Wes Miller
5/11/2011
Comp Sci 534

Expression Detection in Video

Abstract

Expression detection is useful as a non-invasive method of lie detection and behavior prediction, as many facial expressions are involuntary. However, these facial expressions may be difficult to detect to the untrained eye. This program implements facial expression in video using optical strain as described by Godavarthy[1], and returns frames in which an expression is present, along with a processed image of the face showing the location(s) of strain, so that the expression can be more readily identified.

Introduction

Traditional lie detection techniques have several shortcomings. They can be invasive, requiring various biometric measurements such as heart rate to determine whether a suspect is telling the truth. Additionally, these biometric tests can be unreliable, as they can be manipulated by the suspect. Expression detection avoids both of these shortcomings by using a video camera to passively record a suspect's face. Certain types of expressions are involuntary and difficult to consciously mask, providing a reliable metric for determining intent to deceive.

The detection of involuntary facial expressions can also be used to infer behavior based on emotions. This use is particularly appealing for areas where security is critical, as suspected criminals could be identified through surveillance footage, alleviating the reliance on highly invasive searches.

Motivation

I was first introduced to the application of facial expressions in lie detection a couple years ago by the television series Lie to Me, based on the research of Dr. Paul Ekman[2]. Although the show features plenty of computers used for video capture and

playback, the expression detection is left entirely to the characters. I thought that attempting to detect expressions programmatically would be interesting and could also potentially be helpful for those who aren't trained at recognizing them in real life.

Problem Statement

Although facial expressions are interesting by themselves, they are not particularly useful without additional context. A person showing contempt at an airport could be a potential criminal, or simply could be unappreciative of the airport security process they had to endure before nearly missing their flight. In an interview, the appearance of certain expressions might indicate a lie, but not why the suspect is lying. Therefore, the goal of this program is to not only detect the presence of expressions within a video, but to pinpoint their locations so that the additional context required to interpret each expression's meaning can be obtained.

Related Work

Facial expression detection borrows from many of the concepts used in face detection and recognition. In both cases, a face must be identified by comparing parts of an image to what a face is known to look like. Expression recognition also relies on tracking differences between multiple images of the same person to determine whether their facial expression has changed.

In terms of facial expression recognition, two algorithms have previously been used. The first is an extension of EigenFaces, which uses a low-dimensional "Face Space" to compare a person's current expression to various known expressions[3]. This algorithm is relatively simple, but requires prior knowledge of a particular individual's facial expressions, making it less suitable for use in surveillance or interrogation. A second

algorithm uses optical strain to detect motion in the subject's face, which requires only a baseline, or neutral frame of an individual to detect changes in expression[1]. This project implements the second approach.

Theory

A facial expression is displayed as a deformation of the surface of the face for a period of time. Macro-expressions typically last .75-2 seconds and can appear in several regions of the face, while micro-expression are more localized and last .04-.2 seconds[1]. To detect a deformation, there must be an un-deformed version to compare against. Optical strain is calculated as the change in optical flow over small horizontal and vertical distances, where optical flow is the change in position between two images, or in this case, two frames in a video sequence. However, optical flow in video is caused not only by facial muscles, but also by any motion of the entire face caused by the subject or camera moving. Therefore, the following method, outlined in the paper Microexpression spotting in video using optical strain[1], is proposed:

1. **Face detection** – Locate the face in a neutral video frame; crop surrounding region
2. **Baseline alignment** – Align subsequent frames so that the angle of the line connecting the subject's eyes is collinear; crop to size of first face
3. **Optical flow** – Calculate optical flow between neutral frame and subsequent frames
4. **Optical strain** – Calculate optical strain magnitude based on optical flow
5. **Expression detection** – Find peak strain values in each facial region; flag frames

Method

This method assumes that the following information is provided by the user: a video file, the location of a video frame within that file of the subject displaying no facial

expressions (neutral frame), and the location of a series of frames within the file to be compared against the neutral frame for the presence of expressions (subsequent frames).

Face detection

Face detection is performed using the Viola-Jones face detector, a relatively fast algorithm. However, because the algorithm might not provide consistent results across multiple video frames, the face is only detected in the neutral frame. In subsequent frames, the face is aligned based on the position of the subject's eyes, and cropped to match the size of the neutral frame. This has the advantage of reliably aligning frames for subjects with little head motion, but fails when the size of the face changes from one frame to the next, or the face rotates significantly. The frame is then cropped to the region detected by the Viola-Jones algorithm.

