Summer Research Final Report

Project Title:
Face Detection and Tracking

Name:
Coney Dorsey

Group Members:
Ryan White
James Ashenhurst

Advisor:
Dr. Yan Meng

May 2005 – August 2005

This research is sponsored by the Department of Defense

Table of Contents:

Summary:

During the summer research intern and Stevens Institute of Technology I have learned a lot. When I first came I didn't know what to expect. I didn't know what kind of work I would be doing. Then I met with my advisor Dr. Meng. She gave me and my group members an overview of what she wanted us to do. When she first told me face detection and face tracking I thought this can't be that bad. But then when she told us that it had to be programmed in C++ I quickly realized the challenge I had before me. I had taken C++ at Jackson State University but the programs done there had no comparison to what she wanted me to do. So I knew I had to buckle down and do some real research. Before I came to Stevens I knew nothing about face tracking, face detection, or even Image Processing. So I had a lot of research to do. But Dr. Meng was very helpful. She told us that the purpose of our project was to be able to detect a person's face from using either skin color or facial features and track the face as the person moves out of the parameters of the camera. She also gave us some papers on some of the basics of what we needed to know about image processing. I read that and I kind of had an idea of what it was. Then she emailed me a couple of websites on image processing. I looked at that and got an even better understanding. So as the weeks past along Dr. Meng realized that the project would take too long and be a bit of a hassle if all of us were working on the same thing. So she divided us into two groups. I was assigned to the Face Tracking part and my other partners were assigned to the face detection. That helped a lot because now we don't have to try to learn a lot of things about a broad subject. We can learn our parts and put what we've learned together. So as the weeks went on I went to different websites trying to learn exactly what face tracking was. I found out a lot of information and noted it. I also found different C++ functions and libraries. So when I got all of the information I thought I needed it was time to implement.

That was the hard part. I tried combining everything I had learned about face tracking and the libraries I had found and it seem like nothing worked. I realized that this is harder that I thought it would be. This was my first time really doing a major program. I realized that programming can be a real headache. After numerous tries to put the program together I got really frustrated. Not only about trying to do the program itself, but of things that was in a way hindering me from doing my program. My computer crashed with all my work saved on it. It crashed because of a virus through the network. Although that didn't stop me from doing my research it taught me a lesson. That lesson was to always back up my work. I started back putting the peaces back together. In the end I didn't get the program to run but I think that I have set a strong foundation for the students that will work on this same project in the future. They want have to do all of the research that I had to do because they can pick up where I left off.

Introduction:

Computer Vision

First of all to understand my project you have to understand computer vision. Face tracking and detection features in sequence is an important and fundamental problem in computer vision. This area of research has a lot of applications in face identification systems, model based coding, gaze detection, human computer, interaction, teleconferencing, etc. human-computer interaction, teleconferencing, etc.

Open CV

OpenCV means Intel® Open Source Computer Vision Library. It is a collection of C functions and a few C++ classes that implement some popular Image Processing and Computer Vision algorithms. OpenCV has cross-platform middle-to-high level API that consists of a few hundreds C functions. It does not rely on external libraries, though it can use some when it is possible. OpenCV is free for both non-commercial and commercial use. OpenCV provides transparent interface to Intel Integrated Performance Primitives (IPP). That is, it loads automatically IPP libraries optimized for specific processor at runtime, if they are available.

Whats new in Open CV

- (Linux) Support for more cameras has been added in highgui. Different versions of libdc1394 can now be used. (thanks to Frederic Devernay for the new versions of cvcap_dc1394.cpp, cvcap_v4l.cpp and patches for configure script, thanks to Sfuncia Fabio for the patch for cvcap_v4l.cpp)

- (Linux) More types of video files can now be read by using libavformat and libavcodec from ffmpeg-0.4.9pre1 (thanks to Frederic Devernay for new version of cvcap.cpp and patches for configure script)

- (Linux) Python wrappers for OpenCV have been created by Olivier Bornet and Mark Asbach using SWIG. See opencv/interfaces/python and opencv/samples/python. While the wrappers should be OS-independent, so far they have been built on Linux only.

