Data Flow And System Architecture

1-Kinect output

Kinect sends colored image, depth image and Joints positions . each package of them called frame with about 30 frames per second.Joints of the user represented as 3D points. sign language depends mainly on the hands so we only interested on the hands, arms and shoulders.

2-Hand segmentation

the process of hand segmentation has two main steps

- Extract Hand Properties:

  this step aims to extract the hand Dimensions and the length and width for each finger, using the hand position data from the Kinect and by asking the user to raise his hand and space between his fingers as shown in figure 1, the program makes two loops from the hand point one scans the depth channel vertically and other horizontally to find the first big change on the depth values which means the end of the hand palm and the start of the fingers part as shown in figure 2.

Figure 1. Shows Person Raising his hand

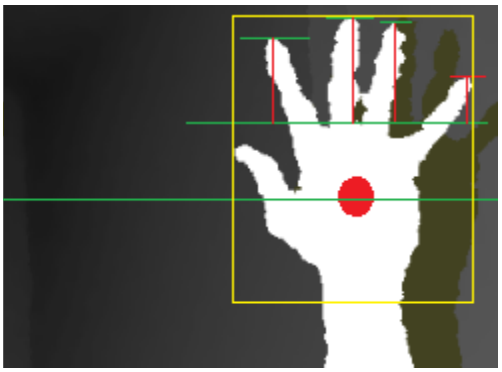


Figure 2. shows the results

  this step occur only once and its results used along the tracking time.

- Hand Extraction And Segmentation

  in this step and depends on the values calculated on the previous step this application cuts the rectangular part around the hand as seen in Figure 2 in yellow, every frame from the depth channel and color channel and sends them as a package to the finger detection unit.

3- Finger Detection

this unit extracts the fingertips and combine the results with shape matching to determine the fingers states .

to extract the fingertips the program finds the contours around the hand as shown in figures 3 and 4

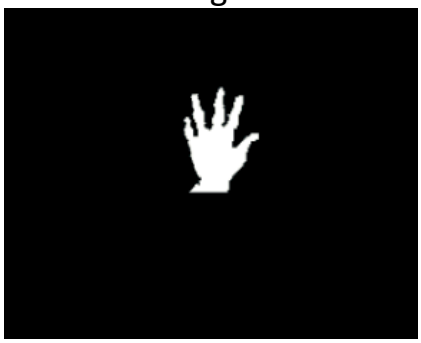


Figure 3. the hand after Segmentation

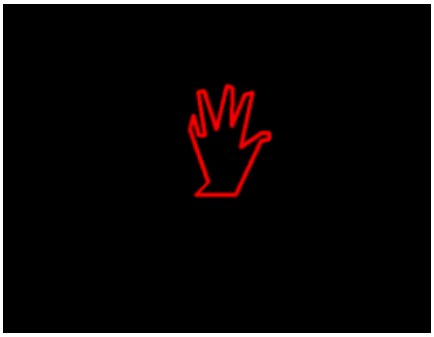
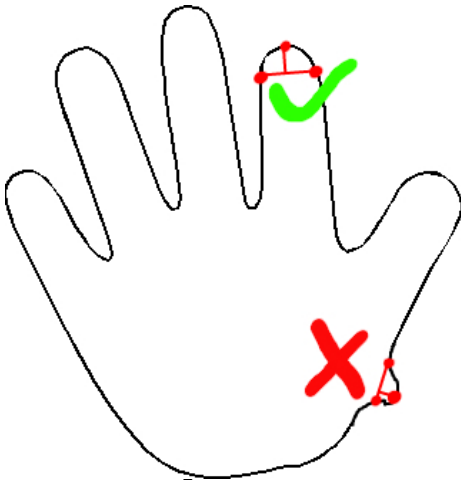


Figure 4. the contour around the hand

using Suzuki algorithm, the next step to scan the contour to find the points that have the highest curvature values using cauchy definition for curvature , these points called the candidate, from each candidate point in the contour we take two points one from left and the other from right and make virtual line between them , finally make projection for that candidate point on that line if the the projection point was around the center of the line and the projection line has length not smaller than constant number , this will be a fingertip point. figure 5. Describes the algorithm and shows two candidate points one represents a finger tip and the other is not note how simply it is to distinguish between them.



now these data will be added to the package of data and sent to the Shape Matching unit.

4- Shape Matching Unit

this unit almost uses all the data comes from the package to match between database of templates and the current frames.

