Intelligent Interactive Systems (course 1MD032)
Uppsala University – Spring 2016
Project Report by CVML Group 8
Project 1.4 — Gesture Recognition using OpenCV

Alexander Ek, Robert Källgren,
Andreas Widmark, Mikael Östlund

***Abstract*—This report explains the process of creating a web camera based Python application that uses OpenCV and machine learning techniques to recognise hand gestures from users in order to allow them to place orders in a fictional bar. The system uses text-to-speech to communicate with the user in combination with text output. Our evaluation of the system shows that we have achieved a 82% precision rate over 8 gestures, with two gestures being difficult or sometimes very difficult to detect.**

I. Introduction

As technology progresses we are able to start experimenting with alternatives for traditional ways of getting things done. One of the most traditional way of communicating is through spoken language, but in some situations it might not be the optimal way. Deaf people is a demographic that need another alternative, and in noisy environments it might be difficult for automatic speech recognition software to perform well anyways. A potential solution is to rely on communication through gestures instead.

We decided to develop a bartender robot (hereafter, the system), through computer vision and machine learning, is able to detect and identify hand gestures from a live web camera feed. More specifically the system acts as a bartender which allow users to place an order in a fictional bar.

The goals for the system are the following:

- Customers should be able to order either one alcoholic or one non-alcoholic drink (we assume that those are the only drinks served in this bar) and the meal of the day.
- Customers should be able to pay in cash or with credit card.
- There should be eight gestures related to ordering drinks and food in the bar.
- The gestures should be distinguished between using machine learning.
- The system should respond via written text output to have a simple dialogue between the system and user.
- The system should be trained using data that we collect from users.

The purpose of this report is to explain the process of the creation of such a system as well as evaluation. The report is organised as follows; in Section II the methods used are described. In Section III we present the results. In Section IV the results are discussed. In Section V we suggest what future work could be done to take the system further. Finally, in Section IV conclusions are drawn.

II. Methods

This section introduces the methods used in the context of the project. During the project we continuously posted about our progress on a dedicated blog that can be found at https://iiscvmlgroup8.wordpress.com/.

The main tools used for creating the system were *Python* using the libraries *OpenCV* and *Sci-kit Learn*. OpenCV is a widely used open source library for computer vision, and Sci-kit learn is used for machine learning.

A. The Ordering Process

The process when placing an order looks like the following:

1) The customer initiates the process by doing an opening gesture.
2) The program asks if the customer wants an alcoholic or a non-alcoholic drink.
3) The customer responds by making a gesture for either an alcoholic or a non-alcoholic drink.
4) The program asks if the customer wants the meal of the day.
5) The customer makes a gesture for either a yes or a no.
6) The program asks whether the customer wants to pay with cash or credit card.
7) The customer makes a gesture for either cash or credit card.
8) The program asks the customer to confirm their order.
9) The customer makes a gesture for either a yes or a no.

The customer can abort the process at any time by making the corresponding gesture.

The system can detect eight different gestures as shown in Figure 1, one for each action listed below.

- Initiate the ordering process
- Order an alcoholic drink
- Order a non-alcoholic drink
- Respond with “yes”
- Respond with “no”
- Wish to pay with cash
- Wish to pay with credit card
- Abort/reset the current order

The gestures were primarily selected based on three factors:

- Physical performance difficulty and how renowned the gestures are. The difficulty level had to be low in order to minimise the learning curve.
- Difficulty for the system to distinguish features from the images of the gestures.
- Relevance to the meaning of the gesture in the context of placing an order. For instance, the gesture to initialise is similar to a greeting (waving hand), and the gesture for “yes” is well known as a symbol for “OK”.

B. Pre-processing

To avoid having to search through the whole frame to find the user’s hand, which could introduce additional noise if the background is complex, we decided to make the program require the user to position their hand in a corner of the frame.

1) Image Pre-processing: To reduce unwanted noise in a frame, which would yield less effective feature extraction, we decided to perform image pre-processing on it. The process involves converting colorspace from RGB to grayscale, followed by applying a low pass filter to mitigate “salt-and-pepper noise”. Lastly a threshold is applied to make the frame binary. Once this has been done, the frame becomes very manageable as a clear distinction is made between pixels of significance.

