Markerless Motion Tracking

Kelly Byron, Linh Pham, Joshua Situka
Mentors: Ken Kopecky M.S., Song Zhang, Ph.D.
Graduate Program of Human Computer Interaction
Virtual Reality Applications Center
Iowa State University

Abstract

Marker-based motion tracking is currently a popular method being used in both feature films and video games to create realistic movements and special effects. However, this technique requires physical markers to be placed on a human. Multiple cameras track the markers, which often cannot be done in real-time. Our work tracks a human hand without using any physical markers. This is done by using a skin detection algorithm to recognize the hand, a curve approximation algorithm to smooth the outline of the fingers, and trigonometric analysis to uniquely identify each finger. The user's hand is able to control a 3D hand model that copies its movements on the computer screen. This system uses a single camera, the ZCam developed by 3DV Systems. The ZCam is able to acquire the depth information as well as the 2D information in real-time. We realized finger tracking and mapped its motion to virtual 3D hand model in real time, albeit the motion was not realistically because only a few points are track for this study. We also created hand outline and found that more control could be realized the rotation around z axis could be up to 170 degrees. Because real hand motion was tracked, it can be used for numerous human computer interaction (HCI) applications. For example, we successfully applied finger tracking recognition to interact with 3-D water naturally and easily.

Introduction

Marker-based motion tracking is a system that uses physical markers that are placed on a human. Typically, multiple cameras are used to capture the movement of the markers, which is later used by a computer to generate a 3D model with the same movement as the body. This method has become commonly used in both feature films and video games to aid in creating realistic special effects. Recently, it has been used in several popular films such as The Polar Express, Beowulf, and Benjamin Button. It is also often used in video games to animate athletes and other characters in the game. As popular as this technology has become, it still has many disadvantages. The cameras only follow the motion of the markers, therefore the rest of the animation still needs to be done by a computer with enormous manual inputs, which can be a very time consuming process. This existing system also requires a large number of cameras and equipment, which makes it very expensive. Finally, many marker-based systems do not allow real time visualizing of the data, and since real life performance does not always translate to the 3D model as expected, this can result in problems that require an entire scene to be re-captured. As a solution to these problems, methods of markerless motion tracking have started to be developed. This would eliminate the need for physical markers as well as solve many of the other issues with marker-based tracking. However, few of these algorithms are able to accomplish finger detection since they are often limited in 3D motion.

Previous Work

Over the past few years, markerless motion tracking technologies have advanced drastically. Cheung et al developed a markerless motion tracking system by using shapes from silhouette images (Cheung et al. 2005). This technology uses human kinematic models that are used to track the motion of the person. However, their system does not work in real time. Near real-time hand recognition with an IR time-of-flight camera has been accomplished by Breuer et al. (2007). However, this system cannot track different movements of each individual finger. Segen and Kumar (1999) used a light source, a camera, and shadows, to achieve real time motion tracking of the hand. They were also able to recognize gestures with two fingers such as point, click, etc. However, shadow variations caused consistency issues due to the range of distance of the hand from the background. Soutschek et al. (2008) also tracked hand gestures for a user interface, which they did by using a single time-of-flight camera. The time-of-flight camera provides 3D image data, giving the depth as well as the color information at each point. This provided more reliable depth information, but only used recognized specific gestures with markerless tracking. A color-based method for tracking hand gestures was developed by Kang et al. (2008). They used a color detection algorithm to locate the hand and contour analysis to find the fingertip. This method is limited because it does not allow all five fingers to be located and differentiated. Holub and Nekolny (2009) developed a method combining skin and depth detection of the hand for American Sign Language recognition. However, their algorithm does not recognize non-trained finger tracking movement.

In this research, we are using a time-of-flight camera to achieve real-time markerless motion tracking of a human hand, as well as each individual finger's motions. First, motion tracking of markers on finger tips was used to train the 3D hand model. The markers are later removed when the mathematical algorithm was developed to separate finger tips from a blob of skin color. Next, calibration of color had to be made to distinguish markers and later skin color for increase robustness under different lighting conditions. Eventually, the markerless algorithm of tracking the hand was accomplished by using skin detection and a fast connected components algorithm to extract the hand from the background and remove background noise. A curve approximation algorithm is used to smooth the outline of the finger contours and trigonometric analysis is used to uniquely identify each finger. Finally the tracked markerless motion is then transferred to drive the 3-D hand model.

