Project Report

cs5090: Virtual Environments with Dr. Scott Kuhl

Ruimin Zhang and Harriet King

April 2011

Abstract: “Show Your Face”, is a game played by face tracking with a webcam. The game makes moves on a screen displayed "game board" based on face position and size in the frame. This project requires a webcam and openCV. Expression recognition has been trained for still images with limited success and the trained cascade was not incorporated into the game.

Table of Contents

Original Intention

Our intention was to explore using the openCV library and see how easy it was to use. OpenCV has many image functions and our project was to use a webcam and create a face tracking and expression recognition program that would be a game for the user to play. If we had time, we were going to do 3-d anaglyph of a still image chosen by the user while playing the game.

Overview

We succeeded in creating a game using a webcam programmed for face tracking using the OpenCV library in Visual C++. We were accepted to give a poster at Michigan Celebration of Women in Computer Science (MICWIC) and while at MICWIC, we demonstrated the game which gave us a chance to see the face tracking in action with over 40 users.

We also succeeded in training Adaboost to recognize basic facial expressions on still images. This required preparing a database of images for positive and negative training samples The algorithm recognizes still images only and is separate from the webcam face tracking game, but we did succeed in incorporating the expression recognition for anger, happy, and surprise with the webcam capturing the user. See a video here: http://www.youtube.com/watch?v=UKC15jKmiKY.

We did not do any of the extra time contingencies regarding anaglyph 3-d as discussed in the original proposal.

Details of the Webcam Face Tracking Game

Introduction

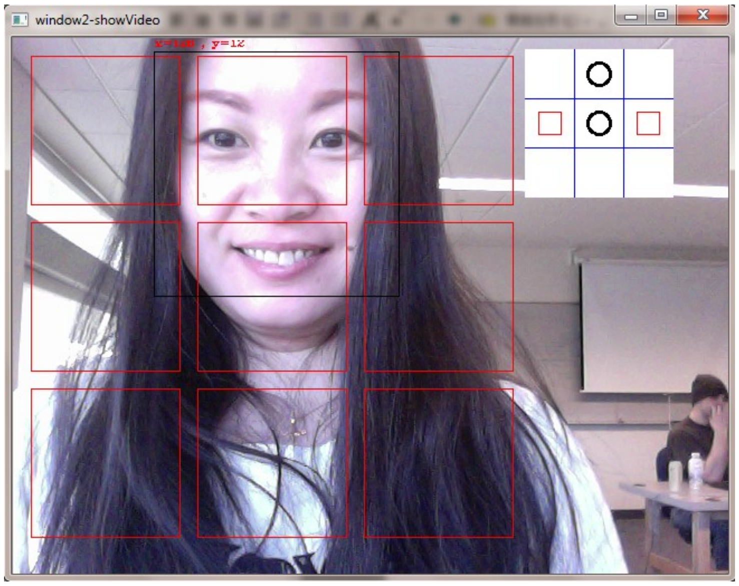
In this project, we used the OpenCV library with a webcam to create a game where the player can use his or her face position to play a Tic-Tac-Toe game with the computer. The user's display includes the webcam display frame and inside that to the upper right is a frame displaying the game board. The user can mimic into the webcam one of the face positions by surrounding a guide square with their face tracking square. If they are successful then the game will mark that square with the player's designated color. Please see example game image below.

OpenCV provided a cascade of classifiers for face recognition as an XML file that could be used directly in the program, then functions were available to track the face in the webcam images. Programming included displaying the live video from the webcam, creating the game interface including the game board with markers, the guide squares and face tracking square, and creating an algorithm for computer play against the human player.

Significance

Webcams are inexpensive and commonly found on laptops and can be used for users' tracking such as the 3 degrees of freedom face tracking we implemented in our game. Using webcams and object detection techniques makes it possible to implement a simple and affordable tracking system.

We were motivated to try the OpenCV library and see how quickly and accurately webcam face tracking could be implemented. Please see our "Future Work" section for more about our ideas.



