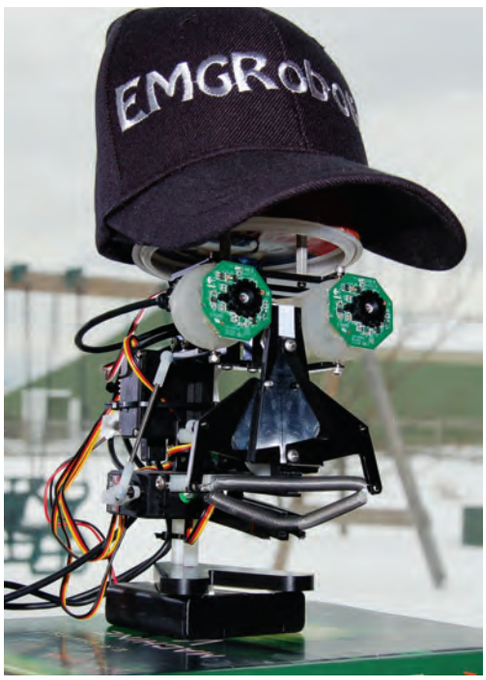
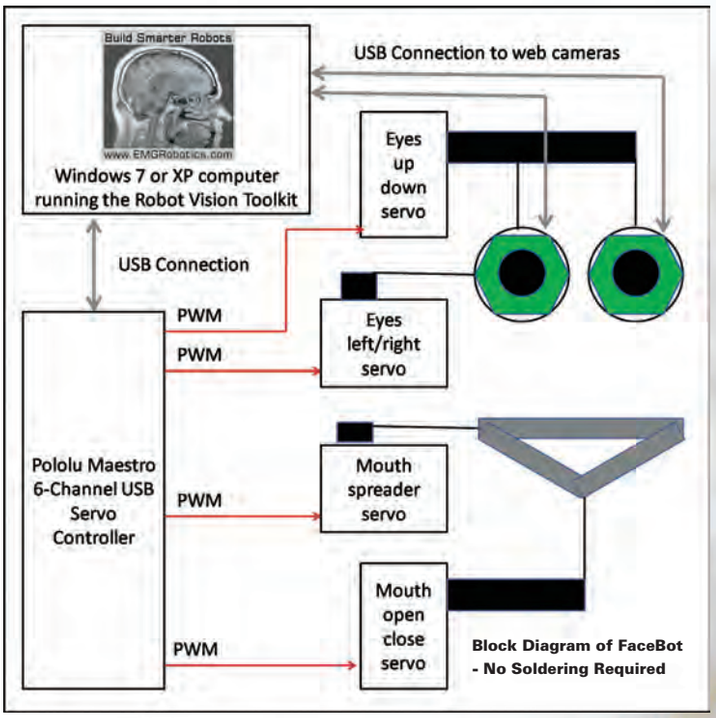
BUILD A FaceBot

by Eric Gregori

A face-tracking, interactive, animatronic head





Block Diagram of FaceBot - No Soldering Required

Face tracking or face detection is an exciting field in Computer Vision. With a simple web camera, some free open source software, and a fun animatronic head kit from www.robodyssey.com. FaceBot will introduce you to face detection and tracking using the easy to learn software from www.EMGRobotics.com.

Tracking a person's face with its eyes while using its mouth to convey a mood, allows us to experiment in human interaction and gestures. The robot conveys two simple moods; happy or sad. The robot appears happy when it sees a face, and sad when it is alone.

ROBODYSSEY ESRA III

The animatronic head for this project was purchased from Robodyssey. The Robodyssey ESRA III (Expressive System for Robotic Animation) is a mechanical face, controlled by four RC servos. The kit ships unassembled, but is easy to assemble by anyone with previous robot kit building experience.

The base kit includes lips and eyes. The lips (controlled by 2 servos) can spread to smile and open to talk. The eyes (controlled by 2 servos) can move up, down, left, and right. The lips are made of springs, while the eyes are mad from ping pong balls. The whole kit is made using CNC cut acetal plastic and comes with easy to follow instructions including pictures for each step.