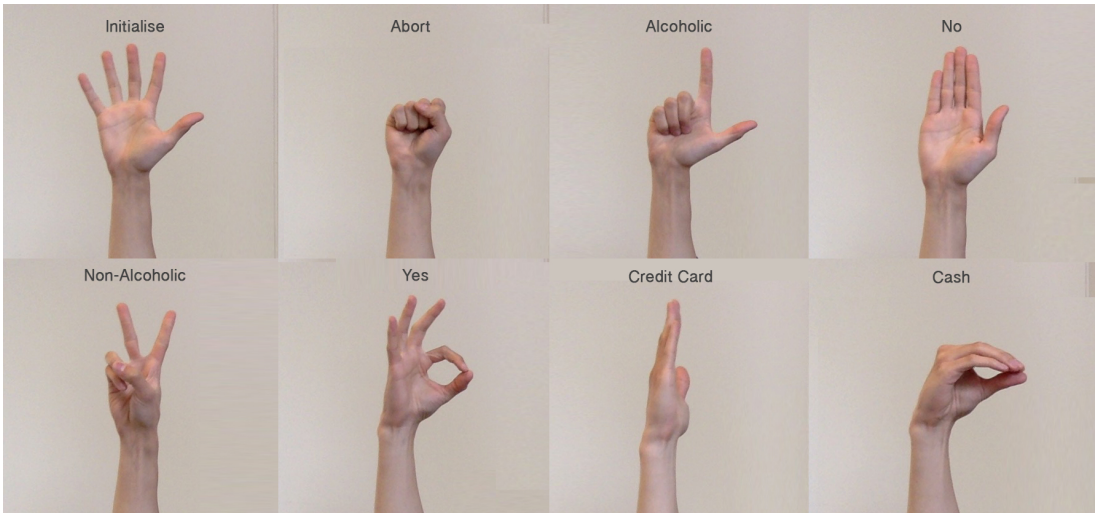

Figure 1. The eight different gestures used for placing orders in the system.

2) Detecting a Stable Gesture: When detecting a gesture from the user the system must first ensure that the gesture is stable. This must be ensured since transition between gestures may more often than not result in the wrong gestures being recognised by the system. In order to reduce this noise factor, we required that the region of interest surrounding the hand of the user to have been stable in 15 frames before trying to identify a gesture. The *region of interest* (*ROI*) is the minimal square that contains the whole gesture. We say that this region has been stable if its x- and y-coordinates as well as the height and width has not changed more than 10 pixels between frames.

C. Feature Extraction

In order for the program to successfully distinguish the gestures it needs to analyse each video frame and determine which gesture to classify the hand posture in them as. Therefore, there is a need to specify what the characteristic features are for each of the gestures, and to be able to extract them from an image.

Most of these features are inspired from the works of Pujan Ziaie et al. which presents several methods of feature extraction [1], [2], [3]. The following features were considered and tested in combinations:

1) **Convexity Defects.** This feature is the only that was not mentioned by Ziaie et al. First, the convex hull of the hand needs to be calculated. The convex hull of a shape is the smallest convex set out of the points of the shape that contains the shape. Between two vertices of the convex hull, the vertex of the contour furthest away from the convex hull is a convexity defect. One thing this allows is essentially a way to count the number of fingers extended from the hand.
2) **Contour Circumference.** The circumference of the contour of the hand, basically a counting of the number of pixels of the contour, then normalised with division by the square root of the area of the ROI.
3) X **and** Y**-Gradients.** X and Y-gradients are the number of changes of direction, in the X and Y axis respectively, that occur when following the contour of the hand. To not take noise into account, we require the contour to not change direction for a length equal to the third of the square root of the area of the ROI before counting that gradient change.
4) **Hu Moments.** Hu moments can be used to describe a weighted average value in an image. This average value can for example correspond to the pixels' intensities. The strength of these moments are that they are unaffected by translation, rotation or scale, and when utilised in an image moment context, they therefore provide a simple-to-calculate set of attributes that can be utilised to distinguish shapes in a binary image [3]. In this project moments up to the third order were used.
5) **Furthest Distance.** The furthest distance from the centre of the ROI to the contour. This is done by counting the number of pixels and is then normalised by division by the square root of the ROI.
6) **Gradient Spacing Deviation.** The standard deviation of the Euclidean distances between a gradient point and the next. The distances are the number of pixels, and each is normalised with division by the square root of the ROI.

Worth to note, that from the works of Ziaie et al., it was ambigious whether the normalisation was with division by the square root of the area of the ROI or the area of the hand. We tried both, and got better results from using the area of the ROI, and hence that is what we used.

D. Machine Learning

There are three main machine learning techniques that are used in this project:

- The k-Nearest Neighbour classifier (`KNN`).
- The Gaussian Naïve Bayes classifier (`GNB`).
- The Random Forest Classifier (`RFC`).

We use a weighted `KNN` with a k-value of the square root of the number of data points.

