MicronTracker Developer's Manual

MTC 3.6

February, 2011

WARNING: This document and the associated demonstration application source code, in electronic or printed form, are proprietary and confidential to Claron Technology, Inc. No part of this document, the detailed programming interface documentation and the associated example source code, shall be copied, published or used for a purpose other than product usage without the prior written consent of Claron Technology, Inc.

Table of Contents

1 Introduction

1.1 Purpose of this Manual

The purpose of this manual is to assist you in effectively using your new MicronTracker product. In particular, you will learn how to:

- Install the MicronTracker software on a workstation or a laptop.
- Design markers optimized for use with the MicronTracker.
- Plan the integration of MicronTracker's software into your application.
- Optimize the performance of the MicronTracker to better fit the requirements of your application.

A detailed description of the programming interface is separately provided in the MTC programming manual, in HTML format.

1.2 Third-Generation Optical Tracking

MicronTracker is a family of optical pose tracking products, all sharing a common software interface, the MTC library, but differing in the properties of their camera.

The two key functions of optical pose trackers are:

- **Detection**: detect the presence of, i.e., identify, specially marked objects in the sensor's field of measurement (FOM).
- **Pose measurement**: report on the location and orientation of each detected object.

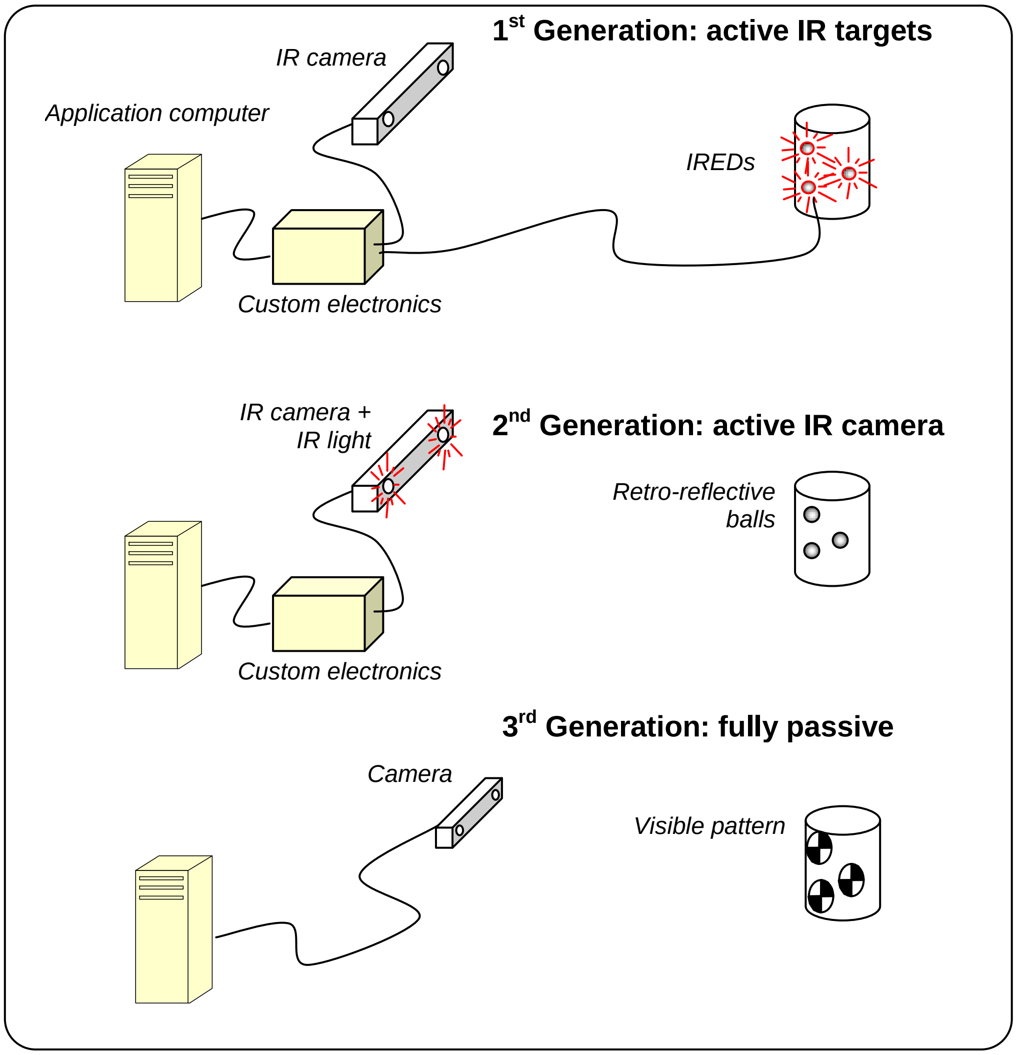
Both functions need to be performed repeatedly at real-time rates (10Hz or faster).

