# Machine Perception Toolbox


**Ian Fasel, Brett Fortenberry and Javier R. Movellan**
Machine Perception Laboratory
Institute for Neural Computation
University of California, San Diego
http://mplab.ucsd.edu

## Contents

# 1 INTRODUCTION

Welcome to MPT, the Machine Perception Toolbox, from the UCSD Machine Perception Laboratory. The MPT (pronounced "Empty" or "Empty Box") supplies cross-platform libraries for real-time perception primitives, including face detection, eye detection, blink detection, color tracking. Soon it will also include expression recognition, predictive color tracking, and tracking based on multisensor fusion. In addition, it supplies many example applications which show how to embed the MPT functions in applications, and which are useful applications in their own right. Finally, Matlab .mex interfaces are provided for many of these core libraries for those who like using Matlab.

## 1.1 History of the MPT

*Include developers, PIs and sponsors*

## 1.2 References to the the MPT

Please use reference [4] in the references section, and the MPLab Web Site (http://mplab.ucsd.edu) for citing work related to the MPT.

*We should provide binary installation with programs and libraries.*

# 2 BUILDING THE MPT

## 2.1 BUILD PROCESS OUTLINE

The MPT is divided into Libraries, which provide the core functionality, and applications (Apps), which use the libraries. While the libraries are cross-platform, each architecture has its own build process. The build process for UNIX-like platforms is handled via Boost.Build, which is a replacement for the GNU automake, autoconf, etc. build process. On these platforms, Jamfiles replace Makefiles, and the entire set of libraries and applications can be built with a single command. Please read below for instructions on how to use this system. For Mac OS X and Windows platforms, each library has its own project file – Xcode on mac or MSVC on Windows. The Applications also come with their own project files.

*Requirements: ImageMagick, liblcms, libtiff, ljpeg. Available via fink for free at fink.sourceforge.net Requirements for Mac OS X: Xcode, available at apple.com for free* The libraries included are

1. libmputil – a library of cross-platform wrappers and utilities.
2. libmpisearch – a library for real-time frontal face detection.
3. mpisearchMex – a Matlab interface for this library.
4. libmpeyefinder – a library for real-time eye detection.
5. mpeyefinderMex – a Matlab interface for this library.
6. libmpcolortracker – real-time color tracking, adapts using frontal face detector.
7. libmpblinkdetector – real-time blink detection, using eye detector.

*The "mp" needs to be added to eyefinder and colorTracker* The applications included are:

- Linux / Unix
  1. mpisearch – a command line executable that finds faces in images
  2. eyefinder – a command line executable that finds faces and eyes within faces in images
- Mac OS X
  1. MPISearch_QT.app – a Mac OS X application that uses a QuickTime compatible camera for input (e.g., an iSight) and finds faces in the image. Note that the libraries may be built with the boost.build system on Mac OS X but this application is not configured to use them. Rather, it assumes that libmpisearch.framework has been built using the Xcode project. However, in this case, the applications in the Apps/linux directory may be used if ImageMagick and X11 are installed.
- MS Windows
  1. MPT – an executable that allows the user to run face finding, color tracking, eye detection and eye-blink detection on live video.

Note that the default setting for all of these libraries is to be shared libraries on all supported platforms. Some platforms require extra effort to get shared libraries to be easily useful. In any event, Read on for your specific platform

## 2.2 TO BUILD ON LINUX / DARWIN / OTHER UNIXES

Here are the instructions to build everything with Boost.Build. Boost.Build relies on jam/Jamfiles, a modern replacement to make/Makefiles, that ease project compilation across multiple platforms[1]

*Note for Mac OS X Users*: You have an option to build MPT using the general UNIX approach explained in this section or the Mac specific approach explained in Section 2.3. If you need to work across platforms, you may find the UNIX approach useful. If you will use MPT on Macs only, you may find more useful the Mac specific approach . The Mac specific approaches uses Xcode and creates frameworks instead of just dylibs.

