Model-Based Techniques for Tracking Human Head for Human-Robot Interaction

Faisal Khan
University of Wisconsin, Madison
faisal@cs.wisc.edu

Varghese Mathew
University of Wisconsin, Madison
vmathew@cs.wisc.edu

ABSTRACT

Humans are known to use gesture cues lavishly in communication. Some estimates say more than 70 percent of human communication is non verbal. Therefore, identifying and tracking these gestures as well as mimicking them has emerged as a focus area in Robotics and Human Computer Interaction. This work looks at one of the aspects involved, viz. Human head tracking. Specifically, we seek to identify the yaw, pitch and roll angles of a human head from a live video stream and make a robot move its head in sync with the subject under observation.

1. INTRODUCTION

Humans' use different gesture cues e.g. gaze, hand and head movements to convey intentions and augment the spoken words. A conversation role among participants is often established by merely looking at the other person. In human-robot interactions where robots are required to take some of these roles, directing and controlling robot's gaze in reference to the direction of human participants becomes imperative [2]. Head tracking is intrinsically linked to a more general problem of gaze tracking. In gaze tracking we try identify the focus and direction of person's eyes.

This ability to quickly interpret orientation and movement of human head is very primitive to humans but for computer vision it is a challenging problem. The challenge is to handle a large number of variances in the captured images e.g. camera distortion, projective geometry, lighting and also difference in biological appearances. The exact description of what is meant by head pose detection can be tricky given how many degrees of freedom are desired. In this project we are targeting detection of full 3D human pose with six degree of freedom (DOF). The three-degree of face comes from yaw, pitch and roll of human face as shown in Figure 1. The remaining 3 degrees are position of head in the scene.

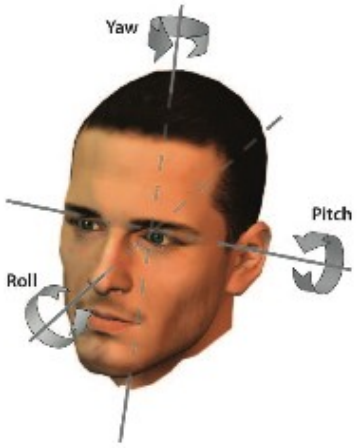


Figure 1: 3 degrees of freedom of human head (Picture taken form [1])

There are wide variety of techniques available in computer vision literature that solve this problem with varying degree of robustness, degree of freedom, complexity etc. In this project we focused only on methods, which are generally referred to as model based techniques. A more

detailed overview of our understanding of these techniques is given in section 4.

At a higher level these techniques are based on detecting features points of an object to track, mapping them to a 3D model of the tracked object and estimating pose preferably after filtering outliers from the previously modeled points. The primary reason for choosing model-based approach is their relative ease of implementation and freedom in choice of different individual components. For example, we can try out multiple feature descriptors (feature points based on Haar classifier, or more robust facial features like position of eyes, lips corner etc). Additionally, using this approach we can do our implementation in a more modular fashion i.e. trying existing methods for feature detection as explained in our implementation section and moving to more robust alternatives. One potential drawback with these techniques seems to be the assumption of having full frontal image of head in the image for initiating tracking.

The reminder of this paper is organized as follows: The next section formally states goals that we are interested in achieving in this project, section 3 gives a brief overview of related techniques that we looked at when selecting our method of choice, a description of our understanding of model based techniques is given in section 4, section 5 gives an overview of robot that we are using for our project, our approach is detailed in section 6, followed by results in section 7, and acknowledgments in section 8. The list of references is presented at the end.

2. PROJECT OBJECTIVES

The final objective that we intend to achieve through this project is to gain a deeper understanding of tracking the human face. Learning by example being the best way to learn, we created a closely relatable problem for us to tackle. Having access to a robot which has the capability to move its head with the same degrees of freedom as a human, we venture to develop a system that can track the head movement of the person interacting with the robot, and make the robot move its head mimicking the person.