Example of Game Play: Center Upper Square

Game Play Rules

- Uses standard Tic-Tac-Toe rules
- User plays against computer
- Watch play on game board (upper right)
- Click ‘S’ key to begin
- Move face in webcam to move face square
- Surround a game square with face square
- Click the ‘C’ key to select square
- Game board will update with black circle
- Computer will counter move with red squares

Technology of Face Tracking Game

Face tracking determines the location and size of human faces in an image or a video. Face tracking often uses machine learning (ML) algorithms to train the machine to recognize human faces. The most popular ML algorithm for face-tracking is AdaBoost which was proposed by Viola and Jones in 2001.

OpenCV is an open source computer vision library providing 500 functions spanning many areas in vision, such as camera calibration, stereo vision, robotics, and user interface.

We used a cascade of classifiers provided by OpenCV trained by Adaboost for face detection.

Observations from Demonstration

For the demonstration, we set up the game on a chest height table using the laptop to play. This allowed users to adjust the screen tilt to best accommodate their height so the webcam could capture their face. Over 40 users tried the game. We also previewed the game demonstration in class and got valuable feedback.

See example image above to understand the basic game play. The user's black square indicates the face tracking and there are red game play guide squares. The user is lining up her tracking square around the upper center guide square. Then when she pushes the "c" key she can "capture" that square on the game board, shown at upper right. In fact, she has already "captured" that square as evidenced by the black circle in that square on the game board.

Feedback from the preview session that we incorporated before the demonstration at MICWIC included:

1. improve the computer's game algorithm to provide a better opponent for user
2. avoid confusion by having the tracking pick up only one face
3. clarify the guide and tracking squares by using different colors
4. increase the clarity of the game board by adding background color to distinguish it from the webcam image
5. give different colors to circle and square markers
6. increase the margins from the edge to the guide squares to make it easier to surround the guide squares with the user's tracking square.

During the demo at MICWIC, attendees passing by were invited to play the game. A two sentence explanation was given to introduce the game play so users knew how to line up their tracked image around the guide squares. Most players succeeded in completing a full game in less than 90 seconds. One player with dark skin was only sporadically recognized by the face tracking algorithm but other players with dark skin were recognized.

Users had a variety of hair and head coverings including a Muslim with her face surrounded by fabric but the algorithm did recognize her face. Users often turned their faces when they first began playing but quickly realized they needed to stay facing the webcam centered at the top of the laptop's screen to have the game recognize their face.

Also, users came very close to the camera and found their face was too large to be recognized. They needed to keep a little bit of distance. Too much distance made their tracking square too small to surround the guide squares. More than one user suggested making the game recognize when the face tracking square was completely inside the guide square instead of surrounding the guide square.

Details of the Expression Recognition Algorithm

Introduction

Unlike the face tracking algorithm, there were no OpenCV classifiers ready to use for facial expression recognition. We chose to use the training function provided by OpenCV to train Adaboost with haar-like features. There are not many resources for facial expression training. Human facial expressions are very complex, so we intended to train for just the six basic expressions: anger, surprise, disgust, fear, sad, and joy.

After overcoming many obstacles, including OpenCV not including the necessary "_Cvhaartraining.cpp" file in their downloads, and preparing enough images to effectively train, we have succeeded in training for anger, happy, and surprise. A basic effective cascade needed to be trained at least 30 hours with more than 7000 positive training samples and 3000 negative samples.

Background Regarding Facial Expressions

Expression recognition is a complex endeavor because expressions are highly individualistic and subtly convey a blend of emotions. Although somewhat cross cultural, expressions do vary across demographics. The most obvious expressions, even in other species, are anger and extreme contentment and they seem disjoint from each other. But even with someone you know very well, disgust and fear can be difficult to distinguish.

If one is looking for just the "basic" six: sad, angry, surprised, disgust, afraid, and happy, there are many fuzzy combinations. Are we interested in happy surprise? Scared surprise? Both? Should we include sad afraid with afraid?