Cut the switch wires from the web camera PCB

The kit was built following the instruction out of the box. After the construction was complete, the plastic irises were removed from the ping pong ball eyes. Two logitech C200 web cameras were disassembled. The web camera circuit boards were hot glued in place of the iris's in the ping pong ball eyes.

The web camera comes apart easily with a single screw. Carefully separate the sides of the web camera and slide the circuit board out. Cut the wires to the switch. Hot glue the web camera circuit boards to the front of the ping pong ball eyes. The LED indicates the top of the web camera board.

The 4 RC servos are controlled by a Pololu Micro Maestro 6-Channel USB Servo Controller. The Micro Maestro is connected to a laptop via USB. From a software point of view, the maestro appears as a virtual com port. Servos can be controlled by one of three command protocols: compact, Pololu, and mini SSC. The micro maestro must first be configured by the Pololu tool, by following the instructions in the user's guide (see links). The Pololu line of Maestro servo controllers can control from 6 to 24 servos from USB or TTL serial.

The head "talks" using a mouth made of two springs to represent lips controlled by two RC servos. One servo spreads the lips to smile or frown. The other servo opens the mouth, separating the lips, to simulate talking.

SEE THE VIDEO!

Scan this code on your smartphone with a bar reader app or type in **find.botmag.com/051101**. On that webpage you will also have access to schematics, diagrams and further details on the Viola-Jones algorithm, as well as tips on using the face detector in the Robot Vision Toolkit!

Extensions for the head can be purchased from Robodyssey. The eyebrow extension uses one servo to control the center of a unibrow. By raising or lowering the center of the unibrow, expression of anger or surprise can be created. The ear extension uses two servos to control the motion of two "ears". I am not sure what to do with the ears yet.

CONFIGURING THE POLOLU MAESTRO USB SERVO CONTROLLER

The Maestro servo controller has to be configured using a free Windows tool called the Pololu Maestro Control Center available from the Pololu website. Install the Pololu Maestro Control Center and plug the servo controller into your computer via USB. The software will automatically detect the servo controller.

Under the *Serial Setting* tab, verify that USB Dual Port Mode is selected. Then select the *Setup Channel* tab. Under the *Mode* column, select servo for all the servos. This enables the channel to control a servo. Without setting the mode correctly, the servos will not move.

The Pololu controller is a smart controller. Each servo can be configured with an acceleration and speed. This ability is important when you do not want direct control of the servo. The servo controller internally generates a motion profile based on the current position and the desired position. This feature is used for the robots mouth. The PC simply sends a destination position to the servo controller. The servos controller will then use the acceleration and speed setting to move the servo from its current position to its commanded position. This significantly decreases the amount of data that the PC needs to send the servo controller.

The automatic motion profiling can be turned off and on a per-servo basis, by setting the acceleration and speed to 0. This turns the "smarts" of the servo controller off. The servo controller simply translates the command from the PC over USB, into a servo position, and sends it to the servo. The servo will then move at its maximum speed to the new position.

The eye motion is controlled by a simple closed-loop algorithm written in RobotSee, running on the PC. Motion profiling is turned off for both eye servos because it interferes with the closed-loop algorithm controlling the servos from RobotSee.

Screen shots of both the channel and serial tabs described above, videos and much more are available if you type **find.botmag.com/051101** into your web browser. The full configuration file is also available from the web site. The configuration file can be loaded into the Maestro Control Center using the File button in the upper left corner of the window. If using the configuration file, be sure your servos are wired to the servo controller in the same order as specified in the *Setup Channel* screen shot also available on the Robot Magazine web site.

FACE DETECTION

Using face detection and tracking, the web camera eyes track people's faces while the head is interacting with them. The images from the web camera eyes are passed through an algorithm. The algorithm separates the objects in the image into two categories: background and faces. Once the faces have been identified, the software chooses the closest face and starts tracking its location within the image. The face location within the image is used to control the eye servos. The control algorithm keeps the face in the center of the image by physically moving the web camera eyes using the eye servos.