`GNB` is simple probabilistic classifier which applies Bayes' theorem and assumes independence between the features. It assumes that the values are distributed as in a Gaussian distribution [4], [5].

`RFC` is a meta estimator that fits a number of decision tree classifiers on different sub-samples of the data set. It also uses averaging to control over-fitting and improve the predictive accuracy [6], [7].

With the goal of normalising the precision and accuracy of the gestures somehow, we combined all three classifiers by introducing a voting classifier, where the included classifiers take votes on each

input pattern and the output with the highest value is picked as the label. Voting classifiers in `sklearn` allow for taking a weight vector as input, to put more (or less) emphasis on specific classifiers. We denote the combined classifier as Ensemble Classifier (`ECLF`).

To evaluate the performance of the classifiers, we mainly use *Leave One Person Out Validation* (`LOPOV`). This validation test consists of training the classifier on all but one of the person's gesture data set. Then, test how well the classifier classifies the gestures of the left out person. This is then repeated for all persons. The metrics are the percentage of gestures that classifies correctly, for each person, gesture and in total.

Also, a different evaluation method was used, namely a regular 7-fold cross-validation like the one included in the `sklearn` library.

We also used confusion matrix as an evaluation metric by training and testing on a random sample of the data.

E. Data Set

The data set consisted of pictures of each of the eight hand gestures from seven different people. For each gesture, pictures was taken with three positions of the hand: far away from the camera, near the camera, and in the middle of these distances. For each of these distances, five pictures was taken: where the hand was leaned to the left and to the right (with respect to the camera), where the hand was rotated around the axis of the arm in both directions (about 45°), and where the hand was neither rotated nor leaned. This makes 120 pictures per person (40 per distance), 105 pictures per gesture (35 per distance), and 840 pictures in total (about 100 MB of data).

F. Confidence Threshold

To avoid that undefined gestures or "nonsense" pictures (e.g. the background when no one is using the system) to be classified as some of the eight gestures specified in this report, a confidence threshold was used in combinations with the classifiers. The confidence threshold is set to 80%. Hence, if a picture is too different from one of the eight gestures defined in this report, nothing will happen.

III. Results

During initial evaluation of the data set we realised that the gestures captured from the far distance had less convexity defects than they should. We figured that this likely had to do with the picture resolution and we simply had to remove all pictures under the far distance category since they significantly hurt the performance of our system. Possible impacts of this are explained in Section IV

We also discovered that the **contour circumference** was not a good feature to include since while this increased the overall accuracy of most classifiers it significantly reduced the precision of the `abort` gesture. Thus we decided to remove this feature when training the system. Furthermore, the **gradient spacing deviation** feature proved to not affect the performance of the system most of the time, and lowered the performance in some cases. Therefore, that feature was also excluded from the feature extraction.

In the work of Ziaie et al., it is concluded that the seventh Hu moment added no value to the accuracy of the classifier [2]. We examined this in our implementation and our results does not correspond to the one of Ziaie et al. We obtained nothing but a fall-off in performance when removing this moment.

Most of the compatible classifiers available in `sklearn` were tested on the data, but no extensive testing for parameter tuning was made for each explored classifier. Most classifiers measured an accuracy around 60% when performing a 7-fold cross-validation using random sample. Only a few classifiers measured an accuracy of 80% or above. The overall best classifiers that we found were `GNB`, `RFC` and `KNN`.

Even though both `GNB` and `KNN` had an accuracy measure below 80% we noticed interesting properties when investigating the confusion matrices produced through holdout testing by random sampling of the data. Both `GNB` and `KNN` had worse performance than `RFC`, with 63% and 73% accuracy respectively, compared to the 84% accuracy of `RFC`. However, the classifiers did not sport the same properties overall. When examining the confusion matrices for the three classifiers, we found that `GNB` was the only classifier with a very high precision on the `Abort` and `Alcohol` gestures. `KNN` on the other hand had a somewhat low precision on the same gestures, with a precision of 70%, but it had a 100% precision on the `Cash` gesture (the value for `GNB` was 42%). Thus, these two classifiers somehow complement each other. `RFC` had a high precision for most gestures, but only 67% on the `Abort` gesture and 77% for the `Alcohol` gesture.

The three classifiers were then combined into one, namely the `ELFC`. Since `RFC` had much higher accuracy compared to the other two classifiers, more emphasis was put on this classifier. After some testing, we went with $weights = [1, 4, 1]$, where index 0 was the weight for `GNB`, index 1 the weight for `RFC` and index 2 the weight for `KNN`.