1. **Compile the version of bjam used by MPT:** bjam, (BoostJam), is an application needed to build MPT. Note you need to compile and use the version of bjam provided by MPT, for it contains some modifications that were needed to build MPT. These modifications will become standard in future official distributions of bjam. Hereafter we will assume you temporarily located MPT in a directory named *¡MPT¿*.

   ```
   cd ~/MPT/boost-build/jam_src
    ./build.sh
   ```

   Under Linux successful compilation will create the executable *bjam* and place it at

   ```
   <MPT>/boost-build/jam_src/bin.linuxx86
   ```

[1] To learn more about Jam you can take look at the Jam.html file included in the boost-build/jam_src directory of this distribution of MPT.

Under Mac OS X *bjam* will be placed at

```
<MPT>/boost-build/jam_src/bin.macosxppc/bjam
```

2. Add the path of *bjam* to your PATH variable. For example, if you are using *tcsh* under Mac OS X, this would be done as follows

```
setenv PATH ${PATH}:<MPT>/boost-build/jam_src/bin.macosxppc
```

3. **Build MPT:**.
   (a) Modify the file

   ```
   <MPT>/boost-build/user-config.jam
   ```

   For Linux, the file should look as follows

   ```
   #  Mac OS X / darwin configuration
   #   using darwin ;

   #  GCC configuration

   #  Configure gcc (default version)
      using gcc ;
   ```

   For Mac OS X the file should look as follows

   ```
   #  Mac OS X / darwin configuration
      using darwin ;

   #  GCC configuration

   #  Configure gcc (default version)
   #   using gcc ;
   ```

   (b) Go to the root MPT directory

   ```
   cd <MPT>
   ```

   (c) Build MPT using bjam

   ```
   bjam release
   ```

   *Note:* Some targets may fail if you don't have Matlab installed or you don't have ImageMagick. Don't worry, your libraries may have still compiled fine even though boost reports that some targets failed to build.
   (d) Successful compilation creates the directory

   ```
   <MPT>/Libraries/bin
   ```

   and places the following libraries there: libmpisearch, libmputils, libmpeyefind, libmpcolortracker, libmpblinkdetector. *So far only libmpisearch, libmputils, libmpeyefinder, also we need to add the mp to libeyefinder*
   Command-line applications using ImageMagick under Linux or under Mac OS X are placed in

   ```
   <MPT>/Apps/unix/bin
   ```

4. **Run the Applications:**
   (a) Set the environment variables needed for programs to find the shared libs. Under linux:

   ```
   setenv LD_LIBRARY_PATH MPT/Libraries/bin
   ```

   or under Mac OS X

```
setenv DYLD_LIBRARY_PATH MPT/Libraries/bin
```

Alternatively, you could softlink or copy these into /usr/local/lib or /usr/lib. Finally, you could build everything with static linking, like this: bjam release toolset=gcc link=static

(b) Run the applications:
First let's run the eyefinder on an example image.

```
<MPT>/Apps/unix/bin/eyefinder <MPT>/Apps/Media/ExampleImage.jpg
```

This should create a file named "face1.jpg", which contains a face, cropped from ExampleImage.jpg, with the eyes marked.

*Note:* The Apps will only work if you have installed ImageMagick with the Magick++ developer headers included. If you can run the command

```
Magick++-config --version
```

then you probably are good to go. Otherwise, do what you will need to install ImageMagick and make the above command work. You can download ImageMagick from wwww.imagemagick.org

### 2.2.1 MATLAB SUPPORT

To get Matlab support, you have to set the MATLAB environment variable to be the path to the Matlab installation (not just the binary, but everything) before running the bjam release command. e.g.,

```
 setenv MATLAB /Applications/MATLAB7
<MPT>/bjam release
```

After successful build you shall see the mex files at

```
/Users/movellan/MPT/Libraries/mpisearch/matlab
```

They are named *.mexmac under Mac OS X and *.mexglx under Linux.

