Computer Vision Component Report for Prezence

Max Poynton (map213)

***Abstract*—A key requirement for Prezence is for it to be able to identify human gestures and posture. In this report I outline the key requirements of the Computer Vision module, explain the motivation behind using computer vision methods, describe existing methods, and propose a solution that is suitable for the purposes of Prezence.**

I. Introduction

During a presentation, something possibly just as important as the words we say, is the way we say it; from our body language to our eye-contact (or lack thereof) [7]. Inviting an audience to listen, or paying for a professional speech tutor is not an option for most people every time they have to practise a speech [9]. Furthermore, these good practices do not just relate to what immediately comes to mind when we think of a "presentation". In daily life, in our jobs, the ability to present our ideas clearly and persuasively is an important asset [10]. Feedback has been identified as one of the most important aspects of learning when it comes to improvement and the ability to retain corrective information [5].

It is important for the purposes of testing our second hypothesis that we provide real-time feedback. It is also therefore important that we capture accurately the behaviour of the Presenter, including their body posture, their gestures, their gaze direction and so on. Intrusive methods of doing this such as sensors worn on the body would not be suitable for these purposes [4]. It is our aim for the system to distract the user as little as possible. In fact, much care has gone into designing the real-time feedback to ensure that it is as subtle as possible while still remaining useful. Computer vision techniques are the natural solution. A camera can easily be directed at the presenter with no interruption and the field of computer vision provides powerful techniques for analysing the visual data retrieved.

II. Specification and Integration with Other Subsystems

There are three key elements which will need to be extracted from the images - gaze direction, posture, and gesture. Posture includes whether the Presenter has their back to the audience, and whether they are moving around too much. The head, body, and particularly hands need to be recognised and tracked in order to do this. To accurately gather data for the thresholding of behaviours based on the metrics, the vision system will need to be continuously gathering information about the Presenter, and transforming it into useful output, for example the direction vector of the Presenter's head.

The vision submodule will take the form of a single OpenCV program which takes as input the video stream from a Kinect 1.0 and outputs to a text file, which is polled at regular intervals by the Central Processor. The data taken from this output will have thresholds applied to it to determine whether it falls into different categories based on the metrics. In this way, the decisions, thresholds and metric definitions can be held together in just one place, so that if they are to be changed, this can be done easily through environment variables, for example, rather than having to edit a hard-coded implementation for each submodule. The information should be appended to the text file whenever a change occurs, rather than at a certain frequency. This is because some important information might be missed if the Kinect input is not polled continuously. An unfavourable gesture could happen quickly and be missed by the system, for example. In order to prevent this, the Kinect input will be continuously polled and whenever a change in gesture, posture, etc. occurs, the text file will be appended to. Included will be a timestamp of when the transition in behaviour occurred. In this way, the Central Processor will be not only be able to be sure it did not miss a gesture, but also it will be able to determine the duration of certain behaviours by comparison of their timestamps.

Specifically, an example output to the text file is as follows: [Forward,Pointing,1]. The first element represents the direction of the Presenter's head. This will be placed into categories dependent on the yaw and pitch as ascertained by the OpenCV program. For example, "forward" would be the result of a yaw and pitch of 90 °, "up-left" would be 135 ° yaw, 45 ° pitch. The second element represents the latest gesture, and the third is a binary value denoting whether the Presenter's back is to the audience. From here, the Central Processor will build up an understanding of with what frequency each gesture occurred, what proportion of the speech had the Presenter with their back to the audience, etc.

Finally, a method for the Presenter to begin and end the live session is required. This can of course be performed by the Speech Processing submodule. The Presenter would have to say a key phrase e.g. "Prezence begin", in order for the system to know when the speech had started. However, the Computer Vision submodule could also be used to implement this behaviour. A gesture such as a "thumbs up" could be used to start the session, and by raising and holding both arms in front of themselves, the Presenter could end the session (or some other appropriate gesture that is not likely to happen accidentally). This is similar to the official Kinect gesture used for calibration. It would be useful to us for this very reason; it provides a complimentary way of calibrating the Kinect before the presentation has started. The final implementation may use one, both, or neither of these approaches, but since developing such a functionality does not require very much more in the way of techniques than the current system, it can be made just in case it is useful.

III. Other Research in the Field

The problem to be addressed is classified in the field of computer vision as “looking at people”.