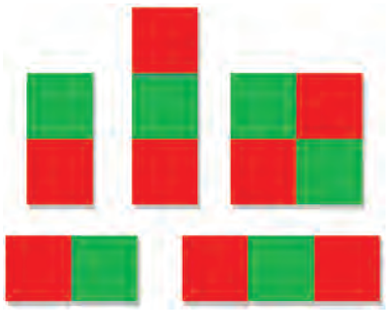
There are many algorithms that perform face detection in an image or video stream. This article concentrates on the Viola-Jones Object Detection Framework as described in their papers titled; Rapid Object Detection Using a Boosted Cascade of Simple Features and Fast Face Detection using a Cascade of Detectors.

The following is a simple summary of how the Viola-Jones algorithm works. You can download a more complete description of the algorithm at **find.botmag.com/051101**. Look for the whitepaper, Viola Jones Simplified by Eric Gregori. The Robot Vision Toolkit provides an easy to use implementation of the Viola Jones algorithm. Using the Robot Vision Toolkit and RobotSee, you can be detecting and tracking faces in a matter of minutes with little or no programming skills.

The Viola-Jones algorithm uses Haar features to detect faces. Haar features are simply groups of adjacent pixels in a specific pattern. Pixel values from half the group are subtracted from the other half. The resulting value provides a unique signature for that group of pixels.

The face pictures in this article were provided courtesy of the University of Massachusetts "Labeled Faces in the Wild" project. A Haar feature consists of multiple adjacent areas that are subtracted from each other. Viola - Jones suggested Haar features containing; 2, 3, and 4 areas. The encode signature of a Haar feature is calculated by taking the sum of the pixels under the green square and subtracting the sum of the pixels under the red square.

By encoding the difference between two adjoining areas in an image, the Haar feature is effectively detecting edges. The further the value is from zero, the harder



A Haar feature consists of multiple adjacent areas that are subtracted from each other. The Viola - Jones approach suggests recommends Haar features containing 2, 3 and 4 areas. The encoding signature = Pixels under Green – pixels under red.

Each Haar feature encodes a small portion of the face. The feature shown detects a bright region in the center, with two dark regions on either side.

or more distinct the edge. A value of zero indicates the two areas are equal, thus the pixels under the area have equal average intensities (e.g., the lack of an edge). Individual pixel values are from 0 to 255, with 0 being black, and 255 being white (grayscale).

The unique signature created by summing the Haar feature is thresholded, resulting in a true or false result for each feature. Thresholding is a simple function that converts an analog signal to a digital signal. Any value above the threshold (including the threshold) translates to a true, while any value below the threshold translates to a false.

One Haar feature of a specific type, size, and at a specific location is considered a weak classifier. Classifier is just another word for filter in this context. A weak classifier can be thought of as a filter that sometimes mistakes noise for facial features. A single weak classifier does a poor job of detecting faces by itself. It often mistakes background objects (noise) for faces, because it's only looking for a small portion of the face.

CREATING A STRONG CLASSIFIER

Multiple features of different types, sizes, and at different locations are combined to create a strong classifier. A strong classifier uses multiple weak classifiers (each looking at a different part of the face) to decrease the number of false positives created by the filter. A false positive simply means the classifier calculated that an object in the image was a face, when in fact it was not a face. A strong classifier does a significantly better job of differentiating between faces and background objects.

Multiple strong classifiers are combined in a cascade to create a detector.

Think of a detector as one big filter made up of many smaller filters. A cascade is simply the process of connecting one filter to the next. For example, if you were filtering rocks, you may create a box with multiple filters in it. Each filter lets rocks of smaller and smaller sizes through. The whole box containing multiple filters would be called a detector. The stack of filters inside the box would be called a cascade.

The face detector is designed for a specific size face, commonly 20 x 20 pixels.

The detector is slid across the image looking for a face. At each XY position in the image a 20x20 block of pixels is sent to the detector. The detector indicates a true if it sees a face in the block or a false if it does not. This process is repeated through the whole image.