Optical pose trackers perform these functions by observing marked locations, or **targets**, on each tracked object from multiple angles of view, then triangulating the lines of sight to the targets to calculate each target's' location. The 3D locations of at least 3 targets are needed to calculate the object's pose, i.e., its position and orientation in space relative to the camera. Pose sensing is sometimes referred to as 6D measurement, since at least 6 parameters are required to fully describe a pose: 3 space coordinates (X, Y, Z) and 3 orientation angles (e.g., azimuth, elevation and roll).

MicronTracker is the first **third generation**[1] optical pose tracking product (see illustration). Optical pose sensors of the first generation use light-emitting targets, such as LEDs (light emitting diodes). Second-generation sensors emit light from a ring surrounding each camera lens and use balls or discs as targets. The balls or discs are coated with a retro-reflective material that mirrors the light back to the lenses. Both first and second generation trackers require the targets to be observed as bright spots over a dark background. Since this cannot be guaranteed under visible light conditions, Infra-Red (IR) light is used. The intensity of the light reaching the camera falls with the square of the distance between the target and

[1] Claron Technology Inc. holds patents on key enabling technological innovations related to 3rd generation optical tracking.

the camera. To prevent sensor saturation, first and second generation trackers do not allow targets to be placed close to the camera. Placing the camera far from the



FOM means that a larger camera is needed to obtain the required accuracy. It also increases the likelihood of line-of-sight interruptions and makes it more difficult to optimally position the camera, especially when space is at a premium.

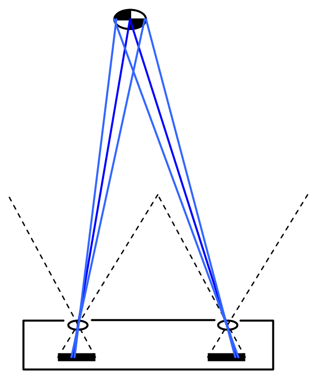
Third generation trackers are fully passive. They use available visible light illumination to observe targets stereoscopically much like humans do. Tracked objects are marked with a visible painted or printed target pattern. The camera is interfaced to the computer using a standard digital link (e.g., FireWire or USB 2.0). Software running on the PC processes the camera images to detect and pinpoint visible target patterns in the images. The MicronTracker's target pattern is a set of high-contrast regions called a **Marker.** The regions forming the marker are geometrically defined and arranged according to specific rules dictated by the tracker's designers.

Third generation trackers offer many advantages over trackers of earlier generations, amongst them:

- Simpler, more reliable, lower-cost, design, which uses off-the-shelf electronics;
- Smaller and lighter camera that can be placed closer to the FOM, easier to ship, move, store and manipulate;
- Lower cost markers, which can be easily produced or durably painted on objects;
- Better immunity against measurement errors resulting from partial occlusion of targets;
- More reliable identification of marker patterns;
- Practically unlimited number of objects tracked concurrently;
- New target patterns can be easily learned by the software in the field;
- Great flexibility in optimizing the designs of the cameras and target patterns to fit specific application requirements;
- Easy product upgrades in the field through simple downloadable software updates;
- Application software access to camera images for easier set up, augmented reality display, documentation and remote diagnostics;
- Support for multi-camera configurations, used to eliminate line-of-sight interruptions or expand the FOM.

Over the next few years, the third generation trackers will undergo rapid evolution. In particular, expect to see markers becoming smaller and less intrusive, accuracy and measurement rates improve, and many additional software features added to allow configuring and customizing trackers to the specific needs of users in a wide range of applications.

1.3 Principles of Operation



Detection of markers is achieved by processing the images of the two sensors, left and right, to detect and correlate visual patterns matching the descriptors in the marker templates database. Each target point in a marker has to be correctly recognized in each of the two images. This requires that the checkerboard intersection region cast a minimum projection "footprint" in each image. When the smallest target region projection width falls below the minimum required footprint diameter (currently configurable to 11 or 9 pixels), the target is not detected. Camera models with higher sensor resolution or longer lenses (narrower field of view) generate larger projection image footprints, and are, therefore, able to recognized smaller targets, or targets at larger distances. Tilting a target region away from facing the camera reduces the width of its footprint proportionally to the tilt angle's cosine.

