we look at the top classifier, we can interpret this as:

if(Angles(WristLeft, Head, WristRight) >= 0.6 f)



```
Done
       Num weak classifiers: 1000
       Duration: 0 minutes, 8 seconds


Step 4 of 4: Optimizing detection parameters
       -Generate matrix of test results using different detection parameters...
       Filtering 7 frames using detection threshold 0.049000
       Duration: 0 minutes, 0 seconds
Done


Testing on Training Data
    Raw Per Frame Results:
       % Accuracy True Positives: 100.000000 % (220/220)
       % Error False Positives: 0.000000 % (0/5920)
    Filtered Per Gesture Results:
       % Accuracy True Positives: 100.000000 % (4/4)
       % Error False Positives: 0.000000 % (0/4)

Top 10 contributing weak classifiers:
       MuscleTorqueX( SpineBase ) using inferred joints, fValue >= 1.099998, alpha = 1.850818
       Angles( KneeLeft, SpineMid, KneeRight ) using inferred joints, fValue >= 142.000000, alpha = 1.0423
       Angles( WristLeft, Head, WristRight ) using inferred joints, fValue >= 60.000000, alpha = 0.623671
       DiffPositionY( WristLeft, FootLeft ) using inferred joints, fValue < 0.000000, alpha = 0.590872
       DiffPositionY( HipLeft, WristRight ) using inferred joints, fValue >= 0.000000, alpha = 0.580418
       Angles( SpineMid, Head, ShoulderRight ) rejecting inferred joints, fValue >= 62.000000, alpha = 0.5
       Angles( SpineMid, Head, ShoulderRight ) using inferred joints, fValue < 42.000000, alpha = 0.531662
       DiffPositionZ( Head, ShoulderRight ) using inferred joints, fValue >= 0.000000, alpha = 0.516758
       DiffPositionZ( Head, ShoulderLeft ) using inferred joints, fValue >= 0.000000, alpha = 0.503720
       Angles( KneeLeft, SpineMid, KneeRight ) using inferred joints, fValue >= 144.000000, alpha = 0.5010

Done
```

Figure 5: A portion of the log file from an AdaBoostTrigger project that indicates the top 10 weak classifiers defining a gesture



Name	Value	Type
ges	True	BOOL
gesProgress	0.8276912	FLOAT

Clip Analysis Results:

ID	Build	Attribute	Value	Confidence		
1	ges.gba-20150424161852	ges	True	0.965345	Hide	Generate Tags