To detect faces bigger then 20x20 pixels the detector is scaled up and scanned across the image again. This process is repeated until the detector reaches a maximum size.

This image shows just a few of the thousands of features that make up a detector.

The size of a face in an image is proportional to the distance of the face from the front of the camera. Faces further from the camera are smaller and will be detected by the smaller detector. Faces closer to the camera are bigger and will be detected by the bigger detector. This allows the software to get a rough estimate of how far the face is from the camera. This data can be used to tune the software to only detect faces that are relatively close or relatively far away. For instance, if you want to detect close faces while ignoring distance faces, you can simply configure the algorithm to ignore faces detected by the smaller detector. Face detection is easy with the Robot Vision Toolkit, as described in the whitepaper, Using the Face Detector in the Robot Vision Toolkit available at find.botmag.com/051101.

SOFTWARE

The software for FaceBot was written in RobotSee, using the free Robot Vision Toolkit from www.EMGRobotics.com. The Robot Vision Toolkit makes face detection easy by doing all the hard work for you. The Robot Vision Toolkit's face detector implementation is highly configurable but defaults to commonly used parameters. This makes adding face detection to your projects an easy learning experience. It takes only 13 lines of RobotSee to start experimenting with face detection. You can start with the default values and experiment with each parameter to see what happens.

Controlling the servo through a USB connected Pololu servo controller is easy using RobotSee. When the servo controller is plugged into the computer, it creates a virtual com port. Connecting to the USB servo controller is as easy as connecting to a serial port.

13 lines of RobotSee code to detect a face

```
// Global Variable
global( position )
global( x )
global( y )

// Open FaceDet Language Extension
version()                                          // print Interpreter version
loaddll( "RobotSee_FaceDet_DLL.dll" )              // Load FaceDetector Language extension
print( "\nFace Detector DLL version: %x", facedet() )   // Print FaceDet Language Extension Version

// Initialize the face detector to look for full frontal faces.
// Additional details about where to find haar cascades, and
// even create one yourself can be found at www.emgrobotics.com
facedetinit("..\\haarcascades\\haarcascade_frontalface_alt.xml")
delay( 1000 )                                      // Wait for init to complete

infloop:
      position = facedetread(0)                    // Reads Face Position ( 0xXXXXYYYY )
      x = position >> 16                           // 0xXXXX0000
      y = position & 0x0000ffff                    // 0x0000YYYY
      print( " X = ", x, " Y = ", y )              // Display position on console
      delay(100)                                   // Delay 100ms
goto( infloop )
```

The eyes are controlled by two servos for horizontal and vertical control. The eyes are linked together mechanically. The web cameras are mounted to the front of the eyes. Only one camera (or eye) is required for face tracking. The other camera was added to support future experiments in stereo vision.

To track a face, the software must first find the face in the image. This is done in RobotSee using the facedetread(0) keyword. The facedetread(0) keyword returns a 0 if no face is detected in the image or the XY position of a face if one is detected. That's it! It's that simple to detect a face using RobotSee.

If no face is detected, the robot goes into a 'sad' mood. In this mood, the robot scans left and right looking for a face while showing a 'sad face' using its mouth. The robot remains in a sad mood until a face is detected. When the facedetread(0) keyword returns a none zero, a face has been detected.

When a face is detected, the robot enters a 'happy' mood. It shows a smile on its face and starts tracking the face with its eyes. Face tracking is done using two control algorithms, one for horizontal tracking and the other for vertical tracking. The control algorithms are identical and very simple. The goal of face tracking is to keep the face centered in the image by moving the eyes. Using the horizontal or X control algorithm, as an example. The algorithm first subtracts the X position of the face in the image from the value for the center of the image. The RobotSee face detector uses a 300 x 200 image. The X center of the image would be 150. The algorithm subtracts the faces X position from the center (150). This results in an error value that varies depending on how far the face is from the center of the image. The error value is multiplied by 10 then added to the servo X position. Since the error can be positive or negative depending on which side the face is on, the servo X position will increase or decrease (negative error causes servo X to decrease). The multiplier of 10 mentioned above is the gain of the control algorithm. Gain is just another word for multiplication. Gain is required, otherwise, the eye motion would be too slow. Increasing the gain will cause the eyes to track the face more quickly. Decreasing the gain has the opposite effect causing the eyes to lag behind the face.