The following description formally summarizes our objective in this project and how we are planning to achieve them. We are interested in:

1. Streaming image from robot front camera over wireless on to a computer with a single person in field of view of robot camera facing towards the robot. Alternatively, we can work with video stream from a web-cam attached to a computer.
2. Processing the above stream frame-by-frame applying methods for head-detection, feature-detection and pose estimation as explained elsewhere in this report. This step will give us a translation and rotation matrix of human head in camera co-ordinate system.
3. Sending instructions to the robot over wireless to move its head based on the rotation matrix retrieved in step 2. We will be using the POSIT algorithm for getting this matrix.

3. RELATED WORK

There are wide varieties of approaches available for head and face tracking that differs in method they use. In this project, we were mostly interested in exploring model-based techniques [5] [6] given that they are easy to implement and relatively robust.

An excellent survey of techniques in computer vision for human's head pose estimation is given in [1]. Here we give a brief summary of some of the methods, which we studied for our project. We encourage readers to look at [1] for a more comprehensive review.

Template based techniques (e.g. [8]) matches a view of a person's head to a set of sample images at different known head positions. A match is generally based on minimizing mean squared error between above two sets of images while comparing at multiple resolutions. More robust template based methods uses Support Vector Machines (SVMs) for matching face and estimating

head pose. Few of the drawbacks include need to have a large set of training images with estimated pose which is usually hard to gather and pair-wise matching in case of pose has the potential for a lot of errors. We initially intended to model our solution based on these techniques but decided otherwise given the difficulty in obtaining a accurately labeled dataset.

Another approach is *non-rigid head tracking* as proposed by Baker & Simon [3]. In their approach, they track the face as a whole, as well as, componentize the motion into individual elements like movement of the mouth, eye etc. This can be used to determine mental state / emotion of the person being observed. Towards this, they use Active Appearance Models (AAMs) with which they can perform this task at a very fast rate. They also build a 3D model from the 2D image and tracking the head with that is even faster as they claim. We really like there solution and consider it to be one of the best way to implement a robust face tracker where you can also track the individual movements of eyes and other face landmarks with respect to human head. Given the complexity of this implementation and short time we decided to go with slightly simpler model based techniques.

The solution offered by Bradski's paper describes another approach. In this a *continuously adaptive mean shift* (CAMSHIFT) algorithm is developed, which works on colors and color gradients. This algorithm is applied to video feeds to do a flesh tone based face tracking with 4 degrees of freedom. These techniques are good for tracking human face but we were not sure if they will prove any better at determining 3D pose.

4. MODEL BASED TRACKING

Model based tracking is one of the forms of visual tracking employed in computer systems usually to track rigid bodies. As the name suggests, such a tracking system uses a model of the object being tracked to identify and track the same.

There are four basic steps to any model based tracking system:

Measurement: This step covers determining where the object is, and identifying key feature points on the object that correspond to the feature points as specified by the model. As far as head tracking is concerned, one can imagine many ways of going about with the detection of feature points. These include SIFT, Haar classifiers, image thresholding to identify eyes, nostrils, mouth etc.

Pose estimation: In this step, the feature points detected are matched against the model to construct the 3D orientation and location of the object with respect to the camera. The POS algorithm [4] is one that can be used to accomplish this.

Filter: Oftentimes, the feature point detection and pose estimation brings in some amount of noise into the calculations. In this step, we filter out such noise components. Towards this, we can use some form of smoothing function based on the assumption of smooth motion for rigid bodies. For head tracking, we could find the best pose estimate by rejecting outliers using the RANSAC method, as described in the POSIT [4] algorithm.

Prediction: Since the problem at hand is that of tracking the object in real-time from a video feed, it helps to use the past information to predict the new location of feature points. This is founded on the assumption that rigid objects flow smoothly. Using the past in this way to predict the feature point location helps in narrowing down the search area for feature points. This accelerates the process of locating feature points thereby improving real-time performance of the algorithm. Methods like tracking optical flow using Lukas-Kanade method [12] can be used to achieve this for head tracking.

Most of the model based solutions vary in their choice of techniques or solutions that they employ at different above mentioned stages. This lead to one of our main rationale of choosing model based techniques as we can try out multiple solutions and compare their results.