Once a marker pattern has been identified in the images, the exact 3D position of its target points is calculated by triangulating the two projection lines associated with the two image positions in which the target center is observed (see diagram). These projection lines are obtained from a calibration file, computed from data collected in a series of experiments carried out before each camera is shipped from the factory.

The pose of each marker is obtained from measuring the individual 3D positions of its target points, then calculating the coordinate space transformation that best matches the template coordinates of each target to the camera-space coordinates measured, i.e., resulting in the minimum RMS error between measured and transformed locations.

1.4 Terminology and Conventions

Unfortunately, when it comes to optical tracking, there is no widely-accepted terminology. This is especially true when it comes to a new generation of products. To reduce confusion, this section defines the terms being used in the MicronTracker development environment.

1.4.1 Data Capture

(Stereo) Camera*:* The stereo camera assembly.

Lens/sensor: The optical/electronic part of the camera involved in producing images.

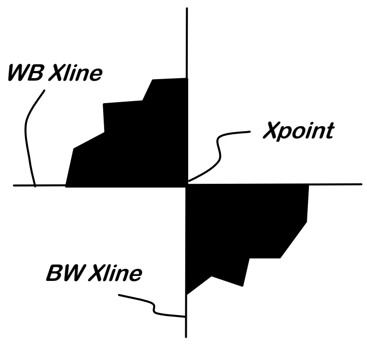
Left/Right Image: The pixels buffer acquired from each corresponding lens/sensor pair.

(Stereo) Frame: A pair of Left and Right images captured simultaneously by the same camera.

1.4.2 Object marking

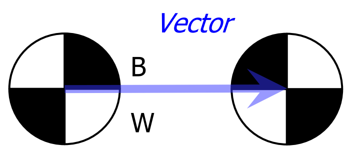
Target: a single physical location whose position in 3-space is being marked and measured. This is a general term used in optical metrology, usually referring to the center of retro-reflective discs or balls. MicronTracker products use Xpoints to mark targets (see next).

Xpoint (or XP): one checkerboard intersection painted on a flat surface. The term is also used to refer to the appearance of such a pattern in an image.

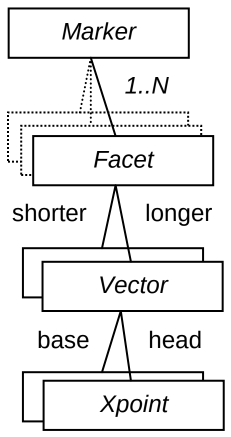


Xlines: The two imaginary straight lines crossing at the Xpoint's center. Viewing the Xlines from the Xpoint center outwards, **BW** Xline is the one with the black on the left and white on the right (hence the letter order), and **WB** Xline is the other. The Xlines are usually designed to be perpendicular to each other, but this is not a requirement.

Xnormal: A unit vector perpendicular to both Xlines, on the painted side of the Xpoint's surface (see the multi-facet marker figure on next page).



Vector: Two Xpoints arranged such that one's BW Xline is aligned (co-linear) with the other's WB Xline. The Xpoint at the BW end is called the vector's **base**, and the other is the vector's **head**. Although not an absolute restriction, the Xnormals of the XPs at the ends of a vector would usually be identical (or at least very similar). Vectors in the FOM are detected and tracked, even when they are not identified as being part of a facet.