Guillaume Duchenne, Facial Expression Research, 1862

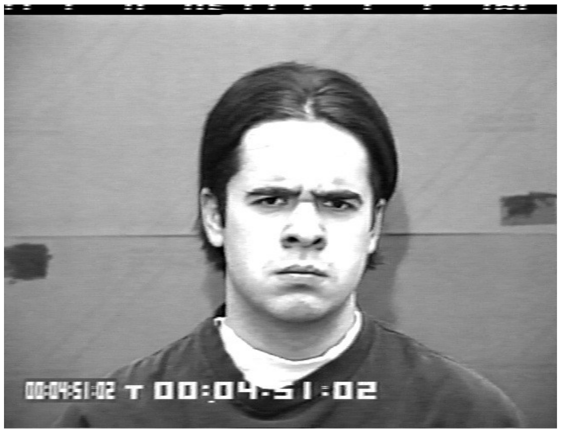
The facial muscles are directly controlled by the nervous system and facial expressions happen instantaneously. Scientists argue that expressions happen faster than one's brain can react, including microexpressions that are impossible to control and appear for as short a 1/25th of a second.

Facial expressions have been the subject of scientific study as early as 1862 when Guillaume Duchenne published his research in *Mécanisme de la Physionomie Humaine* studying which muscles produced certain expressions using electrodes. In the last 50 years there has been extensive research into facial expressions but some argue that this is a waste of time since facial expression is so individualistic [Daniel Laberge].

It has been argued that there are thousands of subtle facial expressions and that if humans did not have language, we would be much better at reading expressions, like our primate cousins who have elaborate communication via facial expression [Daniel Laberge].

Besides the cultural and individualistic issues and the blurring of a variety of emotions into a single expression, there are also concerns about the source of the expression images. We started with a research database of images called the Cohn-Kanade AU-Coded Facial Expression Database. This database was created for research purposes and includes basic expressions from test subjects.

But having a test subject emote a clearly defined expression is not necessarily effective. Compare the images that follow of a test subject's version of anger to an actor's expression of anger, or even a small child's anger which cannot be manufactured. That small child just feels angry and happened to be captured in that moment.



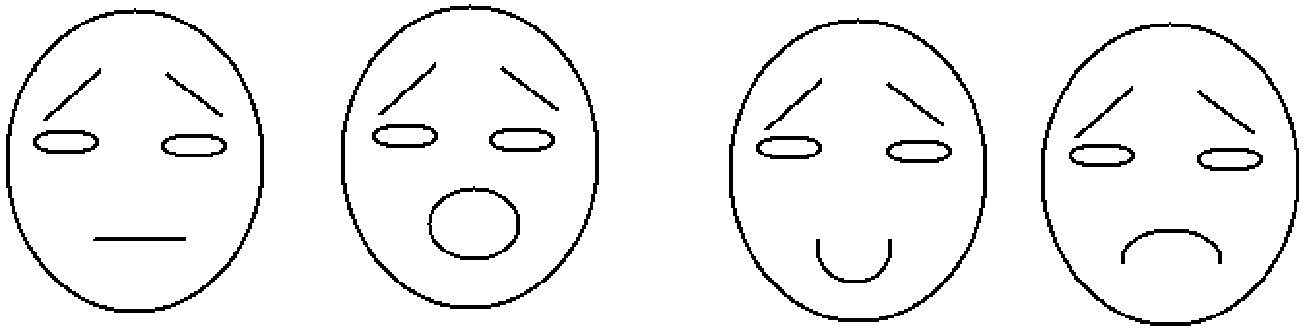
Database Test Subject: Anger

Actor: Anger

Small Child: Natural Anger

Technology of Expression Recognition

The haar-like features describes the ratios of pixel intensities of sub-regions of a rectangular region in an image. It scans the image many times, specified by the number of stages, with increasing size of scan window. If an area is scanned as a target expression multiple times, it will confirm that expression.