Gesture Tag: ges

C:\Users\Salle\Documents\Kinect Studio\Repository\20150422_133115_00.xef

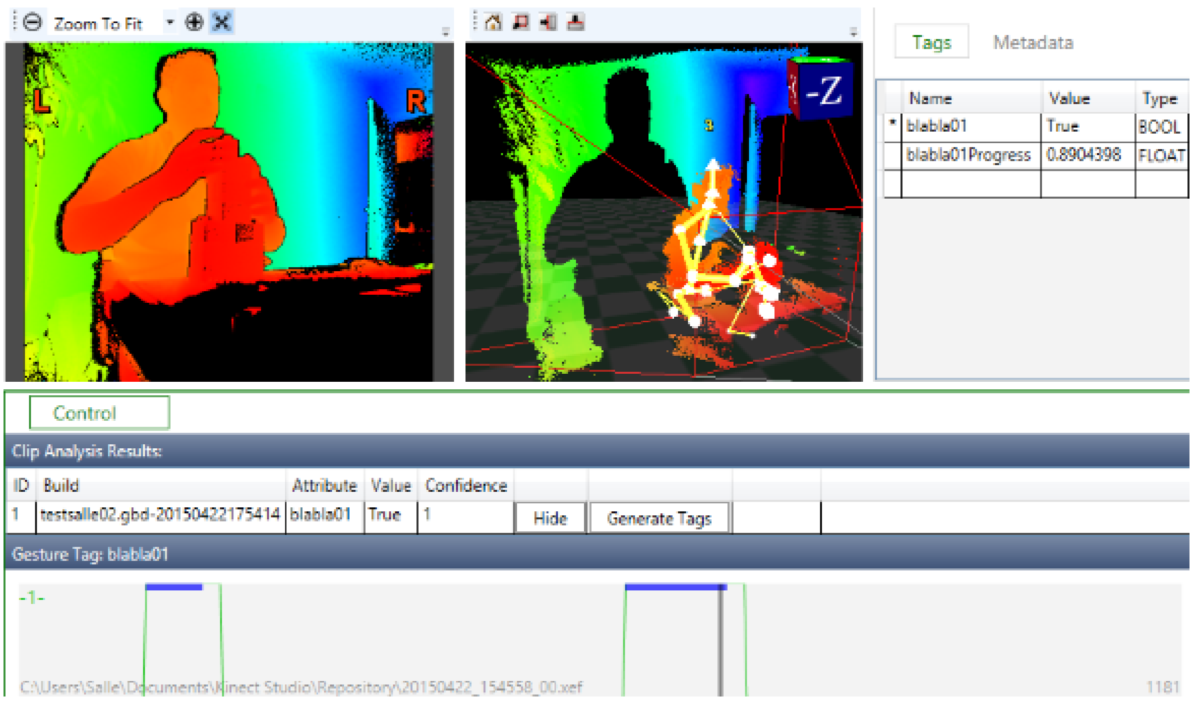
Figure 6: Analyzing of gestures

LeftHandBrickDetected = TRUE;

The AdaBoostTrigger we are able to extract knowledge from the log files to implement our own gesture detection.

5.3 Analyzing gestures

The analyzing tool allows us to test the detection of any gesture rapidly in the project file. The main advantage of utilizing analysis project is that we are able to compare results with many builds such as rejecting any tags or accepting tags. We can also determine whether we need to add more clips for training the data set for improving better detection or not. During the training phase if we can analyze and