A. Head pose estimation

The solutions to the problem of estimating the direction in which a human head is pointing can be classified by the following three groups:

1) 2-D approaches without explicit shape models
2) 2-D approaches with explicit shape models
3) 3-D approaches [3]

2-D approaches without explicit shape models focus on using features extracted from a 2-D image. A large part of computer vision relies on the notion of “features” of an image. Much technical and even psychological work has gone into discovering suitable features to track. For example, the hand may be recognised by using the RGB information of the image and detecting areas of similar colour, corresponding to the skin.

Shape models can be used in 2-D and 3-D to improve the accuracy of estimation. The idea revolves around using a mathematical model of the head or face e.g. the 3D Basel Face Model [8]. It is a model comprised of a set of vertices. 3-D approaches may attempt to map what it thinks are the corresponding vertices in the input image onto these vertices iteratively, and return the most likely result of the position of the head/face.

Although it has been changed to an ”additional feature”, the use of the Hough Transform (which is available as a function in OpenCV - HoughCircles()) applied to circles, can result in good identification of the position of the eyes. The eye-gaze direction can then be inferred from comparison with other facial features and, along with the relative position of the head, the gaze direction can be inferred.

In terms of great accuracy, promising results have been obtained [11] using the built-in Lucas-Kanade calcOpticalFlowPyrLK() function and goodFeaturesToTrack() in OpenCV [15]. This determines the strongest edge features in an image and then uses their movement to better track the object. The algorithm uses optical flow to predict where it expects certain feature points to be in subsequent frames, thereby increasing tracking reliability and accuracy [17]. The method assumes that:

1) Pixels in proximity with one another will have similar movement
2) The intensity of pixels do not change much from frame to frame

B. Posture

There have been several cutting edge techniques for calculating head pose algorithmically. One such algorithm is the Particle Swarm Optimization [6]. The proposed algorithm maps depth images of different viewpoints of the head onto a hypothetical surface patch, and then compares them to a previously-obtained frontal view. The aim is to use an objective function to maximise their similarity at different predicted poses to interpret the actual pose.

C. Gesture

Despite the suggestion that gestures were the first and sole manifestation of communication among early human beings and primates [1], algorithmic analysis, detection and classification of gestures presents a number of challenges. For low-quality cameras, or poor lighting environments, indistinct features can make recognition less accurate or unattainable. Distance from the camera will also affect the accuracy, and since the Kinect has a 57° horizontal and 43° vertical field of view, the position of the Presenter in the frame is important. Maintaining accurate tracking and recognition when the user is in all parts of the frame is difficult, and occlusions of the arms behind the body present another challenge. Unlike face recognition problems, less concern is given to image-changing factors such as clothing. However, since in the context of Prezence, the human-being being looked at will be moving around, turning around, and may be holding notes, or pointing at their slides, or some other auxiliary material, the design must still be robust to variations in the appearance of the Presenter.

IV. Proposed Implementation

Having previously proposed that vision techniques are the least intrusive solution to the problem of obtaining position information about the user, there are still a few options to consider. A pair of stereo cameras could be used to extract depth information, or a single high-quality camera could be used to increase the accuracy of 2-D based algorithms. Since the system already uses a NAO, which possesses two cameras which provide a up to 1280x960 resolution at 30 frames per second, one possible option is to use these for the vision input method. However, due to limited processing power, we have already outsourced the processing to a laptop, and due to small frequency range and inconsistent sensitivity of the NAO internal microphone, we are using a MOVO MC1000 microphone for the speech processing. Due to it’s compatibility with OpenCV, the device of choice is the first generation Kinect. The Kinect’s RGB and IR depth-finding cameras provide flexibility when it comes to computer vision implementations - This is especially useful since skeletal tracking can be simplified by using depth information, whereas many popular hand-gesture algorithms use 2-D approaches. Furthermore, it is practically plug and play when it comes to accessing the video streams. OpenCV support can be obtained via the OpenNI library, and NiTE middleware and there are several open-source pieces of sample code that provide easy-access to some impressive gesture recognition techniques. The Kinect is also cheap and readily available.