Sample Expression Diagrams

Expression recognition requires the algorithm to break down the features of the face and then be trained in what combination of features, eyebrows, eyes, mouth, etc., are regarded as a certain expression. Angle of eyebrows for example can be "read" as identifiers of separate expressions. Please see Appendix for "Sample Expression Diagrams".

An effective cascade is trained by iterating through the training images for many stages and this takes at least 30 hours.

The images need to be prepared for training by manually categorizing the image as a positive or negative example, noting the location coordinates of the face in the image and the face's width and height and then combining the information into a DAT file. The images are all converted into one single VEC file and fed for training.

There are two ways to use haar-like features to analyze facial expressions. One way is to analyze a still image, the other way is to analyze the haar-like features hidden in dynamic images which is a sequence of images describing one facial expression movement. For our experiments we directly applied haar-like features to still images.

One mathematical way to consider the possibilities of expressions is to consider the 6 basic expressions as on or off, then it is 2 to the 6th possible combinations, which is 64 expressions. But psychologists would argue that there are hundreds or even infinite subtle expression variations across individual differences.

Preparing the Image Database and Training Results

The Cohn-Kanade image database contained too few images to effectively train the algorithm. Solutions were found to run programs to slightly distort each image into 100 images. The 84 positive examples for "happy" thus became 8400 positive samples. Training runs required more than 30 hours and needed to reach at least level 15 to be effective, 20 levels is better.

The training process for each facial expression includes:

- find all jpg images of that training expression and detect the position and size of faces in images and write every image's path name, file name, face position and face size into a DAT file
- use the binaries (opencv_createsamples.exe) to generate a VEC file with 100 distortions for each face
- merge all VEC files into one single VEC file
- use the other training expressions' images as negative samples
- conduct the training and return a folder of cascade of classifiers
- Use the binaries (convert_cascade.exe) to convert the cascade to XML file
- performance test

We found the Cohn-Kanade database to be inaccurately labeled for each facial expression and had to manually classify the images by expression. Please see the "Complications" section for more about difficulties preparing the images.

The following table summarizes the training images and results.

Expression	# Original Images	# Positive Samples	# Negative Samples	# Training Stages	Finish	Training Time (hours)
Happy	84	8400	More than 400	16	Exit before finish	30
Angry	64	6400	945	15	complete	36
Surprise	78	7800	3118	19	complete	90

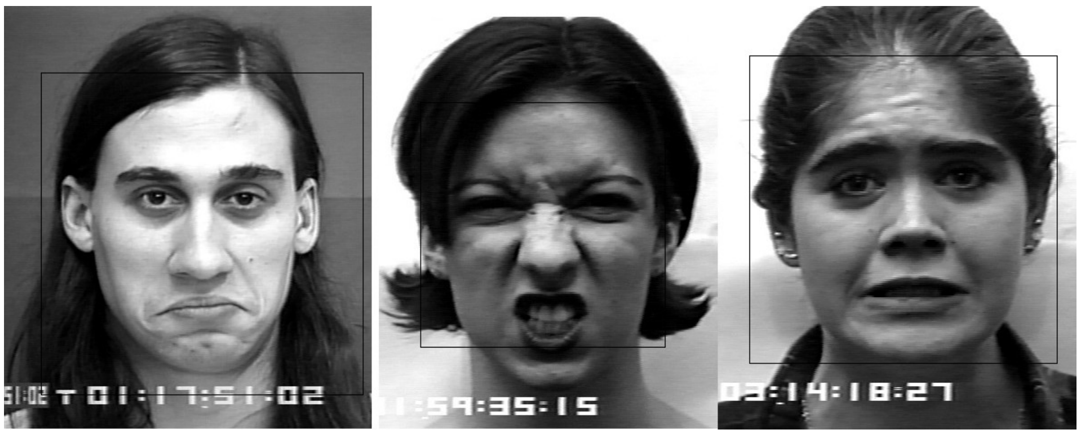
Note: The “happy” cascade has high accuracy in detecting a happy expression but it also has a high rate of false positives where itdetects other facial expressions as happy.

