# Approach

The approach we are going to use consists in four steps. (1) Defining and capturing the data required to recognize the user´s hand. (2) Preprocessing the captured data to ensure the correctness of it. (3) Defining features and segmentation of the data to create the feature vectors for each instance of the data. (4) Selecting a classification algorithm for training with labeled data to create a classification model and classifying the new unlabeled instances, this is recognizing the unknown captured hand gestures.

## Image capture

The data for this project are going to be images captured in which the users are performing some sort of gestures with the hand. Each image’s data will be the color information of the scene and we are going to add the depth information of each pixel. This information is sufficient enough to get good results as stated in some previous works [REFERENCES].

The image acquisition has two main steps to capture suitable data. (1) The first step is the selection of the capture sensor: there is a considerable big amount of sensors to select from but we have to select the one that best fits for our thesis. (2) Capturing images in the real world may produce some noise in the data, so defining an ideal world for the capturing is crucial: this ensures low environment variance.

### Sensor selection

There are many ways to capture images of users performing Sign Language, but in our project we are going to use color images and the corresponding depth images. Capturing the color images it is done by common video cameras but for the depth data we need a different type of sensor. Sensors like Kinect, Time of Flight (ToF) cameras or stereoscopic cameras allow the measurement of the depth information from the scene.

|  |  |  |
| --- | --- | --- |
| Time of flight | Kinect | Stereoscopic |
|  |  |  |

The ToF cameras can produce an accurate depth images at a high frame rate (50 fps), but they have a relatively low resolution, up to 176x144. The low resolution may not be able to detect some features of the scene, and thus the segmentation or recognition may not be the correct one. Since the sensor, to capture the depth information sends a light signal and then measures the time the signal spent traversing the scene to get the depth, lighting conditions are an important fact for this type of sensors: bright lighting may affect to the measurement of the sensor and get wrong values.

The Kinect camera has a VGA camera with a 1920x1080 resolution; therefore it can capture color images in high resolution. It has a QVGA sensor with a 512x424 resolution too; it captures the depth information of the scene. This sensor has an optimal capture range which is set up at 1.5-3.5 meters; objects very close or very far from the sensor may not be measured well. The two sensors, VGA and QVGA, works in a frame rate of 30 fps, since the minimum frame to get fluid captures is 24 fps it is enough to record videos. The developers of the Kinect sensor developed a SDK for using the camera and track the bodies in the scene, so it avoids the need to create our own body tracking system. The Kinect has some problems capturing the images; the most common problems are the reflective materials and that the capture may not work well with bright lighting conditions.

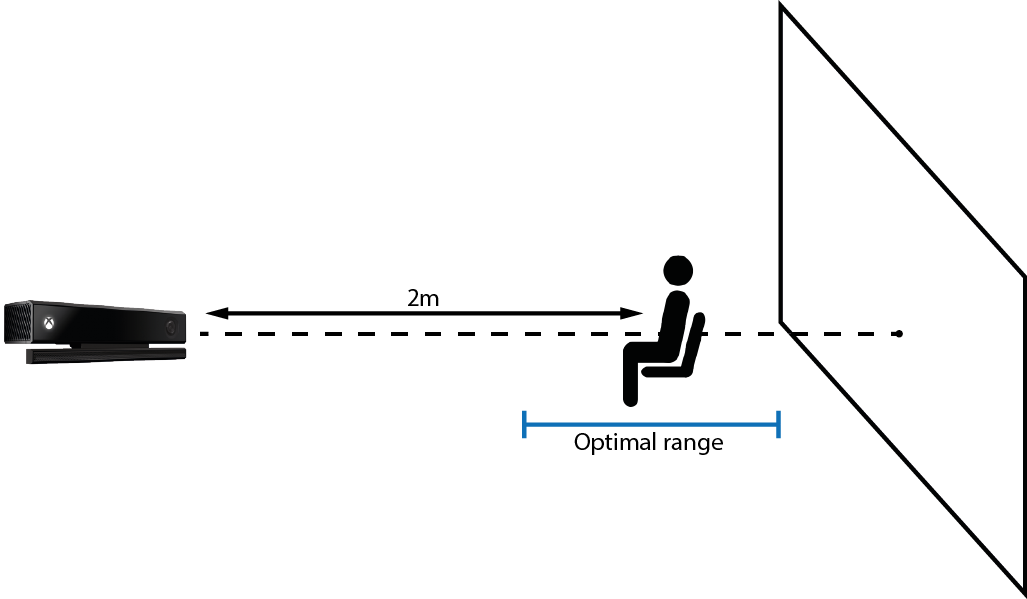
Stereoscopic cameras work differently than the previous two sensors, these types of cameras use two cameras to get the depth information. They simulate the human eye system to measure the depth, they have two cameras with a small distance between them: the right and the left camera. Since the two cameras are pretty close and there is a variation in the location of the cameras it is possible to triangulate the position of each pixel, and thus get the depth of each pixel in the scene. This technique has a high computational cost and the resulting depth map has low precision. The good points of these type of cameras is that they can be built with normal cameras, they work well with different lighting conditions and the resolution and the capturing frame rate can be very high, even though when computing the depth map the system´s frame rate may decrease.