A. Gaze Direction

It has been recommended that, due to time constraints, very accurate gaze direction detection based on eye tracking be set aside in favour of simpler indications of gaze direction such as head pose estimation. This was also a problem because it is difficult for one camera to be positioned such that it has an accurate view of the eyes at the same time as being able to fit the whole body in frame. Therefore, the problem becomes

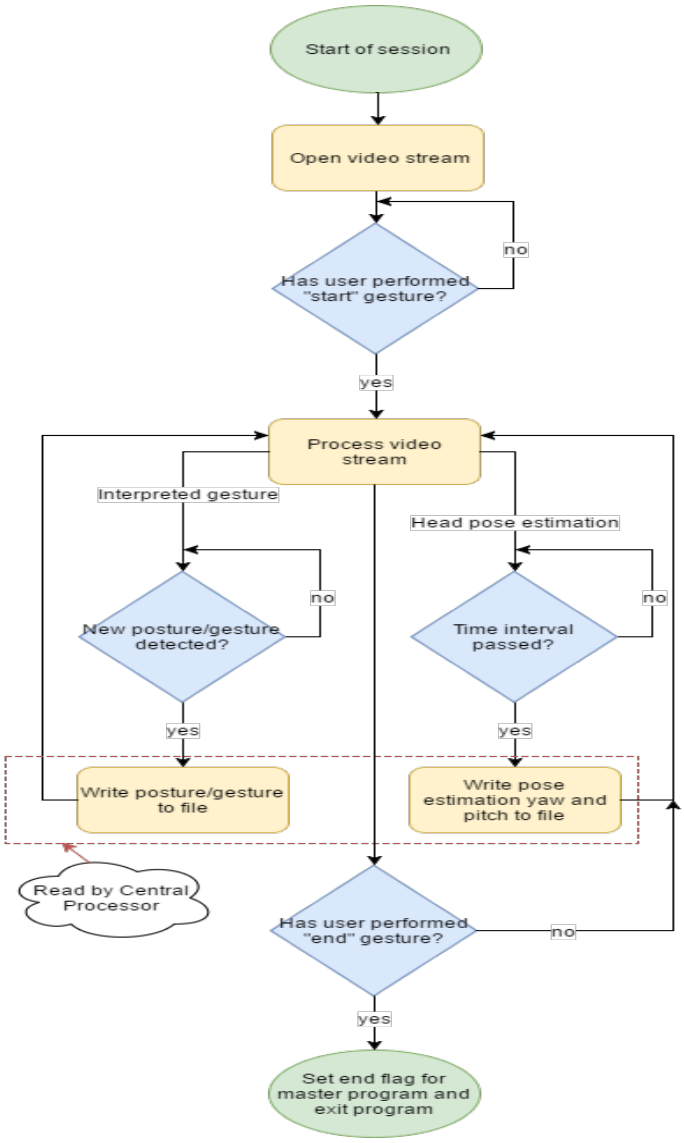

Figure 1. Overall Computer Vision Submodule

one of detecting the Presenter's head in an image frame, and estimating the direction in which the head is pointed. This problem can be simplified further by considering recognising a face in the image, rather than just a "head". The face has many features which are suitable to track [2]. For example, the corners of the eyes, the corners of the mouth, etc.

OpenCV (Open Computer Vision) offers a rather simple method of tracking faces and eyes using Haar Cascades - namely calling the detectMultiScale method on a CascadeClassifier object. The Haar Cascade is essentially a set of Haar features, which act similarly to convolution kernels. They produce a classification of an object (e.g. a face) based on properties of faces that it accumulates via machine-learning. The function returns a Rect(x,y,w,h) object with x and y coordinates, along with the width and height of the bounding rectangle of any faces found using the classifier [14]. This method has proved successful for drawing bounding rectangles around the face and eyes using open-source pre-trained Haar Cascades for the face and eyes. However, custom classifications can also be produced. This can be done by collecting a large number of "negative" images, which do not contain the object to be tracked. Then, a large number of "positive" images should also be created or found - these are images of the object itself [18]. OpenCV includes a command "opencv traincascade" which runs a training program on the sets of images to create the Haar Cascade [19][16]. Until comprehensive testing has been done, it is unsure whether the open-source face and eye clusters used so far will be sufficient for this application.

B. Posture Recognition

It has also been recommended that we aim to develop a working system with basic functionality before adventuring into the realms of emotion recognition and multiple speaker support. Therefore, the original single category of "Gesture Recognition" has been split into two subcategories, namely "Posture Recognition" and "Gesture Recognition". Another reason for this is that mentioned in the Specifications section - In order to start and stop the Prezence session, a gesture cue is required.