tagging the data correctly then the result will be great as in figure 6 we can see that several frames are tagged and accepted as shown by the blue color. The gray vertical line shows that the confidence level that we can achieve is near about 100 percent. The frames at the end of this gray line and near to the green line we have rejected that by training the data.



Name	Value	Type
blabla01	True	BOOL
blabla01Progress	0.8904398	FLOAT

Clip Analysis Results:

ID	Build	Attribute	Value	Confidence
1	testsalle02.gbd-20150422175414	blabla01	True	1

Gesture Tag: blabla01

C:\Users\Salle\Documents\Kinect Studio\Repository\20150422_154558_00.xef 1181

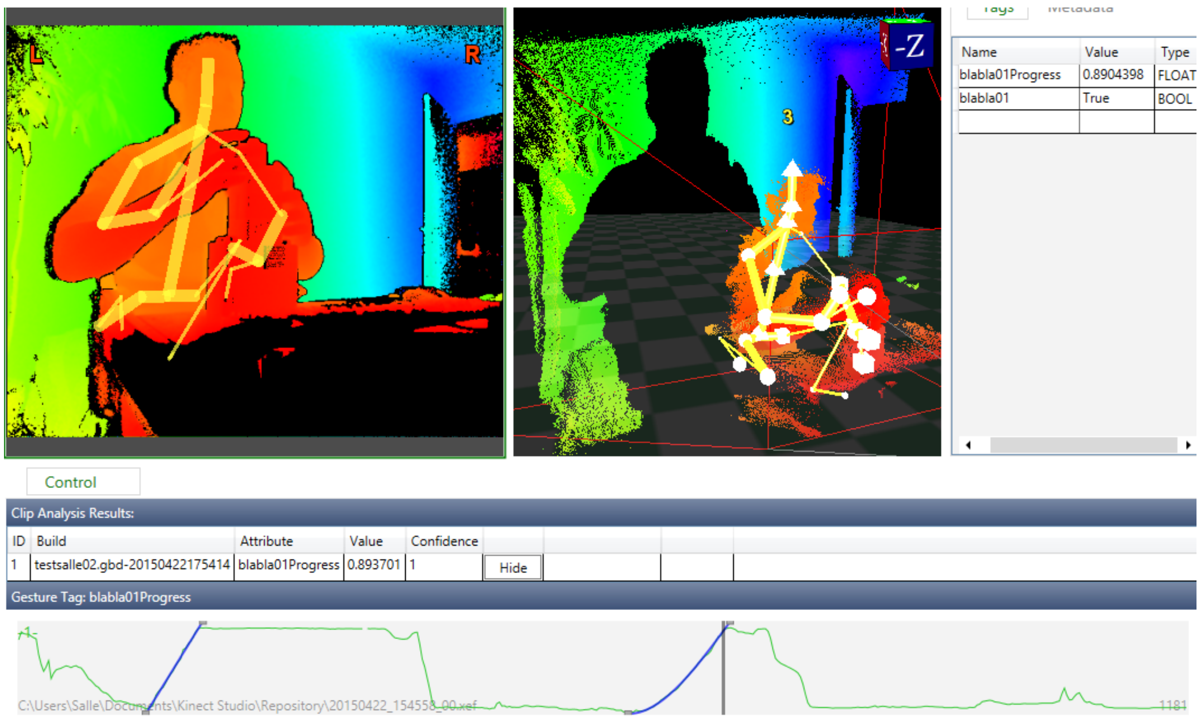
Figure 7: Analyzing of gestures after tagging the data