- OpenCV now builds and runs on 64-bit platforms: EM64T (a.k.a. AMD64) and IA64 (Itanium). Extra configurations have been added to project files for MsDevStudio 6.0.

- Performance tests for cxcore and part of cv have been created. the output format is plain csv and is similar to the one used in IPP. run "cxcoretest -t" and "cvtest -t".

- Haartraining now automatically produces .xml database along with the usual directory tree.

- Script for creating custom dynamic library for a subset of IPP, used by OpenCV, has been created. Look at opencv/interfaces/ipp

- Several new functions have been added: Background/foreground segmentation (see cvaux/include/cvaux.h, "Background/foreground segmentation" section andcvaux/src/), cvHoughCircles (circle detection), cvPointPolygonTest, cvRemap (generic geometrical transformation), cvLogPolar (log-polar transform), cvEqualizeHist (histogram equalization), cvCornerHarris (Harris corner detector).

- Camera calibration and epipolar geometry functions have been completely rewritten, API was simplified, and docs updated.

- New checkerboard detection algorithm (based on Vladimir Vezhnevets' code) is now used.

- cvCvtColor supports new color models (HLS, CIE L*u*v*), for every RGB->something transformation the inverse is provided, 32f format is completely supported, 16u is partially supported.

- Distance transform was extended to find the nearest connected component of zero pixels for every pixel, not only the distance to it.
- (Windows) Highgui now remembers positions of a last few opened windows in registry.

Image Processing

Computer manipulation of images. Some of the many algorithms used in image processing include convolution (on which many others are based), FFT, DCT, thinning (or skeletonisation), edge detection and contrast enhancement. These are usually implemented in software but may also use special purpose hardware for speed. Image processing contrasts with computer graphics, which is usually more concerned with the generation of artificial images, and visualisation, which attempts to understand (real-world) data by displaying it as an artificial image (e.g. a graph). Image processing is used in image recognition and computer vision. Silicon Graphics manufacture workstations which are often used for image processing. There are a few programming languages designed for image processing, e.g. CELIP, VPL, C++.

Face Tracking

Face detection and tracking are important in video content analysis since the most important objects in most video are human beings. Research on face tracking and animation techniques has been improved due to its wide range of applications in security, entertainment industry, gaming, psychological facial expression analysis and human computer interaction. Recent advances in face video processing and compression have made face-to-face communication be practical in real world applications. However, higher bandwidth is still highly demanded due to the increasing intensive communication. Model based low bit rate transmission with high quality video offers a great potential to mitigate the problem raised by limited communication resources. However, after a decade's effort, robust and realistic real time face tracking and generation still pose a big challenge. The difficulty lies in a number of issues including the real time face feature tracking under a variety of imaging conditions such as lighting variation, pose change, self-occlusion and multiple non-rigid features deformation and the real time realistic face modeling using a very limited number of feature parameters. Traditionally, the head motion is modeled as a 3D rigid motion with the local skin deformation, the linear motion tracking method cannot represent the rapid head motion and dramatic expression change accurately.

The appearance-driven approach requires a significant number of training data to enumerate all the possible appearances of features. The model based approach assumes the knowledge of a specific object is available, meanwhile the requirement of frontal facial views and constant illumination limited its application. All above tracking methods have shown certain limitations for accurate face feature tracking under complex imaging conditions. Different types of facial features, like skin color, edges, feature points, motion, have been used for face tracking. Skin color is tried for tracking face motion in X, Y direction and out-of-plane rotation in. It is often too simple to encode structural knowledge of face, it is thus good for coarse face tracking. An optical flow field has been adopted for face tracking. Dense motion information makes face tracking easier. A major constraint is that optical flow estimation is subject to the aperture effect and usually does not allow big movement. Salient facial feature points are better choice for accurate face tracking. The main shortcoming is that tracking of point features is easily impaired by noise and often the face appearing in video should be large enough to facilitate tracking. Face tracking can serve as a front end to further analysis modules, such as face recognition, face expression analysis, gaze tracking, and lip reading. Face tracking is also a core component to enable the computer to see the computer user in a Human Computer Interface system.