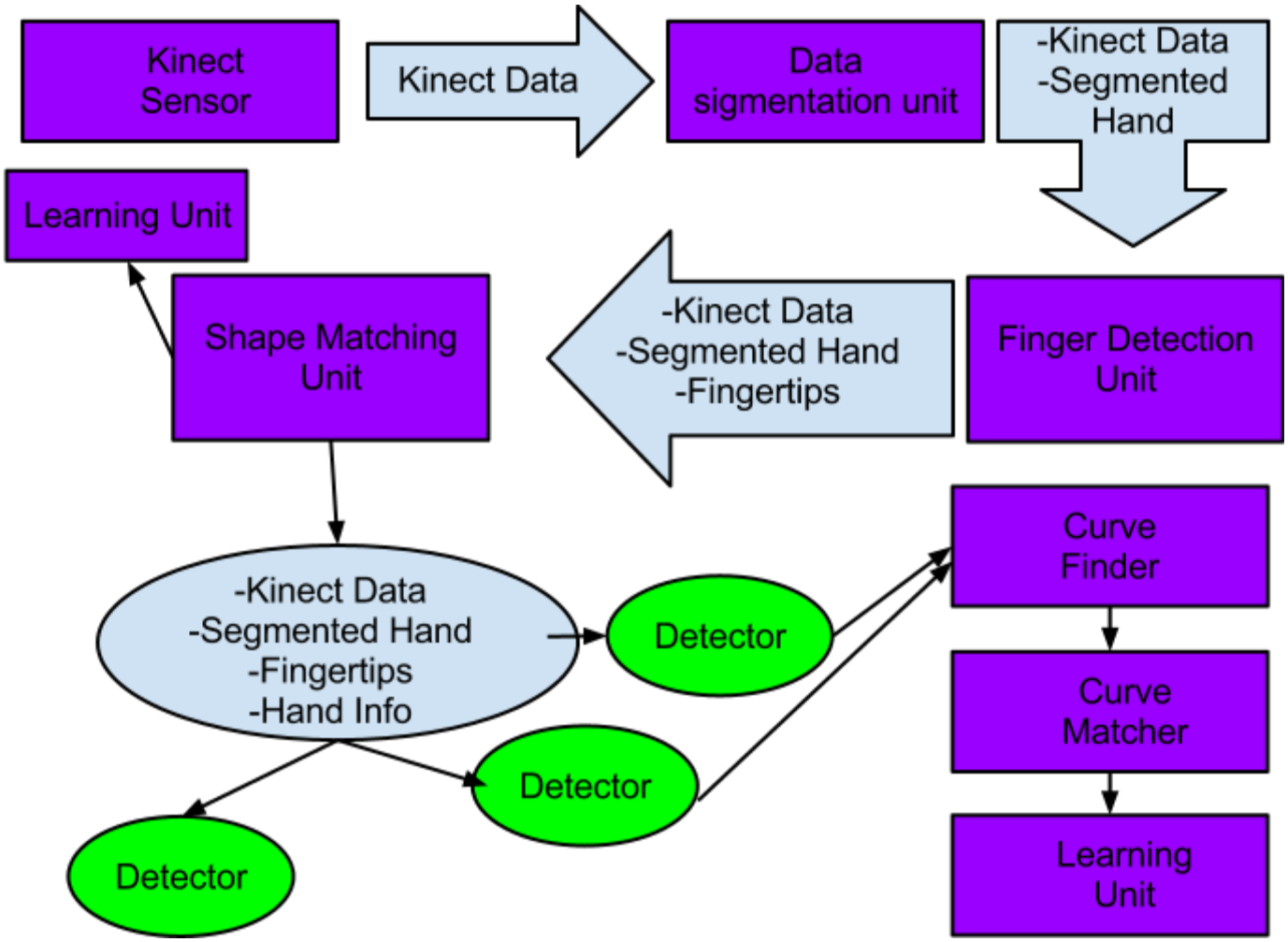
the program uses the (Dynamic wrapping algorthim) not directly to compare the shapes and to find the orientation of the hand.
the orientation of the hand with a predefined values in each template used to map the fingertips to each finger. this part uses a lot of CPU power so by using point tracking algorithm we can track that fingertip and not use the mapping part all the time, it will be used only when we lost the tracking for that fingertip.

this unit acts as the last data generator unit , and this package of data will be sent to all detectors in the system to extract words, from signs and movements.

5- Curve Finder and Matching Unit.

this unit used by the detectors. it keeps tracking the user to build curves from his movement and do two things.