For this project we are going to use the Xbox Kinect camera. This camera is very cheap for the resolution and the precision that it offers and any user could afford it. It has more capturing sensors than just the color and depth sensors, like the infrared and audio sensors. The frame rate is enough too for this project, 30 fps are sufficient to get good captures and to track the bodies. Since the developers provide the SDK to work with the sensor and track the human bodies the amount of work is considerably reduced. In many approaches for hand gesture recognition [REFERENCES], body tracking and Sign Language recognitions, this camera has been used and there is a lot of documentation and forums to learn from. The Kinect camera, as stated before, has some problems, but defining an “ideal” and simple world for recording the users will avoid that problems, so that problems will no longer have the significance they had before.

### Ideal world definition

Depending on the complexity of the scene (colors, reflective materials, and other objects in the scene) the capture may not be good enough to detect, recognize or classify the signs developed by the user. Creating an “ideal” room is a must if we want to achieve good results for our Hand Gesture recognition. This scene should not have anything that will affect to the capture from the Kinect camera.

When capturing each frame of the gesture we are going to use the color and the depth data provided by the camera. These two sensors have to get the frames in which the contents are clear to be able to segment in a proper way.



For the color sensor, the scene must not have a background with a big variation of color, only it is accepted for the pixels that represent the captured user. This will help to segment a person from the background. In this project we are going to use a white background to record all the signs.

Since the depth sensor is a light coding sensor (the camera projects a pattern into the scene and with the variations of that pattern is capable to determine the depth of each pixel) the projected light may be reflected in different ways on some materials. This is a problem because the sensor can measure wrong depths for the pixels that fall in those materials. Eliminating every object in the scene and having only a smooth non-reflective background will avoid these types of problems.

The position of the user in the scene is important too because the depth sensor has a limited optimal range of capture. This optimal range is between 1.5m and 3.5m, so we are going to set the distance of the user at 2m and we will place it in the center of the field of view of the camera to get the best space to capture. The user should be able to move the hands with freedom when performing any sign. Since the gestures are done with the upper half of the body we can capture users sitting or standing up, but in our opinion being sitting the user will fit correctly in the field of view and it is more comfortable for the user, therefore all the users must to be sitting.

Finally some algorithms will require some extra features to perform well, so optional requirements may be added in the future.

## Preprocessing

Since the areas we are going to focus on are the hands we have to extract the two hand areas from the entire image. Once the hands are extracted another problem appears: the size of the user´s hands may vary; not all the people have the same hand size. The variance in the depth of the hand is problematic too due to de difference in the pixel values of the image, so this is another problem we have to solve. We have to normalize al the images to have the same position, size and similar depth values for all the hand segments, therefore once we solved this dissimilarities we should be able to compare the different images.