6 Result

To test and evaluate the result we made a number of recording clips where we trained the data set by tagging the frames to detect the gestures. We then send those training set to analyze section where we generate tags either by accepting or rejecting those frames visually. In terms of static or discrete gesture for instance moving arms or not to place a brick we gave a value either true or false. Then in the continuous gesture that is gesture from start to ending movement we used progressive algorithm and trained the data set by providing the floating point values in range between 0 to 1. As our aim is to detect gesture as correct as possible so we accept tagging those frames near a value to 1. In figure 7 we can see that the confidence level is 1 and the training progress is around 90 percent accuracy.

In figure 8 we can see that machine learning algorithm the progressive trained data reached a value about 89 percent and confidence level is 100 percent. To rapidly prototyping and evaluating the test result we use the gesture builder live preview to test our results. From figure 9 we see that the detecting gesture after we have trained.

In figure 10 we can see a static gesture detection but no progressive gesture as this is a continuous gesture even though the solution failed to detect but the



Name	Value	Type
blabla01Progress	0.8904398	FLOAT
blabla01	True	BOOL

Control

Clip Analysis Results:

ID	Build	Attribute	Value	Confidence	
1	testsalle02.gbd-20150422175414	blabla01Progress	0.893701	1	Hide

Gesture Tag: blabla01Progress

1
C:\Users\Salle\Documents\Kinect Studio\Repository\20150422_154558_00.xef
1181

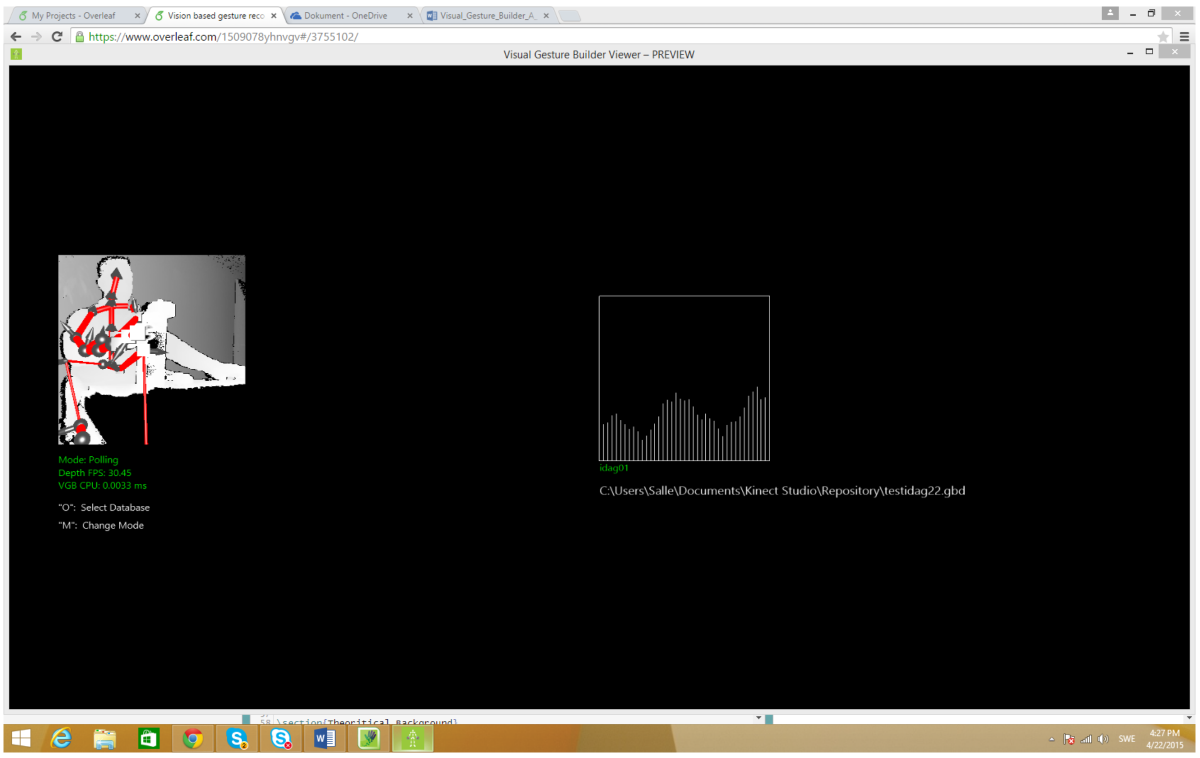
Figure 8: Analyzing of gesture progress

waves are still being viewed. But in figure 11 we can see a continuous gesture that the machine learning algorithm has detected during our training session.

7 Discussion

In this paper we aimed to explore the possibilities of how can data driven solutions can be used to detect gestures. We have added many clips recorded by Kinect studio by many users include the researcher. Our aim was to detect gestures when performing some actions using Lego bricks. When construct some sort of figures we have observed that different time schedule are required for different users. The time placing one object to the rest of the figures can be filtered out by efficient tagging. In our research we explored both static and discrete gestures to detect by the tool. We provide the values true or false for detecting static gestures such as moving one or both hands to place a bricks for constructing something. We avoid both the starting point for a gesture to reduce the latency and also ending the tagging as soon as a user has placed an object for the final pitch.

For the continuous gesture we placed the trained data and given a value in between 0-1 where we accept a value that reach in between 89-100 percent


My Projects - Overleaf
Vision based gesture reco
Dokument - OneDrive
Visual_Gesture_Builder_A
https://www.overleaf.com/1509078yhnvgv#/3755102/
Visual Gesture Builder Viewer – PREVIEW
idag01

```
Mode: Polling
Depth FPS: 30.45
VGB CPU: 0.0033 ms

"O":  Select Database
"M":  Change Mode
```

```
C:\Users\Salle\Documents\Kinect Studio\Repository\testidag22.gbd
```