Next, the eyes are identified. This process is performed manually on the neutral frame to ensure accurate results. Once a right eye region and left eye region are selected, a Haar classifier is used to detect the eyes in subsequent frames. The centroids of the eyes (e_1 and e_2) detected are extracted and a line segment connecting them is drawn. The angle of this line (θ_0) and its midpoint (x_0, y_0) are calculated, and serve as alignment parameters for the subsequent images.

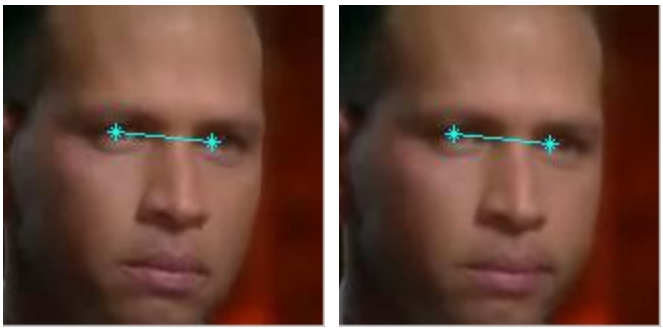
$$\theta_0 = \tan^{-1}\left(\frac{dy}{dx}\right)$$

$$(x_0, y_0) = \left(\frac{e_{x1} + e_{x2}}{2}, \frac{e_{y1} + e_{y2}}{2}\right)$$

Finally, eight regions of the face are selected manually for expression detection: the forehead, either side of the eyes, cheeks, and mouth, and the chin. The eyes, nose and mouth are not included in the detection area, since the eye area is susceptible to blinking, the mouth moves while talking, and the nose is typically rigid[1]. Once the regions are drawn onto the neutral image, they are assumed to be located in the same place on subsequent frames (after they have been aligned). Optical strain calculated outside of these eight regions will be ignored to reduce noise from changes in the background and other undesirable factors.

Baseline alignment

Each subsequent frame is processed in the following manner. First, the centroids of the eyes are detected using the Haar classifier as described in the previous step. If fewer than two eyes are detected, the frame is rejected. The angle and midpoint of the line connecting the eyes are calculated, then the image is rotated and translated such that $\theta = \theta_0$ and $(x, y) = (x_0, y_0)$. Finally, the image is cropped to the size of the neutral image.



Neutral frame Subsequent frame

Optical flow

Optical flow is calculated using the Black and Anandan robust dense optical flow algorithm[4]. After testing several different algorithms implemented in MATLAB, it was determined that this implementation provided accurate results, and did so faster than many of the inferior algorithms. The optical flow is computed between the neutral frame and every subsequent frame to determine how far the current face deviates from a neutral expression. Attempts to calculate the incremental optical flow between non-neutral frames produced far inferior results.

From the optical flow analysis alone, it is possible to detect the presence of expressions. However, optical flow also detects slight variations in the overall alignment of the image, due to inaccuracies in the face detection and baseline alignment steps above. This might be improved in the future by identifying more than two points to be used for image alignment, enabling a more types of 2D transformations. Unfortunately, the use of an algorithm such as RANSAC along with SIFT descriptors is not applicable since not all points between the two images can be considered stationary.

Optical strain

Optical strain is computed by applying a Sobel filter to the optical flow data for a given frame. A horizontal filter S_x is applied to the x component of the optical flow data, and a corresponding vertical filter is applied to the y component.

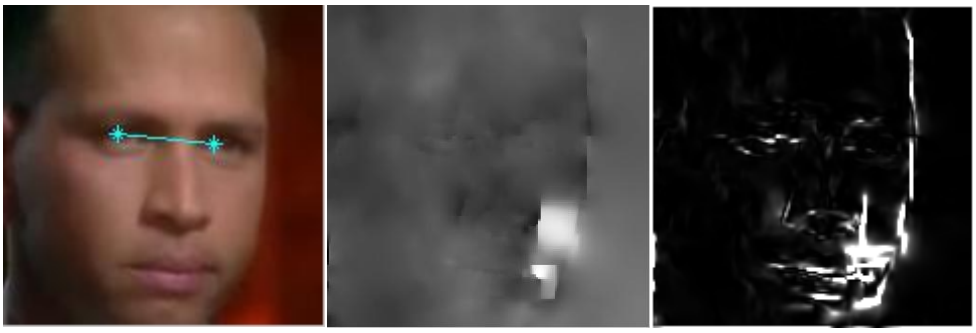
$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Shear strain (calculated by applying the vertical filter to the x component of the optical flow and vice versa) was also calculated, but did not provide reliable results and was therefore omitted from the algorithm.