- Curve Finding:
  the main problem in curve matching is to extract the curve of the word from user , i mean we need to find the start point where the curve begins and the point where the curve ends.
  this unit doing this using something similar to the Hidden markov states , by transferring the curve first to the polar coordinates and ignore the distance between points , then takes this angle as a condition to enter a state , the first state represents the first point on the curve , after reaching the last state , the program has partial curve called candidate curve and maybe represents predefined movement on the database , the curve generated till this point forwarded to the next unit described in the next step here.
  Find more details on part A
- Curve Matching:
  Curve Matching unit using the same DTW algorithm described before but the aim of using it again after the curve finder unit is to make sure that the result is correct because the error rate in curve finder increases with increasing the curve length, the other purpose is to make the system able to learn from the user , by finding the similarity between database curves and the user-generated curves.

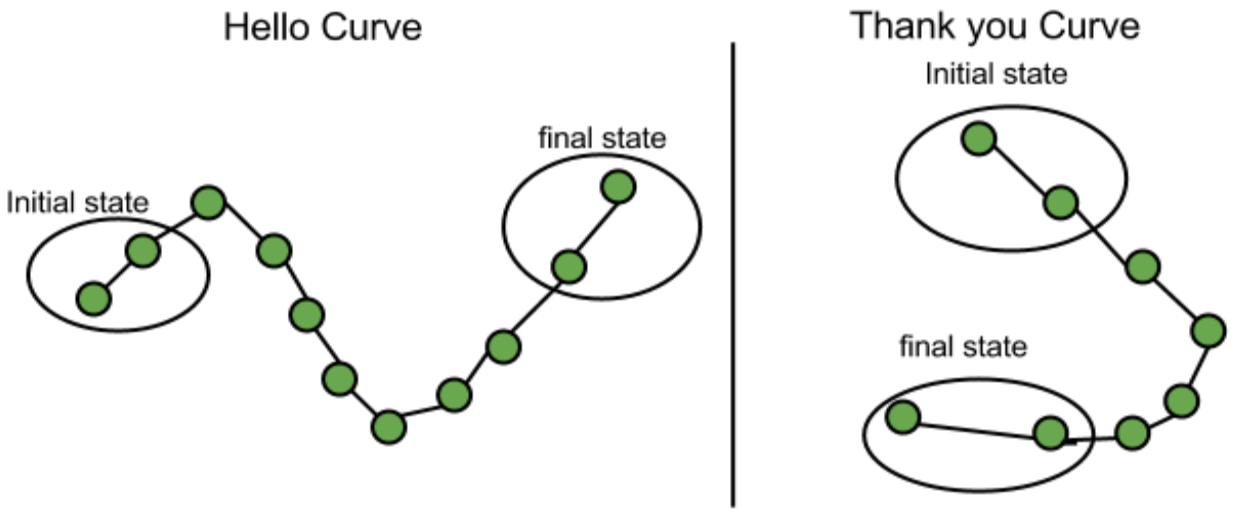


this sequence diagram shows the data flow in the system.

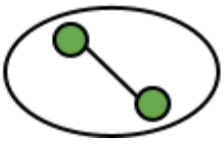
A- Curve Extraction Unit in details

the scenario works as the following , at each frame each detector receives a package contains the shape matching results , fingertips , skeletons and and general hand and user data.

what happens that each detector is responsible to build a curve from these skeleton points comes at each frame to extract words from movements. this done easily by making array to store this data with time to make a curve,but the problem in how to know when special movement starts and when it's end. it's impossible to scan all user movements for all the time from the beginning to detect which movement it is , the traditional solutions is to limit the movement with an interval of time and try to find a curve matching on these fixed intervals, this method wastes a lot of movements and gives a bad accuracy. what our system doing is keep tracking the curve generated by the user and predicts the start of the curve and starts to find when it will end , take this scenario as example

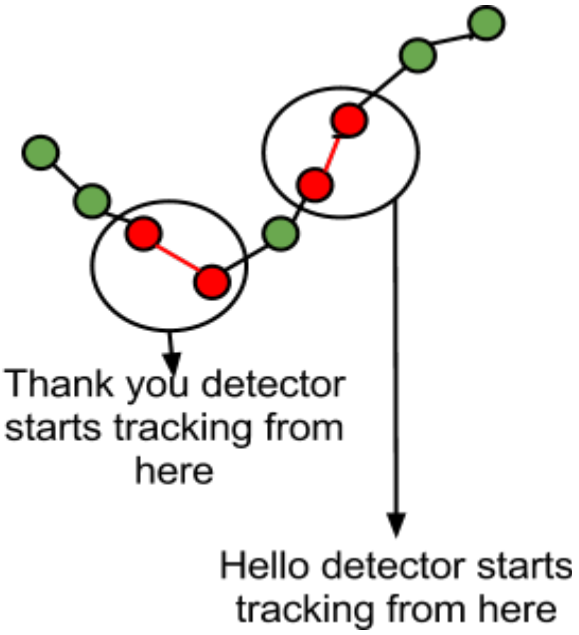


Hello word Initial state, this gives the Hello detector that maybe there will be this word so it will start build the curve to find the final state or go to a state not on the database and stop the tracking.