### Hole filling

The depth sensor of the Kinect has some errors due to the capturing method it is used. Big differences over the surface of the scene produce a shadow on the background with null pixel values. In our case shadows in the hands are quite problematic because they may introduce some noise on the hand image. We have to study some hole filling techniques to be able to fill this kind of holes and to ensure the continuity of the pixel values in the image.



### Hand extraction

The Kinect camera provides the tracked user’s body information with 25 body joints, some of them are hand joints. From this joints we can map the 3D body to the image and crop a surrounding area around the hand. In this cropped image there are some background pixel values which must to be discarded to segment the hand. We are going to use the depth information and a thresholding technique to extract the hand as in [REFERENCE]: usually the closest values to the camera are part of the hands; this way the only positive values of the image are part of the hand.

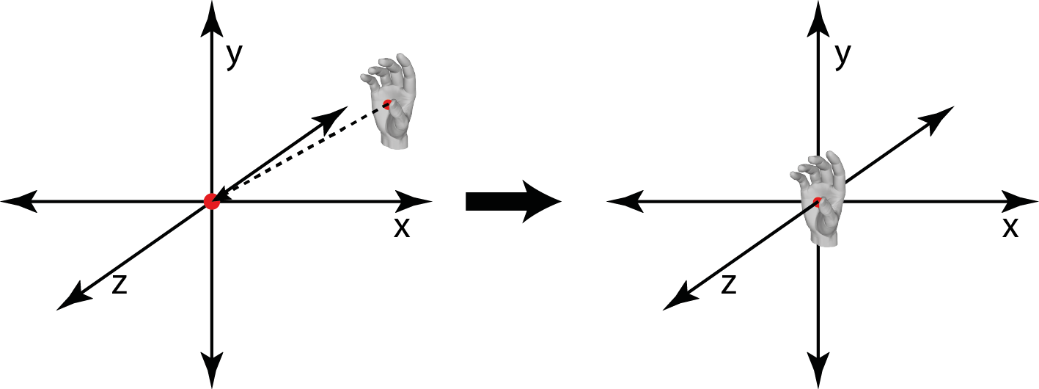
|  |  |  |
| --- | --- | --- |
| Front view | Side view | Histogram |
|  |  | PUT THE \*\*\*\* HISTOGRAM |

### Normalization

Signs recorded at different times and with different people are not going to be always the same size and the hand distance from the camera is going to be different. There are two steps in the normalization process: (1) the hand size normalization and (2) depth normalization. This way when classifying the hand gestures all of them will fit in the same space.

People will have different hand sizes and when trying to match the extracted hands with the already known signs the comparison will not be exact enough for our purpose. Some prior work [REFERENCE] used the body information to resize the hand; the problem arises when the user is performing a gesture towards the camera and the skeleton of the hand is hard to track for the Kinect. Another way to perform this step is to use the shape/area/width to match with a previously defined standard hand size.

Even though we have defined an ideal world, the hands positions will have some variance respect other users, and this may lead to some errors. We have to adjust all the hand pixel values to lay in the same range as the other users’ hands. Computing the average value of all the pixels will return the barycenter of the hand; this point can be translated to the origin and applying the same translation to the 3D pixel points of the hands we will get the hand centered on the origin.



# Feature definition and extraction

Before performing a classification we have to know what key features of the image we are going to consider for representing the images. This way the amount of data is considerably reduced for classification purposes. We should perform a study about the common features/filters used in image processing, select the best ones for this problem and apply them to the images. If in our opinion these features are not the best for this problem we should define our own features to apply.

The features selected should represent the gestures the best way possible to avoid the scalability problem when adding more gestures. When adding more gestures to the system it is harder to know which gesture is being performed and thus the precision of the system is reduced considerably.

We can combine different features to detect the key points of the image too. As an example, we can detect the borders of the hand and then apply another feature to get the features of the hand´s borderline.