Methods

The goal of this project is to track 3D motion of a real hand to drive the virtual hand generated by a computer in real-time. Instead of using a physical device, such as a haptical device, we use a computer vision technology and an imaging device to detect the hand motion and interact with the computer. The time-of-flight camera used was the ZCam and is show in Figure 1, and its specifications are shown in Table 1.

Figure: **1 ZCam**

Resolution	320x240 at 30fps 160x120 at 60fps
Dimensions	85(W)x90(H)x60(D)mm
Depth Data Rate	30fps
Operating Range	0.5-2.5m
RGB Sensor format	1.3M
Min. Res Depth	2 cm, depends on range window.

Table 1: Specifications of ZCam

The ZCam works by sending IR pulses to its field of view, creating active illumination to get depth information from the reflected light by the objects in the field. Figure 2 illustrates an example on how the ZCam attains its depth information about the scene.

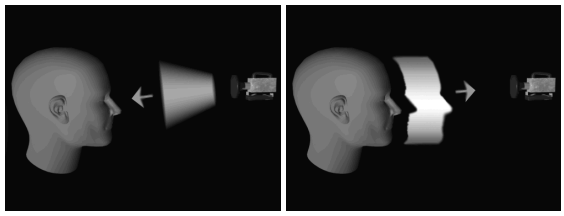


Figure 2: **Light being projected and reflected from camera (www.3dvsystems.com.il)**

There is a fast shutter which truncates returning light, in which the imaging sensor integrates more light from closer objects. It produces a black and white video, in which grey saturation represents the depth (G.J. Iddan and Yahav 2000). The closer the object is to the camera the lighter the grey. In Figure 3 below the depth information is in the bottom left corner of the four views. The hand is noticeably lighter than the head behind it.

Figure: 3 ZCam output

Skin and Marker Detection and Calibration

First we used markers on the fingers to track their motion and train a 3D model, before we transferred the 3D model to a markerless motion tracking algorithm. It was easier to train the model first with markers to distinguish finger tips then later take off the markers and use a mathematical algorithm developed to distinguish fingertips from skin to control the model. We capture the depth as well as the color information of a moving hand with the ZCam. OpenCV and skin detection techniques are used to find the markers attached to the finger tips in real time. Disparate color markers are attached onto the hand at the finger tips and palm. The markers that follow the motion of a hand are used to identify the 3D motion of the points. Holub et al. (2009) skin detection algorithm for sign language was incorporated to detect the marker and skin position from the 2D video images. We use this algorithm to get a color spectrum for each marker color that was used in developing the 3D model of the hand. The

spectrum is analyzed to determine if the pixel in a video image corresponds to a predetermined color we are attempting to find. In OpenCV, we represented a pixel color in the blue, green, and red (BGR) format, corresponding to blue, green, and red respectively. For processing each pixel in an image frame, a proportion of the colors were used (Holub et al 2009). Each color can have up to 255 in saturation value. The values were normalized to accommodate for different light conditions by taking the sum of the BGR intensity values for each pixel. This produced a new scale from 0-1 for B, G, R, but since it is a proportion, analysis of only two colors on a plot is necessary. An example of the calculation is below.

Example- For Red Marker Pigment

Color Pigment Value

Blue Value - 100

Green Value- 100

Red Value - 250

Blue + Green + Red = 450

Normalized Ratio in Color Spectrum

Blue Ratio - 100/450 = .222

Green Ratio - 100/450 = .222

Red Ratio - 250/450 = .556

Red and green were chosen arbitrarily, but any combination of two of blue, green, and red would produce the similar detection results, because the third color's ratio would be implied from the proportion. Figure 4 represents the red marker spectrum normalized in Matlab.