5. WAKAMARU

Wakamaru (see figure 2) is a domestic humanoid robot built by Mitsubishi Heavy Industries. The primary intent of the robot was as an aid to the elderly and disabled. The robot is 1m tall, has two arms and a head which it can move like a human head. [13] It runs a Linux operating system on an Intel microprocessor and has an in built wireless card.

Wakamaru has two cameras on it, one at the front and one on the top of its head. The latter is basically for panoramic view of surroundings. The robot has a scripting API for interacting with its different components like camera, sound, motion etc. It has full 3 degree of freedom head movement that can be controlled programmatically.

Figure 2: Wakamaru humanoid robot from Mitsubishi

6. APPROACH

We focused on using OpenCV[14] for our implementation. Our choice was motivated by many factors. OpenCV is written for C/C++ environment which is something we and most of the research community is very familiar with. At the same time, the library has Python bindings for quick prototype testing. The library already implements functions for a lot of what we call "infrastructure requirements" like capturing a video stream from camera, applying many common image processing procedures etc, allowing us to focus all of our energies on the task at hand rather.

We've so far worked on two different techniques for identifying feature points on the face of the subject. One is using Haar Classifiers for the different features on the face like eyes, nose, lips etc. The other is using image thresholding to identify the feature points [6].

In using the Haar classifiers, we made use of training sets that the research community have developed. These training sets can identify the eyes, nose, lips etc on the human face. With this technique, we first run a face detector to locate a frontal face view of the subject under observation. We assume a frontal face view to start with since most commodity face detectors are best trained for frontal face view, and on the other hand it's reasonable to assume that a subject interacting with a robot would start the interaction with a frontal view. Next, we apply the Haar Classifiers trained for facial features on the detected face view, within geometric constraints to identify the features and feature points. Geometric constraints help us in two ways. For one, it helps eliminate many false positives that can arise from background noise. Second, it speeds up the matching by narrowing down the search space. We also use a flow based tracking in this method, where we predict the geometric constraints for the features in the next frame rather than re-run the face detection.

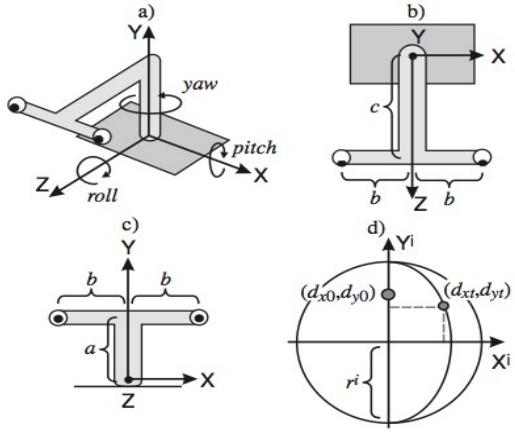
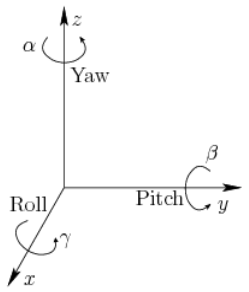


Figure 3: Human head modeled as a robotic arm [11].

In our second approach, we use image thresholding to identify the feature points. Our approach is based on methods developed by Stiefelhagen et al [6]. Here, we first run a Viola & Jones [15] based face detector to locate and extract the face of the subject. Next, we use geometric constraints to map out the regions where the various features are expected to be found. On each of these regions, we apply thresholding to the n-th percentile (where n is a parameter). The idea is that features being the darker parts of the human face, thresholding leaves only feature pixels behind in the image. Now we apply a centroid location technique on the pixels left behind to determine the feature point. An iterative version of the technique where the previous estimate of the feature point can be used to redefine the geometric constraints on the search space is being worked upon by us right now. One interesting observation we made while working on the technique was that the geometry of the human nose causes a shadow to be cast on one side. This causes the thresholding to err such that the detected feature point on the nose is always offset to the side in the shadows. We therefore also do an inverted thresholding on the nose to locate the brightness centroid to find a correction measure for this error. Unlike in the previous approach however, we have not incorporated a flow based tracking yet here, and so we redo all processing on each frame of the video stream received.