Libraries and functions I found

-#include <camera.h>
-mycamera;
-camera.openDevice("/dev/video");
-IplImage* image;
-image = CreateImagespace (image size, camparameters);
-imgWindow("Image",1);
-mycam.grabImage();
-convertScale(faceImage, movingImg);

Tracking part.....
-TrackFram( );
-TrackNextFrame( );

FtInitialization

FtCalculate
FtTrackNextFrame

```
FTrack.FtInitialization (640, 480);
ImgData = ReadAFrame();
RECT FaceRect = GetFacePosition ();
FTrack.FtCalculate (FaceRect, ImgData);
ImgData = ReadAFrame();
FTrack.FtTrackNextFrame (FaceRect, ImgData);
```

(Functions for tracking)

```
#define ORIG_WIN_SIZE   24

//dimensions of the classimage used to detect facial features
#define FEATURES_WIDTH  30
#define FEATURES_HEIGHT 30

CvMemStorage* storage=NULL;
CvHidHaarClassifierCascade* hid_cascade = 0;  //face classifier
IplImage* video_image[2];     //stores the scene
int width,height;             //dimensions of the scene bitmap
int object_location[100][4];  //stores the locations of detected objects
int border_x,border_top,border_bottom;        //border around the face in
pixels
classimage *features=NULL;    //used for detection of facial freatures

int trackfeature[10][10];
bool trackingEnabled=false;
int trackingTimer=0;
```

```
//-------------------------------------------------------------------------------
//sets the border region around detected faces
//-------------------------------------------------------------------------------
```

```
void CFacedetectApp::RCobj_setBorder(int borderX, int borderTop, int
borderBottom)
{
  border_x = borderX;
  border_top = borderTop;
  border_bottom = borderBottom;
}


//-------------------------------------------------------------------------------------
//track facial features
//-------------------------------------------------------------------------------------
void CFacedetectApp::RCobj_trackface(int index)
{
  int lateral_symetry, lefteye_x, righteye_x, lefteye_y, righteye_y, mouth_y,
mouth_width;
  int tx,ty,bx,by,dx,dy,w,h;

  RCobj_detectfeatures(index, lateral_symetry, lefteye_x, righteye_x,
lefteye_y, righteye_y, mouth_y, mouth_width);

  //get the box for the face
  tx = object_location[index][0];
  ty = object_location[index][1];
  bx = object_location[index][2];
  by = object_location[index][3];
  dx = bx-tx;
  dy = by-ty;




  //nose position
  trackfeature[index][3] = tx + ((lateral_symetry * dx) / 100) - w;
  trackfeature[index][4] = ty + (dy/2) - h;

  //left eye
  trackfeature[index][5] = tx + ((lefteye_x * dx) / 100) - w;
  trackfeature[index][6] = ty + ((lefteye_y * dy) / 100) - h;
```

```
  //right eye
  trackfeature[index][7] = tx + ((righteye_x * dx) / 100) - w;
  trackfeature[index][8] = ty + ((righteye_y * dy) / 100) - h;

  trackingEnabled=true;

}


//-------------------------------------------------------------------------------------
//track faces
//-------------------------------------------------------------------------------------
void CFacedetectApp::updateTracking()
{
  int i,x,y,size_x,size_y,searchArea,x1,y1;
  bool found=false;

  for (i=0;i<10;i++)
  {
    if (trackfeature[i][0]==1)
          {
      found=true;

          //get the size of the area to be tracked
      size_x = trackfeature[i][1];
      size_y = trackfeature[i][2];
          searchArea = size_x/4;
          if (searchArea<2) searchArea=2;


          //track head
          x = object_location[i][0];
          y = object_location[i][1];
          x1=x;
          y1=y;
          //closestMatch(x,y,object_location[i][2]-
object_location[i][0],object_location[i][3]-object_location[i][1],searchArea);

      object_location[i][0] += (x-x1);
      object_location[i][1] += (y-y1);
```

```
    object_location[i][2] += (x-x1);
    object_location[i][3] += (y-y1);
        if (object_location[i][0]<0) object_location[i][0]=0;
        if (object_location[i][1]<0) object_location[i][1]=0;
        if (object_location[i][2]>=width) object_location[i][2]=width-1;
        if (object_location[i][3]>=height) object_location[i][3]=height-1;


        //track nose
        x = trackfeature[i][3];
        y = trackfeature[i][4];
        x1=x;
        y1=y;
        closestMatch(x,y,size_x,size_y,searchArea);
        trackfeature[i][3] = x;
        trackfeature[i][4] = y;

        //track left eye
        x = trackfeature[i][5];
        y = trackfeature[i][6];
        closestMatch(x,y,size_x,size_y,searchArea);
        trackfeature[i][5] = x;
        trackfeature[i][6] = y;

        //track right eye
        x = trackfeature[i][7];
        y = trackfeature[i][8];
        closestMatch(x,y,size_x,size_y,searchArea);
        trackfeature[i][7] = x;
        trackfeature[i][8] = y;

      }
}
```