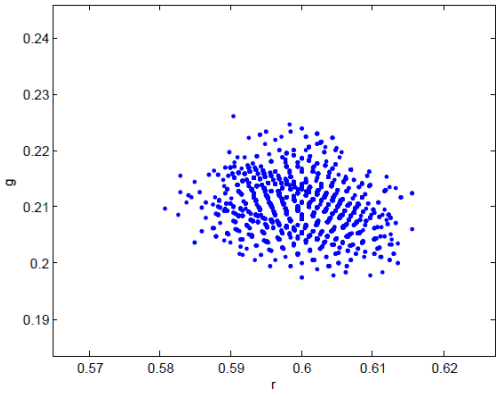
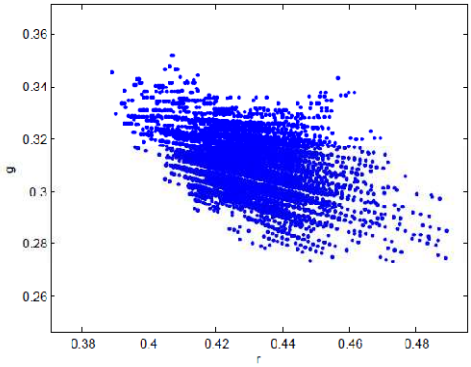


Figure 4: Red marker spectrum

Therefore, if the pixel's color properties fall in the red range from .58 to .62 and green range from .195 to .225 it will be determined to be red. Figure 5 demonstrates the normalized pixel color saturation for skin.

(a) Real skin image



(b) Color sepectrum

Figure 5: Skin with Matlab spectrum

As can be seen by Figure 5 skin pigments have a broader spectrum than red. We used six colors which created overlap issues when color spectrums were similar. To achieve better skin detection, a color calibration is required. We developed the calibration algorithm for the ZCam to better distinguish the colors from one another. Skin is considered its own special color and calibration helped separate the color from other noise in the environment. Also, since the accuracy of the Holub et al. (2009) method was not consistent if the light changes in a room condition, a calibration was needed to improve its performance. During the calibration, we converted the image from BGR color-space to the Hue Saturation and Value (HSV) color-space. We found that increasing raw saturation (S) value a little bit will increase the performance of the calibration, which was interesting. We believe that it is caused by the image sensor used in ZCam is not very good. After adjusting the S value, the color was converted to BGR color space for descriptive statistical analysis. We used descriptive statistics to analyze the pigment data for color calibration. Although the ZCam software comes with its own calibration, it does not specifically serve the purpose for us to distinguish colors. The calibration is based on finding outliers from a five number summary illustrated in Figure 6.

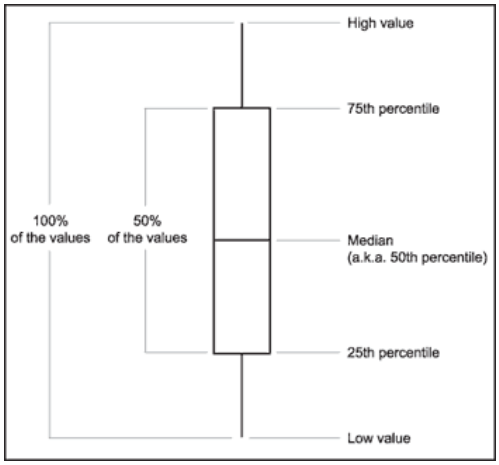


Figure: 6 Box Plot

This method is used because of its flexibility with a video sequence. We took the frame and color that the user wanted to calibrate and processed the frame with the user holding the indicated color or skin part to the camera. The pigments outside of the main color shown to the camera were ignored. This was accomplished by sorting the pixels to find the median, upper quartile (UQ top 75%), and the lower quartile (LQ lower 25%). The Inter-quartile range (IQR) was found by subtracting the upper from the lower quartile. The IQR was multiplied by the "stretch" and added to the upper quartile range and subtracted by the lower quartile range to get the bounds or whiskers. Values outside those bounds were outliers, and will not be considered in forming the detection range for each color. The "stretch" is traditionally 1.5.

This was proved to be an improvement over the Holub et al. (2009) method because of the flexibility in changing the stretch to accommodate different lighting conditions.

Markerless Finger Tracking Algorithm

To implement marker-less motion tracking of a hand, we had to apply image processing methods to improve the quality of the image and mathematical algorithms to find the finger tips. We follow the following steps to find the finger tip motion:

Step 1: First an image was obtained through the camera, cvThreshold() was used with a depth image to eliminate the background. Because we assume that the hand is separate from the body in depth, the depth information with OpenCV segmentation algorithm separates the hand from the result of the body, which makes future hand tracking easier.