## 2.3 TO BUILD ON MACINTOSH (Mac OS X 10.3 and up) USING XCODE

The Mac OS X specific build uses Xcode and provides an application that detects faces in real time.

### 2.3.1 Bulding the mpisearch framework

1. Open the following file using Xcode

   ```
   <MPT>/Libraries/mpisearch/mac/mpisearch.xcode
   ```

2. Build the framework (click the Build icon or Build-¿ Build menu. This shall produce the file

   ```
   <MPT>/Libraries/mpisearch/mac/build/mpisearch.framework
   ```

   and the symbolic link

   ```
    /Library/Framework
   ```

   pointing to the previous file.

### 2.3.2 Building the real time video face detection application

1. Open the following file using Xcode:

   `<MPT>/Apps/mac/MPISearch.pbproj project`

2. Build the application. This shall produce the application file

   `<MPT>/Apps/mac/build/MPISearch.app`

   You can run the application by double-clicking it on the Finder, opening it on a terminal window, or running clicking on the Run icon within Xcode. The application requires a Mac OS X compatible video camera (e.g., iSight, Pyro WebCam, Logitech QuickCam, etc ).

## 2.4 TO BUILD ON WINDOWS (VISUAL C++ 6.0)

### 2.4.1 Configuring and installing the MPT environment

1. Ensure that visual studio is not currently open
2. Edit registry to allow visual studio to compile and deal with .cc files
   (a) See http://www.mvps.org/vcfaq/ide/1.htm
   (b) Or go to

   `<MPT>\Libraries\mpisearch\windows`

   and double click cc.reg. Note Visual C++ cannot be open.
3. Download and Install DirectX 9.0 SDK from Microsoft (only needed for filters)
   (a) Download from http://www.microsoft.com/downloads
   (b) Go to

   `<DXSDK>\SAMPLES\C++\DirectShow\BASECLASSES`

   where "¡DXSDK¿" stands for the directory where the DirectX 9.0 SDK was installed[2]. Open *baseclasses.dsw* using Visual C++
   (c) Build the release version. The settings to choose the release version can be found under Build → Set Active Configuration.
4. Add DirectX libs and includes to Visual C++
   (a) Open Visual C++
   (b) Go to Tools → Options → Directories
   (c) Under "Show Directories For" go to "Include Files" and do the following:
      i. Add
      `<DXSDK>\Include`
      and move it to the top of the list.
      ii. Add
      `<DXSDK>\SAMPLES\C++\DIRECTSHOW\BASECLASSES`
      and move it below the top item of the list.
   (d) Under "Show Directories For" go to "Library Files" and do the following:
      i. Add
      `<DXSDK>\lib`
      and move it below the top item in the list.

[2]Note older versions of DirectX the C++ directory will be named MULTIMEDIA

ii. Add
`<DXSDK>\Samples\C++\DirectShow\BaseClasses\Release`
and move it below the second item in the list.

### 2.4.2 Building the MPT Projects

The current version of MPT contains the following Visual C++ projects:

1. Mpisearch: A black and white, real time, frontal face finder using a Viola and Jones style approach.
2. Eyefinder: an eyedetector based on the mpisearch face detector
3. Blinkdetector: Real time blink detection.
4. Colortracker: A very fast face tracker based on simple color features adapted based on information from a face detector running in parallel.
5. MPutil: Utilites used to make code work across multiple platforms

The projects are located on direcctories of the form

`<MPT>\Libraries\project_name\windows`

Hereafter we will describe how to build and run the Colotracker project. The other projects can be build in an analogoous manner.

1. Go to the directory

   `<MPT>\Libraries\colortracker\windows`

   and open "MPColorTrackerApp.dsw" using Visual C++.
2. Under Visual C++ go to the Build *to* "Set Active Configuration" menu and chose the "MPColorTrackerApp Win 32 Release" option.
3. Go back to the "Build" menu and click on "Build MPColorTrackerApp.exe". This will build the colortracker application.