Figure 9: Detecting a discrete gesture posture

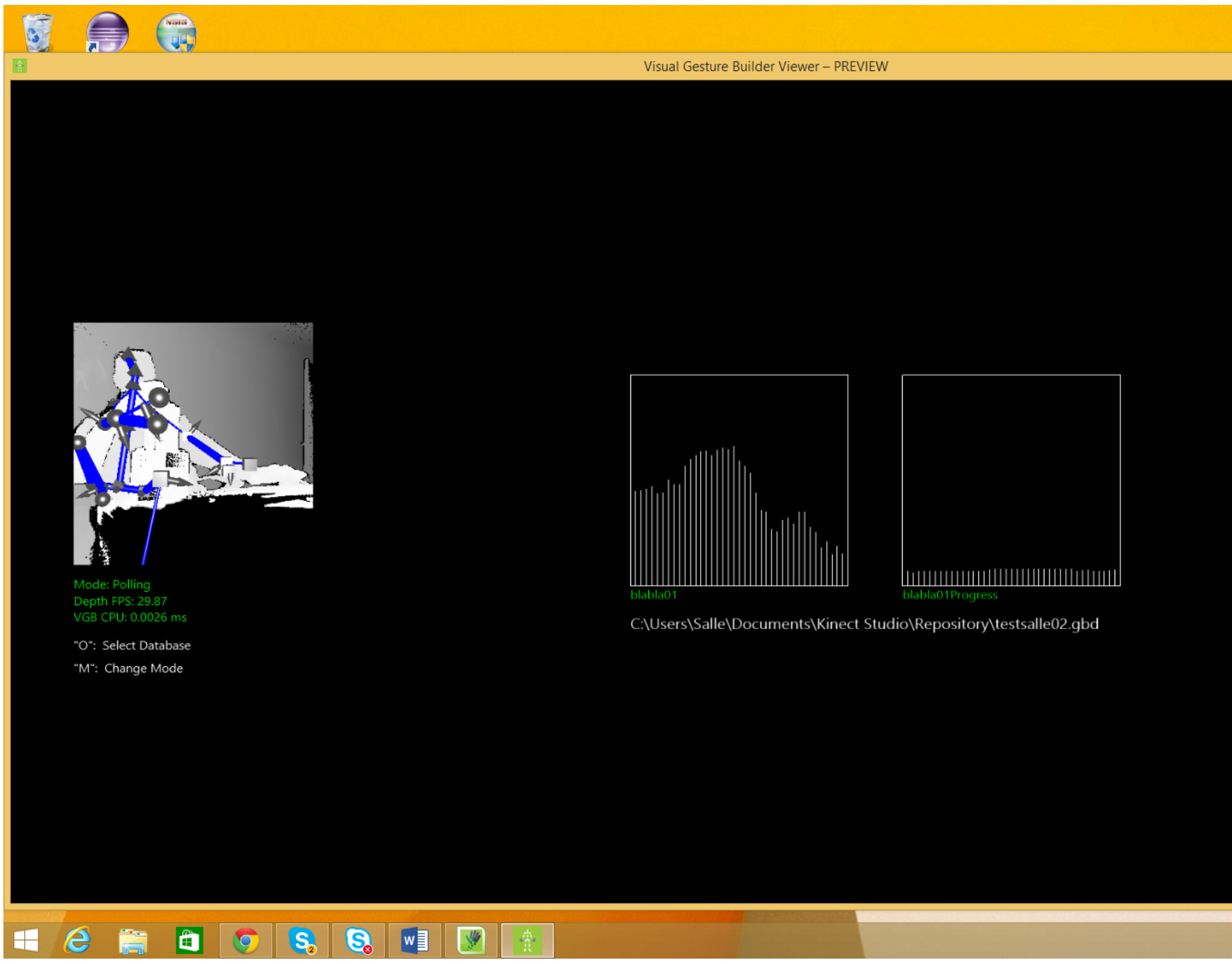


Figure 10: Detecting a discrete gesture with no progress

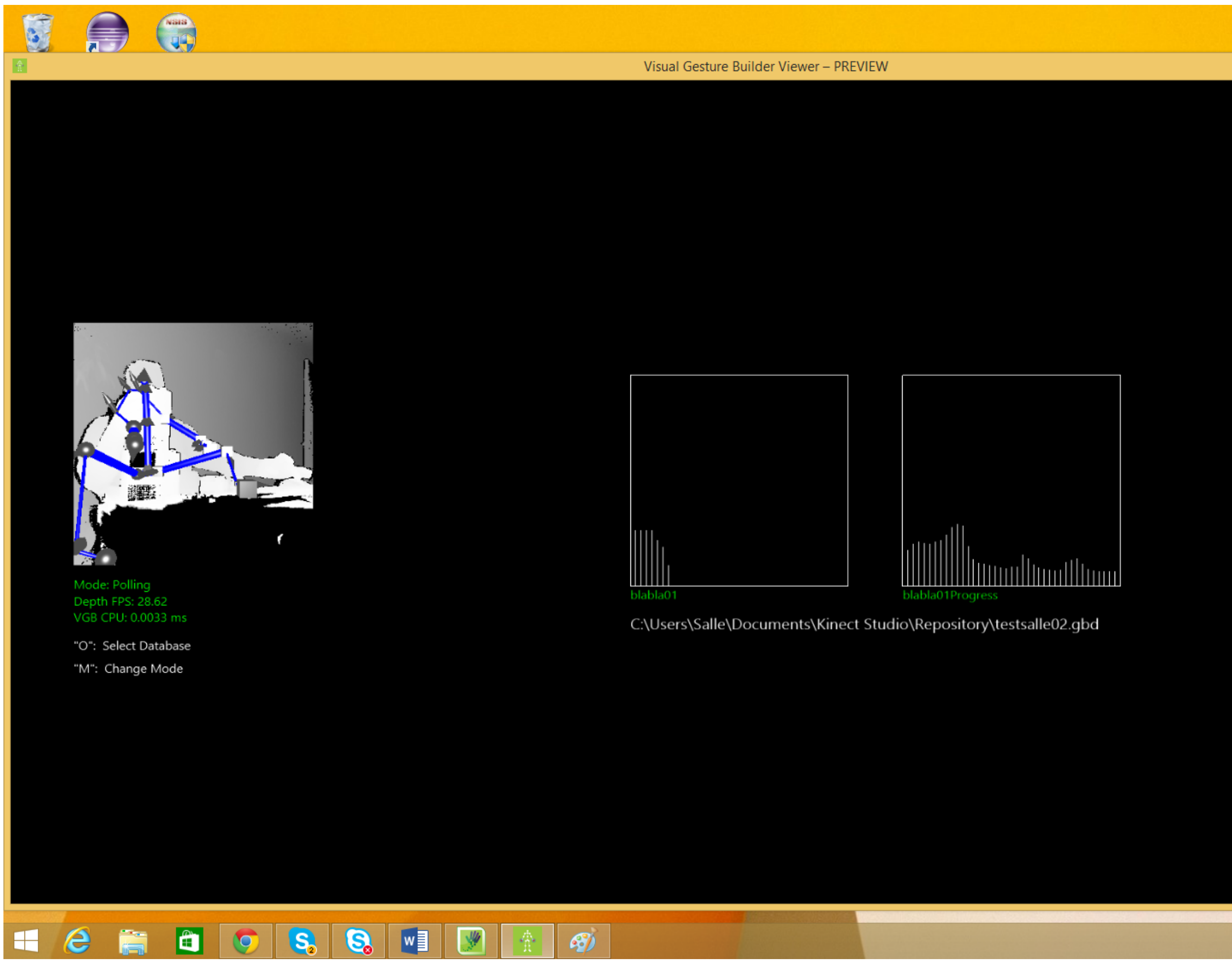


Figure 11: Detecting a discrete gesture and with continuous gesture progress

accuracy. We reject the frames that has the most false positive as we try to trained the data set that has most false negative values. Gesture detection handled by analyzing the content which minimize the engineering efforts. It is possible to quickly building, testing and prototyping an gesture application. Even though we need to add several clips for training a data set the size of database remain independent. The API and future up coming tracking system allow to migrate the clips for measuring the depth and skeleton data to detect any gesture for implementing with any project.

The limitations are the time consuming for tagging the data. Kinect applications specially integrating Kinect sensors requires very powerful PC that includes all the necessary components such as CPU, memory, powerful graphic cards as well as lot of storage as a few minute record requires Gigabit of space. The new Kinect SDk version 2.0 need USB 3.0 that collide with other USB ports and can assuage the speed of Internet access communicate by external USB power. sometime at the run time Windows fails to response cause starting the PC.s once again.Efficient tagging also requires to learn keyboard shortcut command using.

8 Future work

As we choose to construct the Lego with random bricks the future work can be as to mark those bricks in where every pieces should be the application can be able to manipulate objects due to construct time and efficiency. We can also hire some expertise to test the time constraints compare to amateur users. Moreover using both hands synchronizing process can be determined by selecting, adjusting or distinguishing the time in between the users.

9 Conclusion

Traditional method to detect gestures by Kinect sensors is not a trivial task. There is no robust method to capture gesture but writing a lot of codes and adjusting the codes in accordance with the light, heights and occlusion of position of Kinect sensors. Visual gesture builder provides the developer building Kinect application in a productive way. Though gesture recognition is in its infant phase but it can be used in many settings in the near future. The hardware and processing cost make significant step to use gesture recognition in widespread area and utilize it practically in many fields. The results are better quality gesture detection with reduced the latency time. As tagging is the important part of creating, building that produce good gestures so it is recommended to include people that can assure the data that has to be approved for adding and training for a certain project settings.