Note: “Angry” has a lower rate of accuracy in detecting angry expressions and also has a lower rate false positives.

Note: “Surprise” has an accuracy rate between the rates of “angry” and “happy”.

Both humans and trained machines do not have a 100% accuracy rate of reading expressions. A very successful training would have an accuracy rate of about 94 – 96% and a human has even lower performance than that We find that our trained cascade for angry can be confused by happy, worried, and sometimes by disgust. With artificial intelligence, if the expression is not recognized, then it should be negative without false positives.

Factors still limiting the effectiveness of the trained algorithm include absence of enough negative training examples and the cultural and individualistic elements of expressions, including lighting, skin color and test subjects “pretending” to be the desired expression. Below are a few examples of incorrectly identified “anger” and “happy”.



Expression Recognition Errors: anger (left), happy (center), happy (right)

Future Work

In the future, we want to improve the training database so that we will be able to improve the performance of our cascade. We hope our trained cascade can be used in some applications such as intelligent searching. For example, when a user searches, the search engine returns a list of results. The computer can detect how the user feels about the returned result. If it finds the user is not satisfied, it will continue to return new results. Once the computer finds the user is happy, it stops returning new results.

Another interesting application could be an individualized reminder, like in a scenario where someone wants to have a rest every hour when they are working with a computer. Currently they can set an alarm manually. But if the computer can understand their emotions, it can detect the users' impatience, tiredness, boredom and so on, and then the computer can suggest the user take a break or try some other more interesting activity to get rid of that negative emotion.

No single algorithm can have a 100% accuracy rate for facial expression recognition. Normally a 94% - 96% detection rate is really good. A human has even lower performance than that. One solution to improve accuracy might be a combined method of Adaboost with haar-like features and geometrical detection.

One more idea is to use 3 degree of freedom face tracking for a 3d-desktop VE system. In such an application, the user will see a different display depending on their relative face position.

Lessons Learned

Both Ruimin and Harriet learned that although unequal in their background knowledge and skills, they could work well together, capitalizing on their strengths and maintaining good humor to succeed in the public arena of MICWIC with their poster and game demonstration.

We learned that it is very time consuming to train a cascade. Ruimin designed and implemented code to process the images in preparation for training, finding the face in each image and creating negative and positive samples in appropriate VEC files. Ruimin learned that the machine learning model of OpenCV is not exactly user friendly, but it is free. The missing "_cvhaartraining.cpp" file was a big disappointment and demonstrated OpenCV's unreliability.

Ruimin learned that training an algorithm takes thousands of images! And that it is easy for the machine learning algorithm to get the expression wrong. The more training examples, both positive and negative, the better the training.

Harriet learned that it is very difficult to feel fear just for the camera. While sorting the Cohn-Kanade image database, fear was much harder to read. Expression "give-aways" such as scrunched eyebrows, open mouth, pursed mouth and other tell tale signs became clear when searching for all images of one expression. Overall, most of the test subjects did not really have a convincing fear expression.

Harriet was surprised and disappointed to learn that OpenCV did not have a store bought ready to go function for expression recognition and that the project was going to become an artificial intelligence exercise. Harriet learned that just because there are examples of OpenCV projects with expression recognition on the internet, the fine print says “We will not share our code” and assuming that OpenCV provides this functionality is wrong. This was a learning opportunity about the intersection of many subfields of computer science in virtual environments, such as computer vision, artificial intelligence and graph / image precessing.

Harriet was dismayed how hard it was to get the development environment straightened out, including special permissions and help from authorized administrators.

The final presentation was a great example of the strong teamwork that developed between us including our support of each other’s efforts. We not only cooperatively created the content and organization of the slides, we also practiced the oral explanation and succeeded in nimbly sharing and conveying the main points of the project to the class.

Complications

Ruimin conquered difficulties with the image database and with OpenCV excluding the necessary source file for haar-like training. Harriet conquered difficulties with equipment, environment set up, and permissions.