### 2.4.3 Running the Projects

1. Go to directory

   `<MPT>\Libraries\colortracker\windows\bin`

2. Click on CHANGETO_release.cmd. This registers the active X control and copies the appropriate executable into the bin directory.
3. Make sure a camera is connected
4. Run the executable MPColorTrackerApp located in that directory.

## 3 Frequently Asked Questions

### 3.0.4 MS Windows build

1. *Is there example code to reference for the projects?:*
   Yes all projects have a FilterInterface class that is used for the filters to utilize the projects. The class is located at

   `<MPT>\Libraries\project_name\windows\FilterSrc`

2. *The App does not work.*
   This is a basic app that just takes the first camera from a list of WDM cameras. Often there will be drivers for cameras in the cue that are not plugged in. If one of those cameras is higher in the cue the app will try to open the wrong camera. Some higher camera drivers do not register as a WDM camera. To check follow the procedure below:
   (a) Open up GraphEdit Start → Programs → Microsoft DirectX SDK → DirectX Utilities → GraphEdit
   (b) Open Insert Filters under the Graph menu
   (c) Expand Video Capture Source
   (d) If the camera is listed then it is in the cue for WDM cameras

## 4 Research related to the MPT

The following articles report work that used components of the MPT. Please use reference [1, 2, 3, 4, 5, 6, 7, 8, 9]

## 5 History:

- The first version of this document was put together by Ian Fasel on September 2004. It contained 2 pages of build and installation instructions.
- Javier R. Movellan designed the "Empty" logo, and included it on the first latex version of the document.

## References


[1] M. Bartlett, G. Littlewort, I. Fasel, and J. Movellan. Real time face detection and expression recognition: Development and application to human-computer interaction. In *CVPR Workshop on Computer Vision and Pattern Recognition for Human-Computer Interaction*. 2003.

[2] M. Bartlett, G. Littlewort, C. Lainscsek, I. Fasel, and J. Movellan. Machine learning methods for fully automatic recognition of facial expressions and facial actions. In *IEEE International Conference on Systems, Man & Cybernetics*, The Hague, Netherlands, October 2004.

[3] M. S. Bartlett, J. R. Movellan, G. Littlewort, B. Braathen, F. M. G., and T. J. Sejnowski. Towards automatic recognition of spontaneous facial actions. In P. Ekman, editor, *What the Face Reveals*. Oxford University Press, 2003.

[4] I. Fasel, B. Fortenberry, and J. R. Movellan. A generative framework for real-time object detection and classification. *Computer Vision and Image Understanding*, In Press.

[5] T. Kanda, N. Miralles, M. Shiomi, T. Miyashita, I. Fasel, J. R. Movellan, and H. Ishiguro. Face-to-face interactive humanoid robot. *IEEE 2004 International Conference on Robotics and Automation*, in press.

[6] G. Littlewort, M. Bartlett, I. Fasel, J. Susskind, and J. Movellan. Dynamics of facial expression extracted automatically from video. In *IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Face Processing in Video*, 2004.

[7] G. Littlewort, M. Bartlett, C. J, I. Fasel, T. Kanda, H. Ishiguro, and J. Movellan. Towards social robots: Automatic evaluation of human-robot interaction by

face detection and expression classification. In *Advances in neural information processing systems*, volume 16. MIT Press, Cambridge, MA, in press.

[8] J. R. Movellan and M. S. Bartlett. The next generation of automatic facial expression measurement. In P. Ekman, editor, *What the Face Reveals*. Oxford University Press, 2003.

[9] J. R. Movellan, J. Hershey, and J. Susskind. Large scale convolutional HMMs for real time video tracking. *Computer Vision and Pattern Recognition 2004*, 2004.