To emphasize sharp edges in the optical flow, which indicate the presence of strain, each component of the resulting optical strain (S_u and S_v) is squared, before being summed to determine the overall optical strain (S) at a given pixel between the neutral frame and a given subsequent frame.

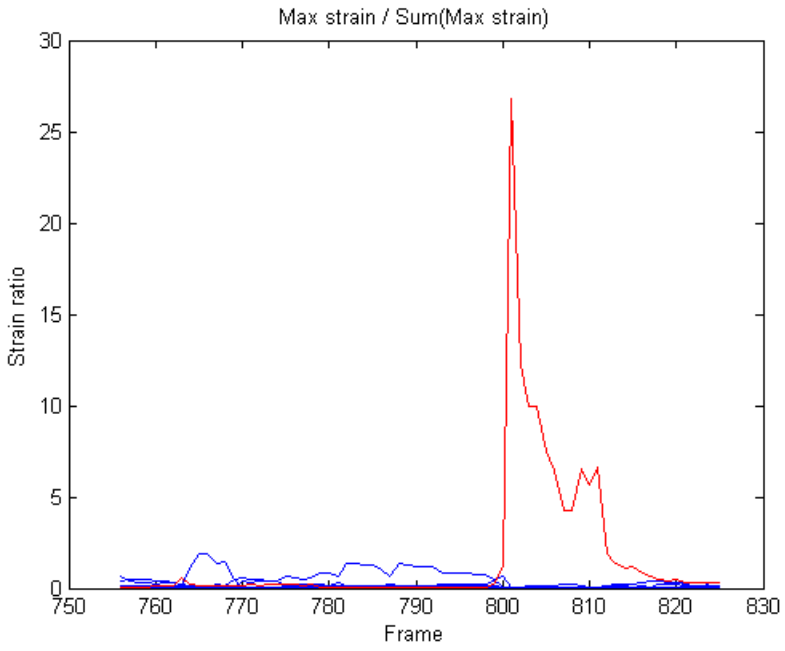
$$S = S_u^2 + S_v^2$$



Expression detection

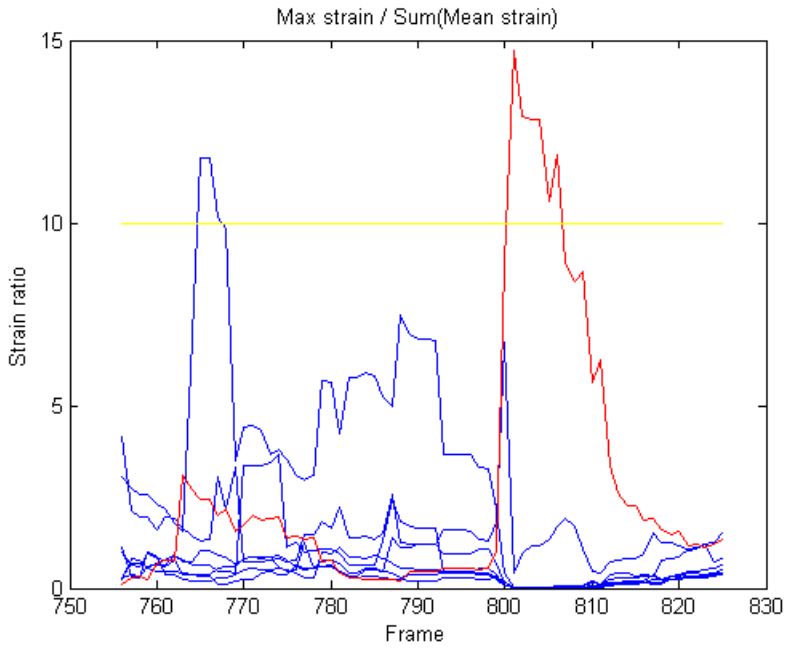
The maximum and mean strain values are calculated for each of the eight regions defined in the first step. Then, in order to determine whether a specific region contains an expression, two criteria are tested. First, the maximum strain value in the section must be c times larger than the sum of the maximum strain values in all other regions. This limits expression detection to micro-expressions, which are more localized than macro-expressions and also harder to detect.