Running the `LOPOV` test on the trained `ECLF` rendered the results found in Table I. The results show that the system is overall very accurate, with an precision of 82%. However, some gestures perform badly. Such gestures are the `Yes`, `Abort` and `Alcohol` gestures. This is also apparent in online testing, where it is often difficult to abort an order and where the `Yes` gesture is very angle sensitive. Other gestures tend to work fine however, even `Alcohol` despite it's relatively low precision in the `LOPOV` tests. For a confusion matrix for `ECLF`, see Table II.

To see the effects of the features featured in our processed data set, we tried enabling and disabling different combinations of features in our feature vector. We noticed that when removing all the Hu moments our system, we measured a LOPOV precision of 59% (in contrast to 82.5% when enabling all features). However, disabling our custom made features and only using Hu moments we measured a LOPOV precision of 74%. Disabling parts of our custom made features made no improvement compared to enabling all features altogether.

In terms of overfitting the classifier we did not allocate much time into avoiding this. `RFC` does include mechanisms to minimise overfitting however.

A. User Interface

We created a simple user interface that displays the most relevant information to the user. The elements displayed are the following:

- A rectangle in the upper right corner of the screen that indicates the area where the user should put their hand in order to perform gestures

Name \ Gesture	No	Yes	Credit	Non-alc.	Alc.	Cash	Abort	Init	TOTAL
Alexander	0.9	0.5	0.9	1.0	1.0	1.0	0.7	1.0	0.875
Mikael	0.9	0.6	1.0	1.0	0.6	0.9	0.4	1.0	0.8
Volunteer_0	0.9	0.5	0.8	0.9	0.7	0.8	0.7	1.0	0.7875
Robert	1.0	0.2	0.9	1.0	0.6	1.0	0.9	1.0	0.825
Volunteer_1	0.6	0.9	1.0	1.0	0.6	1.0	0.8	0.8	0.8375
Volunteer_2	0.8	0.6	0.8	0.4	1.0	0.9	0.9	0.9	0.7875
Andreas	1.0	0.8	0.9	0.9	1.0	0.7	0.8	1.0	0.8875
TOTAL	0.7625	0.5125	0.7875	0.775	0.6875	0.7875	0.65	0.8375	0.8286

Table I

LOPOV PRECISION FOR THE DIFFERENT GESTURES WHEN TESTED ON DIFFERENT SUBJECTS ON THE ENSEMBLE CLASSIFIER ECLF

Actual \Pred.	Abort	No	Alc.	Yes	Init	No Alc.	Cash	Credit
Abort	21	0	0	0	0	0	0	0
No	0	11	0	0	0	0	0	1
Alc.	0	1	14	1	0	0	0	0
Yes	2	0	0	8	1	1	0	1
Init	0	0	0	0	10	0	0	0
No Alc.	0	0	0	0	0	16	0	1
Cash	0	0	0	0	0	0	13	0
Credit	0	1	0	0	1	1	0	7

Table II

CONFUSION MATRIX OVER THE 8 DIFFERENT GESTURES FOR THE ENSEMBLE CLASSIFIER ECLF

- A question like "Alcoholic or non-alcoholic drink?" that the user should answer in order to progress in the order, as described in Section II-A
- For each available response to any given question, a picture that demonstrates the corresponding gesture as well as a text that explains what the gesture means, e.g. "Alcoholic", "Non-alcoholic" or "Abort".
- A list of the items that have been added to the order
- The user's chosen payment method

IV. DISCUSSION

In this section, the results from Section III are analysed and discussed.

Using pictures where the hand was leaned towards and away from the camera as well as the other pictures could have improved the performance of the system. A big reason for this could be that a lean towards the camera is a common (and to some degree, natural) positioning of the hand. Something we did not consider when collecting the data.

The removal of the data points with the far distance led to the system being unable to recognise the gestures when the user was positioned far from the camera during online testing. This is not necessarily an issue however since when applying this system in practice, one can simply demand that the user stands at a specified location.

While only using Hu moments as feature vector were somewhat precise on its own, our time spent creating custom made features were not in vain as the difference of eight percent points precision is somewhat of an achievement.

Instead of using hand gestures one could instead have used full body gestures. It could perhaps be easier to distinguish features from full body gestures. However, such a system would be more difficult to use in practice, since one would have to have more space at the desk of the bar in order to fit the user's whole body in the camera field of view.

One problem with our system is that the `Yes` gesture is the most difficult to detect. This is most likely the system's biggest flaw since this is a gesture that is commonly used and not a gesture that the user may mistakenly do. One such gesture however is the `Abort` gesture, the second-worst gesture in terms of recognisability in our system. The system often struggles with recognising this gesture which could be viewed as both good and bad. On one hand other gestures are rarely recognised as the `Abort` but on the other hand the ones that are falsely recognised as `Abort` are often the `Yes` gestures as witnessed in Table II.