A few of the problems include:
1. the original database had too few images
2. the Cohn-Kanade image database “readme” file had inaccurate file naming conventions to identify the various expressions
3. there was no accompanying information file to identify the positive and negative examples of each expression and the coordinates and size of the face
4. there was no way in OpenCV to create the required a VEC file from multiple images with multiple distortions
5. the function “startSampleDistortion could not be used to modify the training files because its .cpp file (_cvhaartraining.cpp) was missing
6. there was no ready to use expression recognition functionality in OpenCV
7. There were no GUI creation libraries on the lab Windows machines or on Ruimin’s laptop and there were no readily apparent free libraries or WYSIWYG design tools to create a GUI for the game
8. the MTU labs do not provide webcams
9. the webcam we borrowed did not have a driver
10. OpenCV would not install in the Linux labs even with csmaint’s help
11. there were no lab Windows development environments to match Ruimin’s development set up
12. installation was not allowed in the special permission Windows lab
13. the finished game was hard coded for a larger resolution webcam than the borrowed webcam, so the game was not visible with the borrowed webcam
14. Harriet was unfamiliar with the Windows 7 environment as well as anything to do with

artificial intelligence, adaboost, cascades, machine learning, and trained algorithms

Solutions

Ruimin used her prodigious research skills to find help online and other code sources and samples, solving all of the complications. She resourcefully re-used the face tracking code she wrote using OpenCV to create original code to automatically generate the required information file for the expression training images by finding the face coordinates, width, and height and outputting to the data file. She also found online some code that she modified to create the graph of all the training images as a vector, merging 8400 images into one VEC file.

Harriet needed to solve many environment complications and eventually requested and received special permission to use the HCI lab which has Windows machines. She used the helpful workers in CSMaint to get the development environment basically functional and was able to code the game program to have relative webcam size and improved game display functionality. Ruimin patiently explained the basics of the AI used in this project to Harriet.

Harriet manually classified the images from the Cohn-Kanade database and Ruimin did the work of running her original programs to prepare the images as a VEC file and DAT file. See “Technology” and “Preparing the Images” sections for more details.

What OpenCV Provided

OpenCV did provide lots of functionality for image manipulation and since it is open source it was free and there was documentation online. OpenCV also provided some simple drawing capabilities in the webcam display window which helped us to create the game, the guide boxes and tracking boxes directly in the display.

OpenCV also provided the trained cascade and function for face tracking with the webcam which we used in the game. OpenCV also provided a method to create positive training image examples from one image with multiple distortions and the capability to create positive training image examples from multiple images, but not for negative training examples which are also required for effective training.

Division of Labor

We were encouraged to reach outside our comfort zones, but our knowledge and experience backgrounds, as well as access to resources, were unequal. This situation resolved comfortably with a willingness to fill in any gaps and do anything to assist. There were occasional communication confusions which were always addressed right away with satisfactory solutions. Traveling together to MICWIC and presenting the poster and game demonstration cemented the team into a happy partnership.

Clearly Ruimin, who intends to continue this work and is interested in expression recognition for a long term project, had much more background and completed much of the project. Harriet was an enthusiastic helper, doing game display programming, sorting expression photos, and writing most of the documents throughout the project, including the poster presented at MICWIC.

Timeline

Original Proposal Goal	***Expected***	***Actual Task***	***Completed***
Use openCV in C++ with video capture on mac laptop	completed	Same	Mid February
Use openCV in C++ with video capture in Linux w/plug in webcam	Feb 25	failed	
	Feb 25	Get permission to use Windows HCI lab	Mid February
		Get plugin webcam working in Windows HCI lab	Late February
		Use openCV in C++ in Windows HCI lab w/plug in webcam	Late February
Detect face	completed	Same	Mid February
Detect face size and location in the frame	March 11	For still image	Mid February
		Get expression recognition training database	Late February
		Judy reads 3 papers and a book to understand how machine learning algorithm adaBoost works to help recognize facial expressions	Late February
		code of adaboost algorithm working but not enough training images	Late February
		Code to train our own cascade for face-tracking to test if expression recognition code written correctly, code correct, not enough training images	Late February
Program prototype game board	March 11		Early March
Program keystroke control of game, or GUI	March 11	“c” key to capture game play image	Early March
Design final game board	March 25	Keep game board inside webcam display	Early March