ELIMINATING EYE OSCILLATION

What happens when the face is in the center of the image? Under perfect conditions, the error would be zero and the servos would not move. In actuality, the eyes move back and forth from one side of center to the other, never stabilizing. This back and

I've also used face tracking in robots used to attract attention at trade shows, this being an example build on an iRobot Create base.

forth motion is referred to as oscillation. To eliminate oscillation, a deadband is used. A deadband is just a range of positions that are ignored by the control algorithm. The X face locations between 145 and 155 are completely ignored by the control algorithm. If the face is within that X range in an image, the control algorithm is not run and as a result the servos do not move. This eliminates any possible oscillations around the center.

The software may sound complicated, but in fact it takes only 120 lines of RobotSee to make it all happen. The full source code (see) file can be downloaded from **find.botmag.com/051101**.

My kids, Josh and Nick, playing with ShowBot

CONCLUSION

FaceBot was my first experiment in gesture based robotics, using universally recognized facial gestures to convey a "mood". FaceBot currently only supports three moods; happy, sad, and surprise. In the near future I will be adding more facial gestures. In addition to gestures, the mouth can also be used for "speaking". The idea is to move the mouth in unison with an audio file being played back by the computer. This is an interesting study on how we use our lips to speak and convey emotion while we speak. As with everything else with robotics, it sounds a lot easier than it is. Look for my attempts on my web page www.EMG Robotics.com.

Watching people interact with FaceBot is an interesting experience for a robot designer. As the builder of the robot, you can imagine the lines of code executing when the robot responds or performs an action. The casual observer (especially kids) sees it as "magic". It opens up their imagination and gets them excited about how things work. I witnessed this with my kids when they interacted with FaceBot as if it was alive, even though; they watched me build it on my kitchen table. ◎

```
// FaceBot - Face tracking loop written in RobotSee
while( 1 )
    delay( 100 )                                    // loop time is 100ms
    pos = facedetread(0)                            // Reads Face Position ( 0xXXXXYYYY )
    x = pos >> 16                                   // X = Face X position
    y = pos & 0x0000ffff                            // Y = Face Y position

    if( x > 0 && y > 0 )                            // Track face if detected in image
        gosub( showhappy, retval )                  // Show happy face

        //X control loop
        if( x < 145 || x > 155 )                    // deadband
            error = 150 - x                         // 150 is center of image
                error *= 10                         // control gain ( p in pid )
                servox += error
                if( servox < 3008 ); servox=3008; endif()// limit servo's min position
                if( servox > 9216 ); servox=9216; endif()// limit servo's max position
                servonum = 0                        // move servo 0 - eyes left/right
                servopos = servox                   // move servo to position
                gosub( movepololuservo, retval )    // send servo move command to Pololu
            endif()
        endif()

        // Y control loop - code clipped for space
    else()
        gosub( showsad, retval )                    // Show sad face
    endif()
wend()
```

A closer look at the X control algorithm for face tracking

Links

iRobot Create (Item no. 4400), www.irobot.com, (781) 345-0200

Labeled Faces in the Wild project, http://vis-www.cs.umass.edu/lfw/

Logitech, www.logitech.com, (800) 231-7717

Open Interface Specification www.irobot.com/filelibrary/pdfs/hrd/create/Create%20Open%20Interface_v2.pdf

Parallax, www.parallax.com, (888) 512-1024

Pololu, www.pololu.com, (702) 262-6648

OpenCV, http://opencv.willowgarage.com

RoboOdyssey ESRA Iii, www.robodyssey.com

Robot Vision Toolkit, www.EMGRobotics.com

Robot Vision Toolkit questions http://groups.google.com/group/robotsee-users?hl=en&pli=1

Videos of the FaceBot, www.youtube.com/seeprogrammer

For more information, please see our source guide on page 89.