If this system were to be used in a real situation, the performance could be dramatically improved by having separate classifiers for each stage of the ordering process. Since the system only expects up to three gestures per stage, it only has to distinguish between these, which would make the task much easier than distinguishing between eight at a time.

A. Ethics

With any system that interacts with users and collects data it becomes relevant to discuss its ethical implications.

1) Storage and Security: A critical topic for discussion is how to manage data. Who should have access, as well as how it should be accessed. As a precautionary measure, when our system has read a frame from the camera it only uses the a cropped image of the frame once in order to detect and classify a gesture. After this, the frame is discarded and never stored or used again. Under extreme circumstances however, because of the use of the camera, it is still possible to theoretically intercept a web camera's video feed.

2) User Exclusion: When designing a system for practical usage, user exclusion often becomes unavoidable. People with functional limitations, in term of disabilities, or limited motor skills may have difficulty using the system to its full extent.

The current system is only trained with data of right handed people which could lead to a slightly biased system. This design aspect should be taken into consideration for a final system.

The skin tone of the users might affect the ability for the system to properly recognise and classify gestures. The system is built for a white background, so a light skin tone will reduce the contrast between the hand and the background, which could cause problems for the gesture detection.

3) User Experience: Considering the environment of the final product, being in a bar, environmental user interaction aspects need to be taken into consideration. A system that extensively trigger philosophical thinking can lead to anxiety if the interface is not appropriate for its target group. However, if the design leans more towards triggering the heteronomous way of thinking, problems can occur and lead to other relevant issues because of the biological effects of alcohol.

V. Future Work

It could be interesting to use a subpar background (i.e. a noisy background) instead of a clean uniformly coloured background which have been used for this project. Not only is this a more likely setting for a bar, but it would also be interesting to see if this implementation works as well for those circumstances.

Although having an `Abort` action is quite relevant, It would also have been more intuitive if there also was an undo-action that undid the last action.

Additionally, Dense Scale Invariant Feature Transform (DSIFT) [8] is a promising technique that could be investigated.

Another interesting method to improve the classification of the gestures is to purposefully train on "trash" and label them as a trash class. These pictures could constitute pictures of backgrounds, irrelevant objects, or even invalid gestures. It would be interesting to see how much the performance could be improved with this method.

VI. Conclusion

The final product of the system works pretty well according to the authors, and the performance metrics show a 82% precision rate overall; however, the `Abort` and `Yes` gestures are somewhat troublesome for the system to recognise correctly. Apart from these two gestures, the system works as intended. We are confident that if we were to use separate classifiers for each stage of the ordering process, the performance would be sufficient for a working product.

References

[1] P. Ziaie, T. Müller, M. E. Foster, and A. Knoll, "A naïve bayes classifier with distance weighting for hand-gesture recognition," 2008.

[2] P. Ziaie, T. Müller, and A. Knoll, "A novel approach to hand-gesture recognition in a human-robot dialog system," 2008.

[3] P. Ziaie, "Implementing and evaluating hand gesture recognition for human-robot joint-action," 2008.

[4] Wikipedia, "Naive bayes classifier — wikipedia, the free encyclopedia," 2016. [Online; accessed 31-May-2016].

[5] T. F. Chan, G. H. Golub, and R. J. LeVeque, "Updating formulae and a pairwise algorithm for computing sample variances," in *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, pp. 30–41, Springer, 1982.

[6] Sklearn, "Random forest classifier — scikit learn documentation." [Online; accessed 31-May-2016].

[7] Sklearn, "Ensemble forests — scikit learn documentation." [Online; accessed 31-May-2016].

[8] VLFeat, "Vlfeat - documentation >c api." [Online; accessed 31-May-2016].

Appendix A
Contributions and Roles

During the course of the project we started by reading a lot of literature on the subject to find out what seems to be the go-to way when recognising gestures. Then we more or less branched the project a bit in two pairs. One pair, focusing on the data gathering and analysis (Mikael and Alex) and one pair focusing on the interface and interaction with the user (Andreas and Robert). However, we did not work on a completely disjoint set of problems. We spent most of our allocated time in this project by working in the same room and exchanging ideas. It was mostly implementation details that were divided between the pairs. But overall the former pair spent more time on back-end stuff such as data pre-processing and analysis while the latter spent more time with building the interaction with the end user.

Appendix B
Implementation

The system was implemented in Python 2.7.10 with OpenCV 2.4.9 and Sklearn 0.17.1. The source code is not publicly available, but can be given upon request.