Conclusions:

Since I have been here a Stevens Institute of Technology I must say I have learned a lot. I have learned about image processing and a lot of the things that comes with face detection and face tracking as a whole. Not only have I learned things about my project I have also learned different techniques about my project. I have learned the importance of face tracking and face detection in the real world. I learned that there are a lot of face tracking and face detection programs and software but the challenge for programmers today is how to make it more robust and how to modify it to fit each individual specific need. I have also learned things that will benefit me in my future education in school. I've learned things such as programming, teamwork adapting to new environments, and oral presentations. Presenting in front of an audience is also a thing I need to know and I'm glad that we had to present our weekly presentations like that. I feel like that really helped me a lot because on the upcoming semester at Jackson State University I have to take an oral speech course. I think that I will have an advantage on the others students from my experience at Stevens. Working here at Stevens not only gave me the experience about things in technology but I also experienced my surrounding area. I experience New York city and a lot of the historical sites and famous buildings and architectures. I love the environment of Stevens and I think that the institute has a wonder Engineering Dept. I have gain a lot of knowledge that can be useful in the work force when I graduate from college. I am glad I chose to come to this intern and I would definitely refer it to many other students.

References:

Open CV
http://www.sourceforge.net/projects/opencvlibrary

Computer Vision Homepage
http://www-2.cs.cmu.edu/~cil/vision.html

Face Recognition Homepage
http://www.face-rec.org/

Tracking and Image Feature
http://www.lyric.com/fcp-plugins/creating-tracks/creating-tracks.htm#tracking

Face Detection and Tracking from video imagery
http://www.ecse.rpi.edu/~qji/FaceDet/TSWG_slides.pdf#search='face%20tracking%20algorithms

Motion Based Segmentation and Optical flow
http://www.fuzzgun.btinternet.co.uk/rodney/components.htm#Motion

Computer Vision Source Codes
http://www.cs.cmu.edu/afs/cs/project/cil/ftp/html/txtv-source.html

Real Time Face Detection
http://www.iis.fraunhofer.de/bv/biometrie/download/index.html

Vision Based Face Tracking Home Page
http://synapse.vit.iit.nrc.ca/doc/facetracking.html

Facial Video Memory
http://www.perceptual-vision.com/memory/

Facial Video Memory Demos and Downloads
http://www.perceptual-vision.com/memory/download.html

Resources for Face Detection
http://vision.ai.uiuc.edu/mhyang/face-detection-survey.html

Digital Imaging Processing: Face Detection
http://www.stanford.edu/class/ee368/Project_03/Project/reports/ee368group08.pdf

People Tracking
http://www.siebel-research.de/people_tracking/reading_people_tracker/

Ryan White's Webpage
http://rylewh.tripod.com/id2.html

James Ashenhurst's Webpage
http://cs.oberlin.edu/Members/jashenhu/REU/

Open CV Home
http://www.intel.com/technology/computing/opencv/index.htm

Appendix: Weekly Reports:

Week 1:

In week one I basically read a couple of chapters that Dr. Meng gave our group. The chapters weren't actually on the project that we will be working on, but it gave me a start on the basics and some of the terminology that will be used on our project. I also went to a couple of web sites that will be very beneficial towards me understanding exactly what I have to do. I found out a lot on image tracking and motion detecting from the web site that Dr. Meng recommended. I also got a chance to see the camera that we will be using to track the motion in the project. I look forward to the following week because I will be trying to find a way to implement the motion detection and tracking process into the c++ programming code. It should be challenging!!!!!

Week 2:

Basically in week 2 I look at a lot of different web pages on motion detection and image tracking. I also made a list of things I want to accomplish in the following week. This week I will focus on trying to figure out what c++ libraries and functions I will need to make the camera interact with the image and the tracking part of the project. We also decided to divide our group into two parts. I am responsible for the image detection and tracking part. I was giving a helpful web page by Dr. Meng and I am findind it helpful towards understand how to put everything together. I also looked at a couple of sites that helped me better understand object detection and tracking. I found that the tracking part is actually sequences of images which are called frames, displayed in fast enough frequency so that the human eye can detect the constant movement of its content. All image processing techniques can be applied to individual frames. I found that the object detection is locating it precisely for recognition. Object tracking is to monitor an object's spatial changes during a video sequence, including its presence and position. The problem is matching the target region in frames of a sequence of images taken at close time intervals. These two processes

are closely related because tracking usually starts with detecting objects, while detecting an object repeatedly in sequence is often necessary to help and verify tracking.

Week 3:

- found some C++ libraries & functions that I will be using in tracking the object.

- some of the libraries and functions I found was...

-#include <camera.h>
-mycamera;
-camera.openDevice("/dev/video");
-IplImage* image;
-image = CreateImagespace (image size, camparameters);
-imgWindow("Image",1);
-mycam.grabImage();
-convertScale(faceImage, movingImg);

Tracking part.....
-TrackFram( );
-TrackNextFrame( );
.....

-Familiarized myself with different tracking techniques that I can use.

Week 4:

Studied a couple of image traking program algorithms to see step by step the approach they took in writing the program.
Found some information on how to make the box stay around the persons face while they are moving.
Found a paper on Microsofts web site about important facts I need to know about doing a face tracking program.
It included topics like.....
Requirements for robust tracking (anti failure robustness and post failure robustness)
Different approaches to object tracking...

Boundary based approaches
Region based approaches
Found more Libraries and Functions for the tracking part of the program

Week 5:

Got the camera from Dr. Meng
Look at Face Tracking algorithms
Found more libraries useful in program
Studied a lot of already written Face Detection and Tracking Programs
Found some tracking programs and went trough the program demos to better understand what our program suppose to do.

Week 6:
Look at the algorithms of the demos that i found and started writing some code.
got libraries from open cv that would be helpful for tracking
worked on syntax errors after I compiled the code that I have so far
Determined how to interact the camera with the c++ program
Week 7:

Continued coding
Cut errors down to 30 errors
found out how to link the open cv libraries to c++
computer crashed

Week 8:

Fixed Computer:

Created an object of Face Tracking:
MyfaceTrack
MyfaceTrack FTrack;

Has three member functions:
FtInitialization
FtCalculate
FtTrackNextFrame

FTrack.FtInitialization (640, 480);

```
ImgData = ReadAFrame();
RECT FaceRect = GetFacePosition ();
FTrack.FtCalculate (FaceRect, ImgData);
ImgData = ReadAFrame();
FTrack.FtTrackNextFrame (FaceRect, ImgData);
```

Worked on errors from last week:
Week 9:

Wrote more code
Found new libraries
Got with Group member Ryan to try and combine the face detection part with the tracking part
Fixed some errors. (but when I fixed some errors it made more)
Working on another approach on writing some parts of the program

Week 10:
Put some of facial tracking program code together
Worked on final report

Week 11:
Completed Final Report