The next step was to build a 3D head model that can be used with the POSIT [4] algorithm. For this we use some anthropometric approximations [11]. We worked on the model of the human head akin to a robotic arm developed by Garcia-Mateos et al [10] and enhanced it with detail of nose and lips. We plan to use this with the POSIT algorithm to compute the rotation matrix and compute the yaw, pitch and roll angles from it. The computation of the three angles from the rotation matrix is given in the following figure.

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$



$$\alpha = \tan^{-1}(r_{21}/r_{11}),$$

$$\beta = \tan^{-1}\left(-r_{31}/\sqrt{r_{32}^2 + r_{33}^2}\right),$$

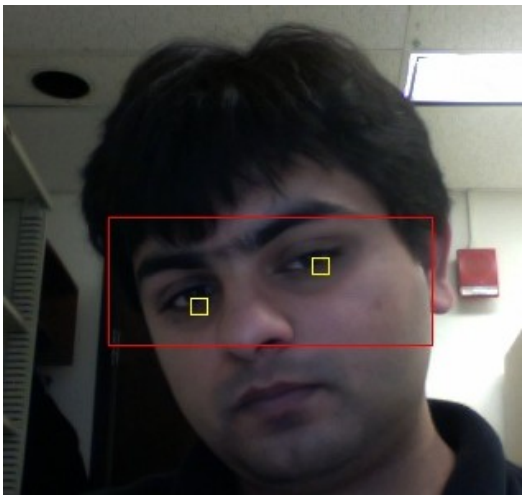
$$\gamma = \tan^{-1}(r_{32}/r_{33}).$$

Figure 4: Computation of Yaw, Pitch and Roll angles from the rotation matrix.

On the robot Wakamaru's end, we have a Java server running on it, which listens for commands transmitted to it over the wireless network. The roll, pitch and yaw angles computed can be sent to the server, and in response it moves the robot's head to the specified orientation.

7. RESULTS

First we developed the Robot's side of our work. We used a commercially available solution called FaceAPI [16] which returns the yaw, pitch and roll angles in degree measure. We captured this and sent it to the robot server we developed thereby creating a proof of concept prior to our implementation, of what we intend to do. We were able to move the robot's head in sync with a human's head successfully.



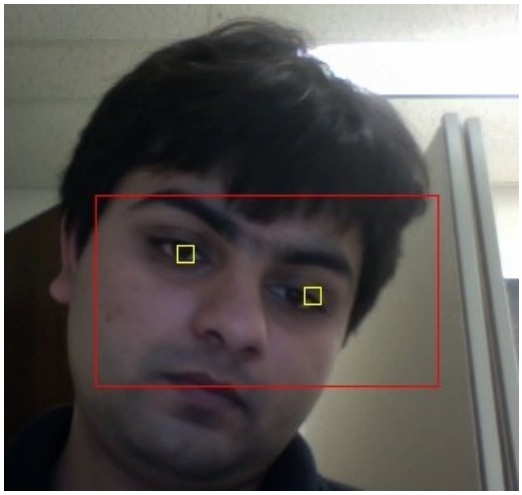


Figure 5: Results from using the Haar Classifier. The red rectangle highlights the search space and the yellow rectangles highlight the identified features.

With the Haar-classifier, OpenCV has a lot of infrastructure that we could make use of. However, locating a good training set was the difficulty that we faced. We managed to get hold of some good ones off the INTERNET and implemented our system with it. We were able to track the eyes successfully with this. However, tracking the nose and lip corners are still remaining.

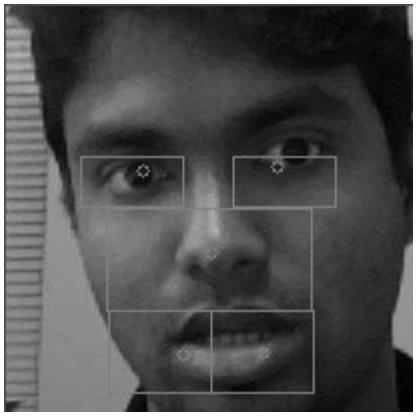


Figure 6: Result of image thresholding, the white rectangles show the geometric constraints used and the circles show the feature locations returned by our implementation

Figure 7: The thresholded image which shows the features along left dark and the rest of the face whited out.