Program game board and game rules, including taking turns or racing the clock	March 25	Keep game simple with tic tac toe	Early March
Detect two faces, size, and location	March 11	On webcam, detects as many faces as fully visible	Early March
		Judy writes poster proposal for MICWIC	Early March
		Detect face size and location from webcam, example: http://www.youtube.com/watch?v=8pog2i40oGY	Early March
		Webcam face tracking works on HCI Windows	Early March
		Game code is done with webcam	Mid March
		Present game and progress slides to VE class	Mid March
		Game code works on HCI Windows, requiring adjustment to relative camera window size for game components	Late March
		Practice presenting poster and game demonstration to class and get constructive feedback to improve game	Late March
		Present poster and game demonstration at MICWIC	April 1
		Categorize Cohn-Kadade image database by hand	Early April
		Create photos of ourselves for testing	Mid April
Detect expression and program expression recognition	March 25	Still image only	Mid April
		Webcam with expression recognition for happy, surprise, and anger	April 21 !
Use expression to dictate game moves	April 8	Contingency not done	
Use expression to dictate starting up music or script or	April 8	Contingency not done	

mini videos			
Program frame manipulation to create anaglyph 3-d image	April 25	Contingency not done	
Program user’s ability to select captured expression to view in anaglyph 3-d	April 25	Contingency not done	
Display anaglyph 3-d captured expression image	April 25	Contingency not done	

Related Documents

See files referred to in Appendices in directory appendices included with this report. Each subdirectory has a README file explaining the contents, including which files are original, which are library, the images used for training and more. See Appendices for a partial listing of the types of related documents found in the Appendices directory.

Physical Resources

We are able to run our game on a Windows laptop with webcam. We installed openCV and learned how to use it. We are able to use the webcam. We both learned OpenCV and wrote C++ code in Windows Visual Studio development environment. We were able to pass .cpp files back and forth via email and compiling them in our respective environments.

References

Machine learning tutorial: http://note.sonots.com/SciSoftware/haartraining.html
OpenCV machine learning reference: http://www.cognotics.com/opencv/docs/1.0/ref/opencvref_ml.htm
Some example of machine learning in OpenCV: http://www.cognotics.com/opencv/docs/1.0/ref/opencvref_ml.htm
Machine Learning data sets: http://vasc.ri.cmu.edu/idb/html/face/facial_expression/ http://www.ics.uci.edu/~mlearn/MLRepository.html
Three papers related to adaBoost Y. Freund, A more robust boosting algorithm. May 2009. URL: http://arxiv.org/abs/0905.2138 Y. Freund and R. E. Schapire, Experiments with a new boosting algorithm. In Machine Learning: Proceedings of the Thirteenth International Conference, pages 148-156, 1996 P. Viola and M. Jones, Rapid object detection using a boosted cascade of simple features, CVPR, 2001
Facial Expressions: http://en.wikipedia.org/wiki/Facial_expression