References

Bezdek, J. C., Ehrlich, R. and Full, W. (1984), 'Fcm: The fuzzy c-means clustering algorithm', *Computers & Geosciences* **10**(2), 191–203.

Freeman, W. T. and Roth, M. (1995), Orientation histograms for hand gesture recognition, *in* 'International workshop on automatic face and gesture recognition', Vol. 12, pp. 296–301.

Gandhi, V. S., Khond, A. A., Raut, S. N., Thakur, V. A. and Shaikh, S. S. (2014), 'A review of various gesture recognition techniques', **3**, 8202–8206.

Garg, P., Aggarwal, N. and Sofat, S. (2009), 'Vision based hand gesture recognition', *World Academy of Science, Engineering and Technology* **49**(1), 972–977.

Gu, L. and Bone, D. (1999), Skin colour region detection in mpeg video sequences, *in* 'Image Analysis and Processing, 1999. Proceedings. International Conference on', IEEE, pp. 898–903.

Ibraheem, N. A. and Khan, R. (2012), 'Survey on various gesture recognition technologies and techniques', *International journal of computer applications* **50**(7), 38–44.

Jana, A. (2012), *Kinect for Windows SDK Programming Guide*, Packt Publishing Ltd.

Kainz, O., Jakab, F. et al. (2014), 'Approach to hand tracking and gesture recognition based on depth-sensing cameras and emg monitoring', *Acta Informatica Pragensia* **3**(1), 104–112.

Khan, R. Z. and Ibraheem, N. A. (2012), 'Hand gesture recognition: Aliterature'.

Li, X. (2003), 'Gesture recognition based on fuzzy c-means clustering algorithm', *Department Of Computer Science The University Of Tennessee Knoxville* .

Mazumdar, D., Talukdar, A. K. and Sarma, K. K. (2013), Gloved and free hand tracking based hand gesture recognition, *in* 'Emerging Trends and Applications in Computer Science (ICETACS), 2013 1st International Conference on', IEEE, pp. 197–202.

MSTech, M. (2014), 'Visual gesture builder: A data-driven solution to gesture detection', `http://aka.ms/k4wv2vgb`. [Online; accessed January 10, 2015].

Murthy, G. and Jadon, R. (2009), ‘A review of vision based hand gestures recognition’, *International Journal of Information Technology and Knowledge Management* **2**(2), 405–410.

Pavlovic, V. I., Sharma, R. and Huang, T. S. (1997), ‘Visual interpretation of hand gestures for human-computer interaction: A review’, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **19**(7), 677–695.

Phung, S. L., Bouzerdoum, A. and Chai Sr, D. (2005), ‘Skin segmentation using color pixel classification: analysis and comparison’, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **27**(1), 148–154.

Sánchez-Nielsen, E., Antón-Canalís, L. and Hernández-Tejera, M. (2004), ‘Hand gesture recognition for human-machine interaction’.

Stergiopoulou, E. and Papamarkos, N. (2009), ‘Hand gesture recognition using a neural network shape fitting technique’, *Engineering Applications of Artificial Intelligence* **22**(8), 1141–1158.

VGBOverview, M. (2014), ‘Visual gesture builder: Overview’, `https://msdn.microsoft.com/en-us/library/dn785529.aspx`. [Online; accessed March 10, 2015].

Wang, C.-C. and Wang, K.-C. (2008), Hand posture recognition using adaboost with sift for human robot interaction, *in* ‘Recent progress in robotics: viable robotic service to human’, Springer, pp. 317–329.

Worsley, M. and Blikstein, P. (2013), Towards the development of multimodal action based assessment, *in* ‘Proceedings of the third international conference on learning analytics and knowledge’, ACM, pp. 94–101.

Zhu, Y., Yang, Z. and Yuan, B. (2013), Vision based hand gesture recognition, *in* ‘Service Sciences (ICSS), 2013 International Conference on’, IEEE, pp. 260–265.