Thank you word Initial state, this gives the thank you detector that maybe there will be this word so it will start build the curve to find the final state or go to a state not on the database and stop the tracking.

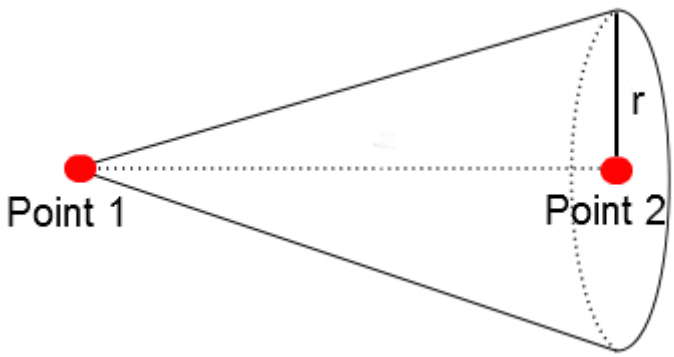
and this is example shows both detectors when start to track curves.



this example shows how the detectors will start tracking the user when the first state found , this will lead the detector to the next state if it is satisfied it will move to the next until the final state. or it will destroy this series of states and go back waiting for initial state again. each state will carry the points that satisfied its condition and create the next state. and there is a for-loop to feed all these states independently of each other , which means that you can see two detectors or more tracking the same word as example.
after reaching final state it will go back to all previous states and take the points as a curve and send it to the curve matching unit.

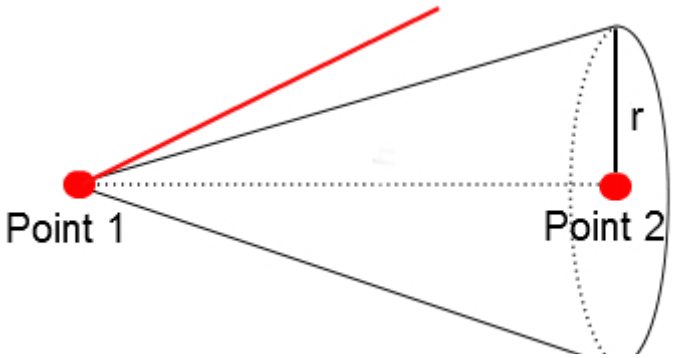
now how the system build those state and generate conditions for it ?

simply by taking each two points from the database as one pair and then make victor from them, and from that vector the program generate a simple cone its radius is the acceptable error rate-which is constant determined experimentally-. if the runtime vector lies on the virtual cone it will be acceptable for the state and it will generate the next state.

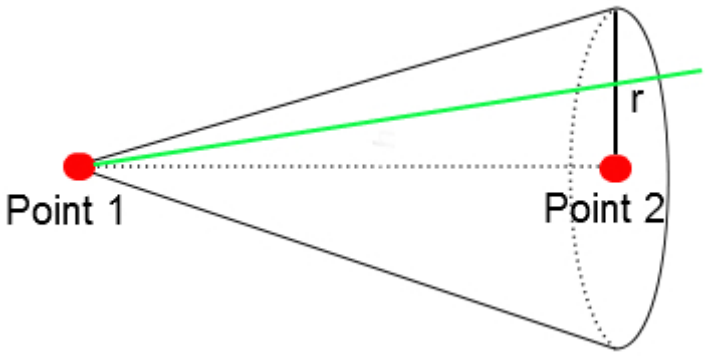


this shows the virtual cone for each state as its condition to start the next state, where (r) is

the error rate.



this is example on not acceptable case



and this is an example on acceptable case

B- shape matching Unit in details

the program uses dynamic time wrapping algorithm but not directly , first i was watching videos about kinect applications then i saw this video

http://www.youtube.com/watch?v=j9JXtTj0mzE&feature=player_embedded
which takes the depth channel as layers and gives each layer a different color. so i decided to do the same with the hand and divide the depth of the hand into different layers and take the contours for every layer and

compare it with group of database shapes , by this the program would be able not only detect the shape of the hand and the states of fingers it as able now to detect the orientation of it and maybe in the future by increasing the depth resolution it will be able to find the angle for the fingers, i mean build a complete 3D model for the hand. the system currently takes two or three contours for hand layers and its enough to know the hand state , but to get more details we need more layers and templates to compare.