Facial Expressions: http://www.daniellaberge.com/grooming/beautyexpressions1.htm
OpenCV 2.1 install video on Visual Studio 2010 http://www.youtube.com/watch?v=XeBhwbRoKvk
"python26_d.lib" error
http://opencv-users.1802565.n2.nabble.com/python26-d-lib-td4963149.html
OpenCV tutorial http://www.pages.drexel.edu/~nk752/tutorials.html
OpenCV machine learning tutorial http://note.sonots.com/SciSoftware/haartraining.html
OpenCV machine learning reference http://www.cognotics.com/opencv/docs/1.0/ref/opencvref_ml.htm
Machine Learning lecture http://stat-www.berkeley.edu/users/breiman/wald2002-1.pdf
Some example of machine learning in OpenCV http://www.cognotics.com/opencv/docs/1.0/ref/opencvref_ml.htm
Machine Learning data set http://www.ics.uci.edu/~mlearn/MLRepository.html
Face data set download http://archive.ics.uci.edu/ml/datasets/CMU+Face+Images http://vasc.ri.cmu.edu/idb/html/face/facial_expression/
Reference for understanding the machine learning algorithm of detecting facial expression in real-time paper: Facial expression recognition using encoded dynamic features paper: Robust real-time object detection, Paul Viola and Michael J. Jones paper: A short introduction to boosting, Yoav Freund and Robert E. Schapire (adaBoost) find all *.png files in current directory and copy them to
convert txt file to dat file mv happy.txt happy.dat
create positive training samples ./opencv_createsamples -info happy.dat -vec positive.vec -w 96 -h 96 -show
convert cascade into xml file ./convert_cascade –size="24x24" haarcascade haarcascade.xml
note: 24x24 is the size of output sample width and height; haarcascade is the directory that save the trained cascade.
Test the performance of trained cascade ./opencv_performanced -data haarcascade.xml -info happy.dat -ni
Create a number of positive samples from a single images

$ createsamples -img face.png -num 10 -bg negative.dat -vec samples.vec -maxxangle 0.6 -maxyangle 0 -maxzangle 0.3 -maxidev 100 -bgcolor 0 -bgthresh 0 -w 24 -h 24
Show training images from a .vec file $ createsamples -vec samples.vec -w 20 -h 20
Use single .jpg images to create a .vec with multiple training images with distortion and then use "mergevec.exe" to merge multiple .vec files into a .vec file ./mergevec collection_vec.dat output_collection_vec.vec -show -w 24 -h 24 note: collection_vec.dat is a file with multiple single .vec filenames; output_collection_vec.vec is the output .vec file
Training ./opencv_haartraining -data haarcascade -vec samples.vec -bg negatives.dat -nstages 20 -nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 7000 -nneg 3019 -w 20 -h 20 -nonsym -mem 512 -mode ALL
Anger training: ./opencv_haartraining -data haarcascade_anger_final -vec anger_positive_collection.vec -bg negatives_anger.dat -nstages 20 -nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 6400 -nneg 945 -w 20 -h 20 -nonsym -mem 512 -mode ALL
3D Anaglyph Reference A simple computation to improve the quality of anaglyph http://www.3dtv.at/knowhow/anaglyphcomparison_en.aspx
A course about anaglyph **http://www.helloapps.com/Stereoscopic/**

Appendix I: Original Brainstorming and References Table

See “ideaBrainstorming.odt”.

Appendix II: Original Proposal

See “projectProposal.pdf”.

Appendix III: Poster for MICWIC

See “Veposter_hk_ruimin.pdf”.

Appendix IV: Example Images

See “sortPics” directory and sub directories.

Appendix V: Sample Expression Diagrams

See “sampleExpressionDiagrams.pdf”.

Appendix VI: Data File Example for Training Expression Algorithm

See file “allN.dat”
sample data from “allN.dat” with explanation:
[dir / image filename] [number of images]
 [x position of face] [y position] [width of face] [height of face]

```
all/S010_001_01594215.jpg 1 231 106 299 299
all/S010_001_01594226.jpg 1 225 95 312 312
all/S010_002_01593902.jpg 1 228 97 300 300
all/S010_003_01595414.jpg 1 229 107 303 303
```

Appendix VII: C++ Code to Prepare Image Database for Expression Training

In the directory of OpenCVConfig

Appendix IX: C++ Code from Webcam Face Tracking Game

In the directory of OpenCVConfig

Appendix X: XML from OpenCV Provided Face Tracking Algorithm

In the directory of OpenCVConfig

Appendix XI: Performance of Training File

See “performance_anger.txt” for anger training.
See “happy_anger.txt” for anger training. FIXME get from Ruimin

Appendix XII: Slides from Final Presentation to Class

final slides3.pdf