Step 2: Skin tracking was used to grab the hand. We grabbed an image, using the values calibrated earlier. Then we applied the "Fast Connected Components" filter. This algorithm goes through an image, and associates "blobs" based on color. For example, it would be able to differentiate between a blue blob and a red blob, isolating the two. The purpose of this filter was to get a better image of the hand, distinguishing it from background noises.

Step 3: We compared the skin image to the depth image using cvAnd(), we were able to isolate the hand. We applied a Gaussian smoothing filter (cvSmooth()) to further filter out the noise. Figure 7 shows the result before and after applying Gaussian filter. It caleraly shows the improvement of the using the filter.

Figure 7: Before and After Gaussian Blur

At this point we had a clean image of the hand. However, when we took the contour of our image with cvFindContours() to find the contour of the hand and the fingers. Finger tip detection method relied on the curvature properties of the contours.

Step 4: In order to simplify the problem, we used the Douglas Peuker Approximation with the function cvApproxPoly() to reduce the amount of points in our contour. This function works by picking the farthest vertices from each other to form a line. The next point picked is the point farthest from the line, while staying within the bounds of the precision parameter of the function. The third point picked is the point farthest from this new line formed. This process is repeated until there are no more points that can be picked, or there are no points that are within the bounds of the precision parameter. The end result was a contour consisting of around 10 points, as opposed to 200 in Figure 8.

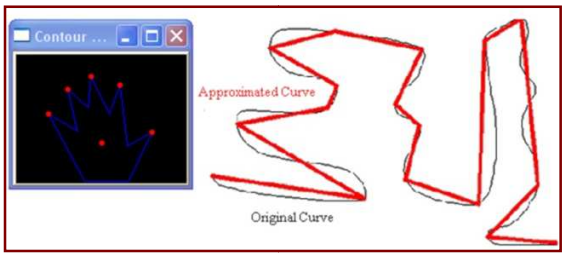


Figure 8: Douglas Peuker Approximation

Step 5: To detect fingers, the function atan2() was used to subtract angles between 2 adjacent points. If the difference was positive, it was a finger. An example is shown in figure 9.

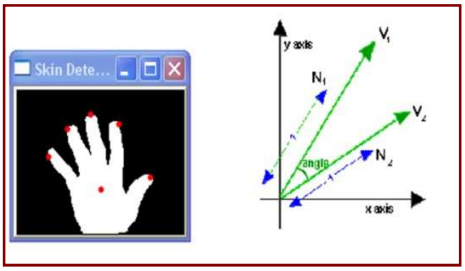


Figure 9: Trigonometry

Step 6: However, to distinguish the fingers from each other, we needed to find the palm. This was done by taking all the points except the fingers, and averaging them. This gave us the center of the hand. By comparing where each finger was relative to the palm, each finger could be differentiated from each other, resulting in marker-less motion tracking of a hand.

Step 7: To increase realism and detection robustness, the time series of each point is stored. Then the previous four points from past finger movement are averaged. Therefore, if in the next frame the finger tip is not detected there will still be a predicted location based on the other previous points. Also, this makes the movement from of the finger tip from one point to the next much smoother.

3D Hand Model

We created the 3D hand model that is controlled by the human hand using OpenGL. It is made out of basic spheres and cylinders in OpenGL. These shapes are then rotated and translated based on the position of the human hand in front of the camera. The previous and current positions of each finger on the human hand in the x, y, and z directions are kept track of in the code. This information is then used to find the difference between the current and previous position in each direction. The arctangent of the x and y differences for each finger is then used to control how much each finger rotates in the x direction. The difference in the y direction in each finger is also used to translate each joint when a finger bends. The difference in the z direction for each finger is used to control the rotation of how far each finger bends forward or back. The result is illustrated in Figure 10.