$$\frac{\max(S_r)}{c} > \sum_{i=1}^{8}(\max(S_i)) - \max(S_r)$$

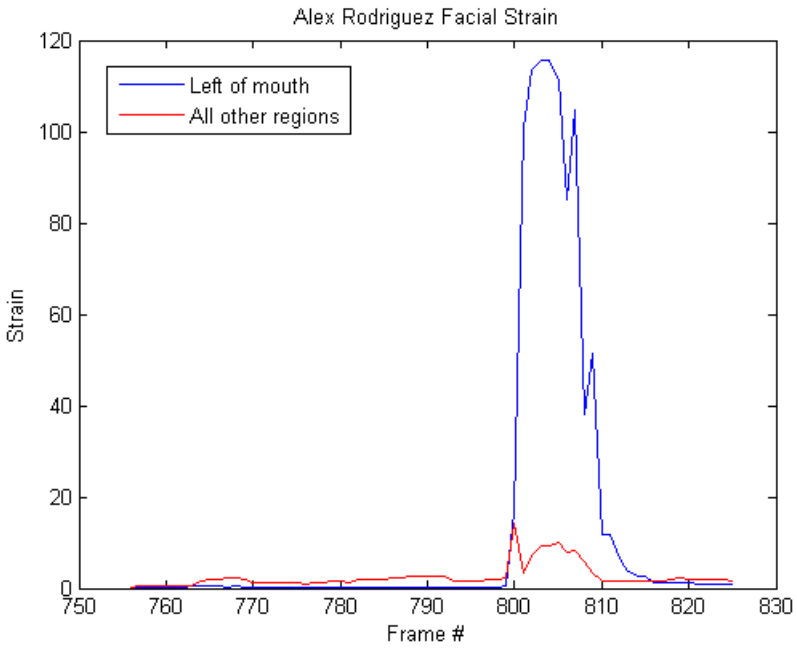


The second criterion is that the maximum strain value in a given region must be d times larger than the sum of the average strain of each of the regions. This isolates large differences in strain while ignoring cases where large amounts of strain are evenly distributed over the frame.

$$\frac{\max(S_r)}{d} > \sum_{i=1}^{8}(\text{mean}(S_i))$$



Note that both of these criteria are independent of the frame size, since calculate strain ratios. The actual values of strain are dependent on the number of pixels the face occupies, since face distortions in an HD video would occur over more pixels than the same distortions recorded on an SD camera, assuming all other camera parameters are the same.



Although this is sufficient for detecting expressions, since these strain values might persist over multiple frames, a simple algorithm was implemented to limit the number of frames returned in response to a single expression. The previous frame is remembered, and the maximum strain value over all regions is compared to the current maximum. Only when the current maximum is smaller, the previous frame satisfied the two requirements described above, and the fifth previous frame did not satisfy the requirements, is the previous frame flagged as an expression. The previous frame, fitting those criteria, is a local maximum, and expression is not too long to be a micro-expression. (This test can be adapted for macro-expressions.)

Experimental Results

Due to the algorithm's requirement of consistent face rotation and size, most of the time was spent tailoring the algorithm to work on a particular image sequence of a 60 Minutes interview with Alex Rodriguez. In that scenario, the algorithm picked out a micro-expression and rejected all frames without micro-expressions. Initial testing with other

video files indicated that modifying several parameters would be necessary; however, I am looking forward to testing this algorithm on further video files, and results will be posted to my web page (http://pages.cs.wisc.edu/~wmiller/facexprs/) when possible.

Concluding Remarks

At this point, the algorithm is able to detect facial expressions in very limited scenarios. This could be improved in the future by improving the algorithm's robustness with regard to out-of-plane motion. Additionally, in order to replace traditional security practices at airports, the algorithm must be able to run in real-time, which it does not do at this point. Ideally, the program could also be expanded to identify specific facial expressions and process context as well for a fully automated lie detection solution.

References

[1] Godavarthy, Sridhar, "Microexpression spotting in video using optical strain" (2010). Theses and Dissertations. Paper 1642. http://scholarcommons.usf.edu/etd/1642

[2] http://www.paulekman.com/about-ekman/

[3] M. Turk and A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, March 1991

[4] Secrets of optical flow estimation and their principles. Sun, D., Roth, S., and Black, M. J., IEEE Conf. on Computer Vision and Pattern Recog., CVPR, June 2010.