## Laplace filter

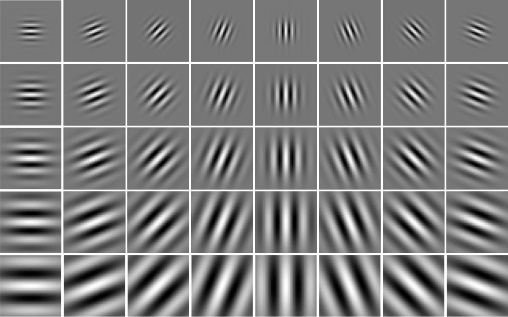
The Laplace matrix usually is used for edge and border detection. Since the Laplace matrix for 2D is defined as, when applying to each pixel in the image (not to the borderline pixels) when there is a big difference in the contiguous pixels the actual pixel is raised, and when there is a smooth transition the value of the pixel does not change a lot.

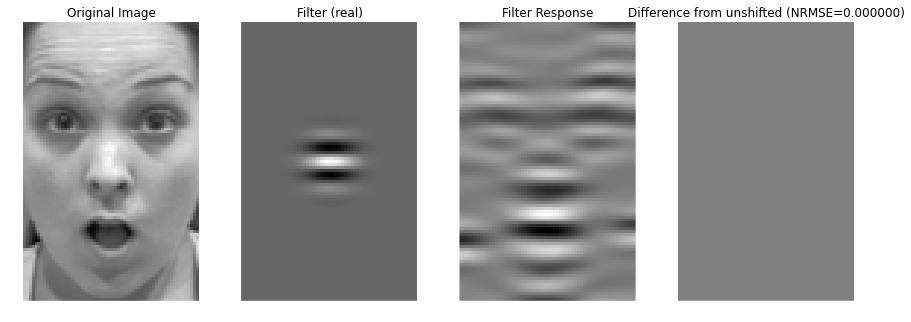
The images we are going to process with this filter are depth images, therefore if in the segmented hand shape the fingers are over the palm we should be able to detect them and have a basic information about the gesture shape it is being performed.

## Gabor filter

The Gabor filter has been used for edge detection in the frequency domain and for face recognition. Different wavelets with different frequencies, orientations and sizes are applied to the image and depending on the intensity transitions over the contiguous pixels some of them are raised and other ones are lowered.



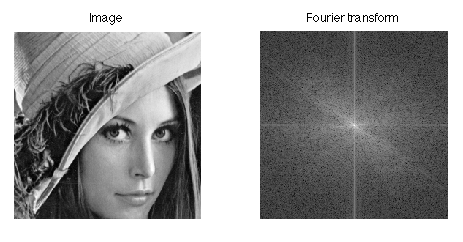


## Fourier transformation

The Fourier transformation discomposes a function into its elemental sinusoidal functions; a continuous function can be represented as the sum of different frequency sinusoidal functions. This transformation can be extended from 1D to 2D, this is, an image. To apply the Fourier transform to a digital image an adaptation of the formula has to be done, we have to convert the continuous formula (integration) into a discrete formula (in this case a sum):

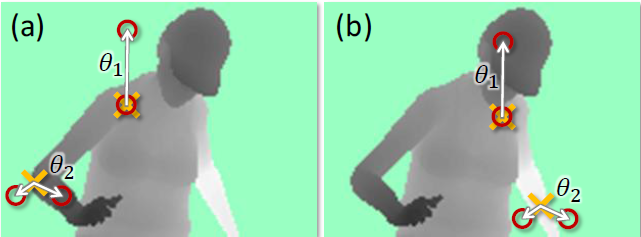
Where,

The computation of the Fourier transformation over an image transforms the image into its principal frequency components creating a new image. This new image is invariant to the scale and rotation, and is the codification of the original image based on sinusoidal components.



## Custom features

Common filters and features for image processing may not be sufficient enough for this problem, therefore we have to study different possibilities and custom features to apply to the segmented hand images. In [RealTimeHandPoseStimationUsingDepth] use a custom feature for the Random Decision Forest. Each feature provide some information about where each pixel lays in the hand.



These new features have to define uniquely each gesture to ensure the scalability of the problem, with more gestures the probability of having similar values for different gestures increases and thus the system precision may decay.

We can combine different features into a unique feature: for example, we can compute the Laplace filter to detect the borders and finger positions, and then use the Fourier transform to know the encoding of the image.