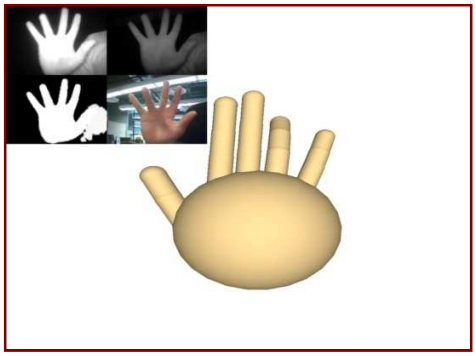


Figure 10: 3D Model

The differences in current and previous position are also used to translate the joint of each finger when it bends. For the mark on the center of the palm of the hand, the differences in the x, y, and z directions are used to control where the hand model should be in the display window and how it is moving within the display window. Although we were able to move the fingers, the bending was inconsistent and unstable with our hand. The finger detection algorithm we used only produced somewhat natural results in combination with mapping of the virtual hand model. One of the problems was it is only able to find a fingertip if the approximation had a sharp corner in the contour. If the situation occurred where a finger was put down, it would not be able to track it anymore. If we could have built a better mathematical model to compensate for this limitation, it would have produced better results.

We created a second hand model outline that had a more control stability. Also, that hand model was able to rotate. This model was created by taking using OpenGL 3D shapes to follow the outline of the hand. The model is illustrated in Figure 11 and 12.

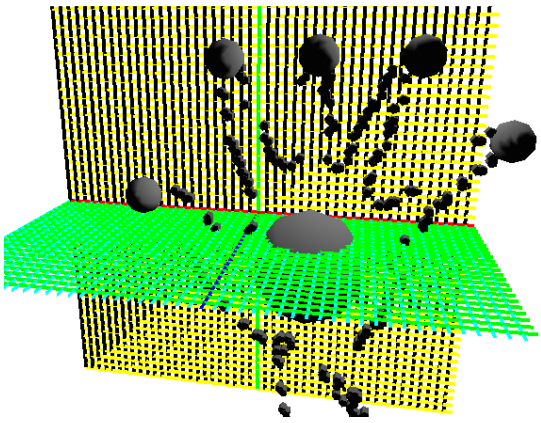


Figure 11: Hand model outline

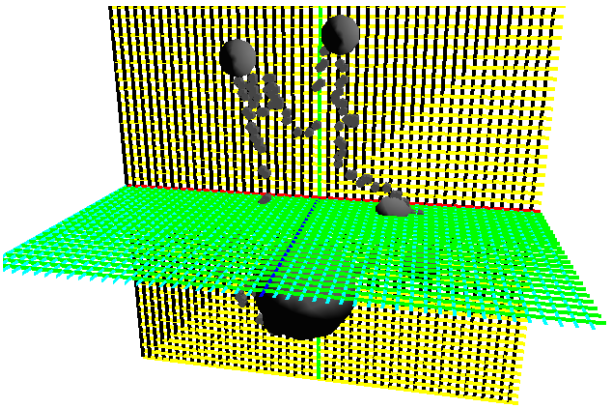


Figure 12: Hand model outline motion tracking two fingers

The response on the hand model outline was much more accurate, which is what is demonstrated with its fluid reaction to change to two fingers.

Applications

Because we were able to track finger tips without markers, we applied this concept to manipulate 3-D geometries, such as water. With this water model, a function would be called if we wanted the water to move. This function would be called if one of the fingertips was at a certain depth, and the water would appear to be touched at the appropriate point as shown in Figure 11 and 12.