For the purposes of posture, Microsoft's Kinect SDK can detect and track a skeletal representation of the joints of a body via a few in-built function calls CopySkeletonDataTo(this.skeletonData)[12]. It also provides functionality to access individual joint positions JointType.ShoulderCenter for example. From this, custom definitions of a pose can be created. For example, if the y coordinate of the left and right shoulder joint increases and decreases in a short period of time, the Presenter may be "shrugging", etc. If this proves to be too inaccurate, or takes too much time to create custom definitions for each pose, Microsoft also provide the Visual Gesture Builder, which allows live poses recorded using skeletal tracking to build up a database of gestures. The library then allows run-time recognition of the recorded posed, along with a confidence score, which could prove useful. [13]

C. Gesture Recognition

A similar approach can be taken here as to with pose recognition, i.e. to use the pre-existing Kinect libraries to generate a skeleton representation of the joints of the hand, and to infer the hand gestures from these. Alternatives include extracting certain features of the hand, e.g. using active contours to model the curves of the outside of the hand. From this a model of the fingers can be constructed, which can be used to determine the gesture, using rules based on the position of various features of the hand, similarly to the gestures above.

Alternatively, if the Kinect does not provide the accuracy for this, the definition of gestures could be relaxed to just include more vague movements such as waving or pointing, where only the position of the hand is required, rather than that of the individual digits.

V. Conclusion

Visual information about the Presenter is valuable to the Prezence system, both for gathering presentation practices

data, and possibly for user control. Computer vision techniques offer some powerful methods for collecting and analysing such data and, depending on the required feature to be detected, the methodology varies substantially. Regardless, OpenCV can be used to great effect for the algorithms, and I have identified methods of doing so in the context of the Prezence project. The next steps in the development process are to finish implementation and evaluation of these different methods and to create the message-passing mechanisms detailed in this report in order to integrate the subsystem into the rest of Prezence.

References

[1] Gordon W Hewes et al. "Primate communication and the gestural origin of language [and comments and reply]". In: *Current Anthropology* (1973), pp. 5–24.

[2] Jianbo Shi and Carlo Tomasi. "Good features to track". In: *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE. 1994, pp. 593–600.

[3] Dariu M Gavrila. "The visual analysis of human movement: A survey". In: *Computer vision and image understanding* 73.1 (1999), pp. 82–98.

[4] Martin Tosas and Bai Li. "Virtual touch screen for mixed reality". In: *International Workshop on Computer Vision in Human-Computer Interaction*. Springer. 2004, pp. 48–59.

[5] John Hattie and Helen Timperley. "The power of feedback". In: *Review of educational research* 77.1 (2007), pp. 81–112.

[6] Pashalis Padeleris, Xenophon Zabulis, and Antonis A Argyros. "Head pose estimation on depth data based on Particle Swarm Optimization". In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE. 2012, pp. 42–49.

[7] Hitomi Yokoyama and Ikuo Daibo. "EFFECTS OF GAZE AND SPEECH RATE ON RECEIVERS'EVALUATIONS OF PERSUASIVE SPEECH." In: *Psychological reports* 110.2 (2012).

[8] Gregory P Meyer et al. "Robust Model-Based 3D Head Pose Estimation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 3649–3657.

[9] Jan Schneider et al. "Presentation Trainer, your Public Speaking Multimodal Coach". In: *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. acm. 2015, pp. 539–546.

[10] FT. *Importance of Presentation Skills in Today's Competetive World*. URL: http://www.ft.lk/2011/03/30/importance-of-good-presentation-skills-in-today%E2%80%99s-competitive-world/.

[11] infrared5TV. *Testing different Methods of Face Tracking using OpenCV*. URL: https://www.youtube.com/watch?v=JO8XHzc6JPQ.

[12] Microsoft. *Tracking Users with Kinect Skeletal Tracking*. URL: https://msdn.microsoft.com/en-us/library/jj131025.aspx.

[13] Microsoft. *Visual Gesture Builder: Overview*. URL: https://msdn.microsoft.com/en-gb/library/dn785529.aspx.

[14] OpenCVDocs. *Face Detection in OpenCV*. URL: http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html.

[15] OpenCVDocs. *Feature detection in OpenCV*. URL: http://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html.

[16] OpenCVDocs. *Haar Cascade Training in OpenCV*. URL: http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html.

[17] OpenCVDocs. *Lucas Kanade in OpenCV*. URL: http://docs.opencv.org/trunk/d7/d8b/tutorial_py_lucas_kanade.html.

[18] sentdex. *Making your own Haar Cascade Intro - OpenCV with Python for Image and Video Analysis 17*. URL: https://www.youtube.com/watch?v=jG3bu0tjFbk.

[19] sentdex. *Training Haar cascade object detection - OpenCV with Python for Image and Video Analysis 20*. URL: https://www.youtube.com/watch?v=eay7CgPlCyo.