For the thresholding based solution, we were able to implement it to track all features; eyes, lip corners and nose. We observe that there's a lot of error in the tracking of lip corners. Further study leads us to believe that it would be better to track the lips as a line rather than just two corners. We are working on this. Likewise, at the moment, our implementation only returns a single point for the nose. We are working on separating this into two points for the nostrils. And finally, we are working on an iterative solution to improve accuracy and reduce instability in the located feature points.

On the frame rate front, we observed that the Haar classifier based approach achieves a rather poor frame rate especially when face is partway turned away from the robot. On the other hand we were able to achieve rather good frame rates, upto 10-13 fps, at a resolution of 640x480 camera feed using the thresholding based solution. We strongly believe that further optimization for performance is possible.

8. ACKNOWLEDGMENTS

We are deeply indebted to Prof. Bilge Mutlu for providing us with the robot Wakamaru to work with, and assisting us in all the bottlenecks that we faced in the project. We also acknowledge the lavish support and guidance given to us by our Prof. Li Zhang both through and outside of the Vision course towards this project.

9. REFERENCES

[1] Murphy-Chutorian, E.; Trivedi, M.M. Head Pose Estimation in Computer Vision: A Survey. Pattern Analysis and Machine Intelligence, IEEE Transactions on Volume 31, Issue 4, April 2009 Page(s): 607 – 626.

[2] Mutlu, B., Shiwa, T., Kanda, T., Ishiguro, H., & Hagita, N. (2009). Footing in HumanRobot Conversations: How Robots Might Shape Participant Roles Using Gaze Cues. In Proceedings of the 4th ACM/IEEE Conference on HumanRobot Interaction (HRI'09), March 2009, San Diego, CA.

[3] Simon Baker, Iain Matthews, Jing Xiao, Ralph Gross, Takahiro Ishikawa, and Takeo Kanade, Real-Time NonRigid Driver Head Tracking for Driver Mental State Estimation, 2004.

[4] Daniel E. Dementhon, and Larry S. Davis, Model based object pose in 25 lines of code, International Journal of Computer Vision, 15, 123141 (1995)

[5] Gary R. Bradsky, Computer Vision Face Tracking For Use in a Perceptual User Interface.

[6] R Stiefelhagen, J Yang, A Waibel - A model-based gaze tracking system International Journal on Artificial Intelligence Tools, 1997

[7] A Gee, R Cipolla - Fast visual tracking by temporal consensus Image and Vision Computing, 1996 – Elsevier

[8] S. Niyogi. Example-based head tracking 2nd International Conference on Automatic Face and Gesture Recognition

[9] Niyogi, S. and Freeman, W. T. 1996. Example-based head tracking. In *Proceedings of the 2nd international conference on Automatic Face and Gesture Recognition (FG '96)* (October 14 - 16, 1996). FG. IEEE Computer Society, Washington, DC, 374.

[10] Gines Garcia-Mateos, Alberto Ruiz, Pedro E. Lopez-de-Teruel, Antonio L. Rodriguez, Lorenzo Fernandez. 2008. Estimating 3D facial pose in video with just three points. Computer vision and pattern recognition workshop.

[11] The Wikipedia entry on Head: http://en.wikipedia.org/wiki/Head

[12] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," Proceedings of the 1981 DARPA Imaging Understanding Workshop (pp. 121–130), 1981.

[13] The Wikipedia entry on Wakamaru: http://en.wikipedia.org/wiki/Wakamaru

[14] The OpenCV website http://opencv.willowgarage.com/

[15] Viola, P. and Jones, M. (2001). Robust real-time object detection. In CVPR.

[16] FaceAPI website: http://www.seeingmachines.com/product/faceapi/