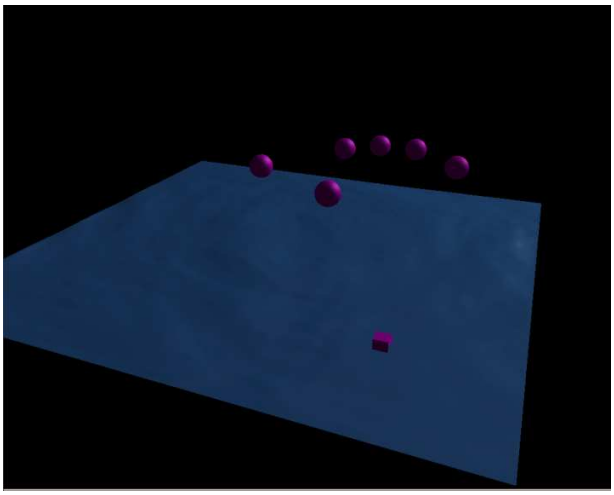


Figure 11: Fingers not touching water

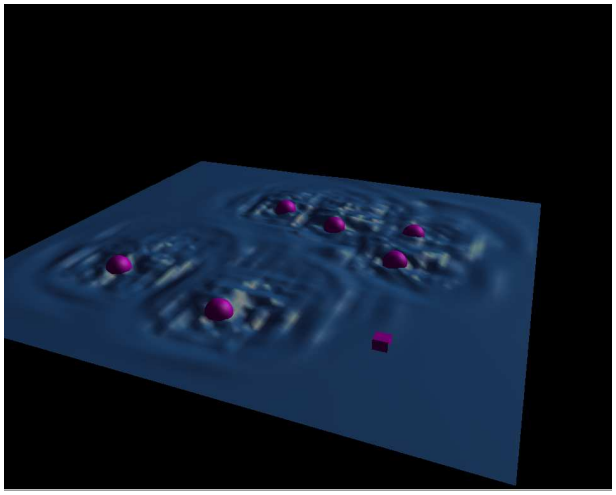


Figure 12: Fingers Touching Water

Results

For Finger Tracking, we were successfully able to differentiate fingers from each other, although the current algorithm only follows a limited range of motion. The fingers going off-screen and full rotation of the fingers into a fist was not accomplished to a point of realism. Crossing the fingers also created an inaccurate motion tracking. Having the fingers close together with no space in between, made it hard for the model to distinguish fingers. The model best followed the hand with the palm facing the camera. Each finger was able to bend but not consistently to get accurate measurement. However, once finger tips out of view of camera then control of the model was lost. The second hand model masks lost control of finger tips because of the 3D shapes. Appearance is more fluid than the first model because of the ability to move in 3D space accurately. The second hand model could make up to a 170 degree rotation turn around the z-axis (If z-axis defined as going from the ground to the ceiling). The depth response of the ZCam was a little unstable. The response can sometimes differ from day to the next even in the same location and daylight time. Calibration was accurate between 15cm-20cm. If color was held to close to camera distortion from infrared feedback can give inaccurate results. The project focus was on motion tracking, therefore, the first 3D model was very simplistic in nature using 29 polygons to make analysis of successful tracking easier, decreasing realism of finger motion to the eye. The motion tracking of the hand worked best between .5-1.5 meters away from the camera.

Conclusion

Even with these limitations, our method is still an improvement of finger tracking in real time. The depth instability made off screen tracking of fingers difficult but manageable with proper filtering techniques. This simple

3D model created in this research could be improved to make a more realistic hand. In the future a mathematical solution can be developed to account for the full rotation of the hand. Furthermore, gesture recognition could be significantly increased with this technology. Because this technology is noninvasive, video game and movie applications can possibly used this method for interactive entertainment.

Acknowledgements

This research was performed at Iowa State University as part of a research internship sponsored by the Program for Women in Science and Engineering and the Department of Human Computer Interaction during Summer 2009.

References

[1] P. Breuer, C. Eckes, and S. Müller, "Hand gesture recognition with a novel IR time-of-flight range camera—a pilot study." A. Gagalowicz and W. Philips (Eds.): Lecture Notes in Computer Science, Springer: Berlin, vol. 4418, pp 247-260, 2007.

[2] K. Cheung, S. Baker, and T. Kanade, "Shape-from-silhouette across time part I: Theory and algorithms." *International Journal on Computer Vision*, vol. 62 issue 3, pp 221–247, 2005.

[3] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a line or its caricature." *The Canadian Cartographer*, vol. 10 issue 2, pp 112-122, December 1973.

[4] J. Holub, B. Nekolny, and M. V. Waardhuizen, "Application for Recognition of American Sign Language using the ZCam." Department of Human Computer Interaction Technical Paper, Iowa State University, 2009.

[5]G.J. Iddan & G. Yahav "3D Imaging in the studio" 3DV Systems Ltd. www.3dvsystems.com.il, 2000.

[6] S. K. Kang, M. Y. Nam, and P. K. Rhee, "Color Based Hand and Finger Detection Technology for User Interaction." *International Conference on Convergence and Hybrid Information Technology*, pp 229-236, August 2008.

[7] J. Segen and S. Kumar, "Shadow gestures: 3D hand pose estimation using a single camera." *Proceedings of IEEE Conference on Computer Vision and Pattern s*, vol. 1, pp 479-485, 1999.

[8] S. Soutschek, J. Penne, J. Hornegger, and J. Kornhuber, "3-D Gesture-Based Scene Navigation in Medical Imaging Application Using Time-Of-Flight Cameras." *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp 1-6, June 2008.