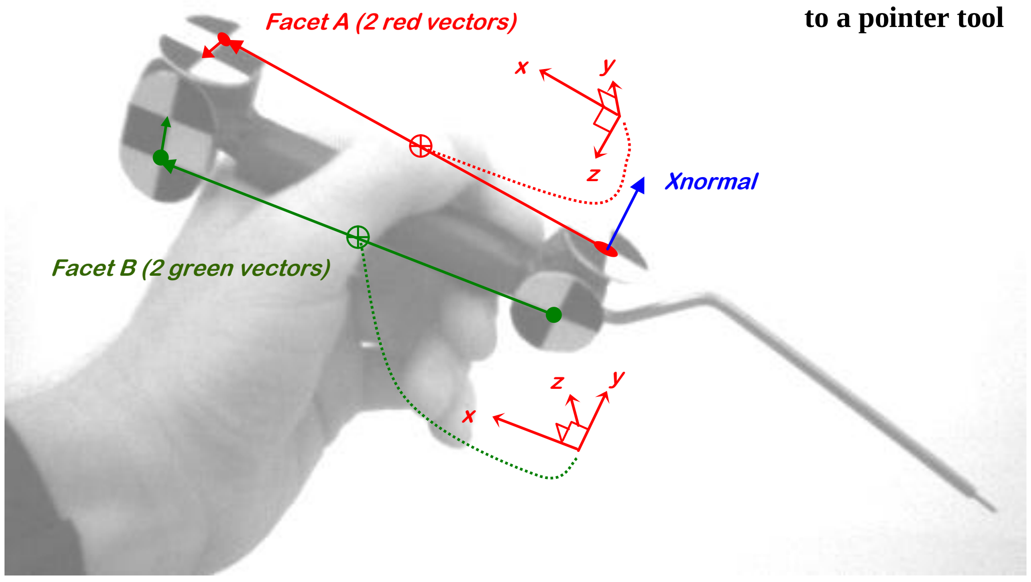


Facet: Two vectors in a rigid configuration that complies with MicronTracker's facet restrictions (one vector is longer than the other, and the angle between the vectors is in the 8-172 degrees range, see 9.5). The vectors would usually be on the same plane, or at least have parallel Xnormals, but this is not a requirement. A facet, containing 3 or 4 Xpoints, is the smallest set of Xpoints that can be tracked in 6D (3 spatial coordinates and 3 axes of rotation).

Marker: One or more facets in a rigid configuration, usually attached to an object to be tracked. In a multi-facet marker, only one of its facets needs to be observed in a frame to allow MicronTracker's software to detect and report on the marker's pose.

Marker Template: A named description of a marker's geometry, enabling its detection and pose measurement. Marker templates are stored on disk as text files.

A Multi-facet marker attached to a pointer tool



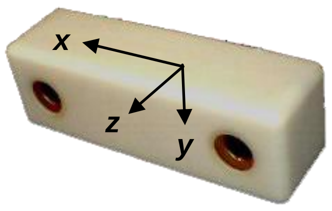
1.4.3 Coordinate systems

All coordinate systems are right-handed. Locations and distances are reported in millimeter units.

Facet coordinates: Each facet has its own coordinate system. The origin is at the middle of the longer vector. The X-axis extends from the origin towards the head of the longer vector. The Z-axis is perpendicular to both vectors, pointing in the direction opposite to the facet's Xnormals, such that it is aligned with the camera's Z-axis (depth direction) when the facet is facing the camera. The Y-axis is defined according to the right-handed coordinate system's rules (see above figure for examples).

Marker coordinates: Identical to the coordinates of the first facet described in the marker's template (even when that facet is not visible).

Camera coordinates: Defined during calibration. The origin is approximately at the center of the camera. The Z-axis extends to the front of the camera (depth), X towards the right of the camera and Y down. When viewing the camera's images, the camera and the screen coordinate systems are aligned (X right, Y down and Z in).



NOTE: *The camera coordinate axes orientations are not accurately aligned with the orientation of the enclosure's external faces, and the FOM is not accurately symmetric around each axis*.

1.5 Contact Us

As is typical for such innovative systems at the beginning of their evolution path, MicronTracker is undergoing rapid change. It is quite likely that the environment in which you will be using your MicronTracker is different from any of the environments in which it was tested in the past, and, thus, you may encounter some difficulties or have new requirements that have not been fully addressed yet. We, at Claron Technology, are keenly interested in your experience evaluating the device and embedding it in your application. We commit to being highly responsive to any feedback we receive.

Our contact coordinates:
Claron Technology, Inc.,
120 Carlton Street, Suite 217
Toronto, Ontario, Canada, M5A 4K2
Tel: +1.416.673-8175, Fax +1.416.673-8174
www.clarontech.com
support@clarontech.com

2 Windows Installation

2.1 MicronTracker parts list

The MicronTracker package contains:

- One MicronTracker camera
- One IEEE 1394/FireWire PCI Host Card (Supports serial bus data transfer rate of 100/200/400Mbps and 800Mbps for H3-60)
- One 4.5m 6 pins to 6 pins IEEE 1394 (FireWire) cable only for H40, H60, S60,Sx60, Hx40 and Hx60 models
- One 4.5m 9 pins to 9 pins FireWire 800 IEEE 1394b cable only for H3-60 model and one 4.5m 9-pin to 6-pin FireWire 800.
- One General Purpose Input Output (GPIO) connector (Only Hx40, Hx60, Sx60 and H3-60 models)
- The MicronTracker SDK CD
- A board demonstrating two markers, one of them supporting tooltip calibration
- A polished aluminum bar (AccBar) to be used with R-Fine software for verification and refinement of the calibration
- A special CoolCard marker, made of the Duramark material, for adjusting the light coolness camera setting
- Printed sheets showing examples of marker patterns

If your package does not contain the any of the above items, please contact Claron Technology support.

2.2 PC Host Configuration

To be able to install and properly run the MicronTracker software, the following system configuration is required:

- MS-Windows 2000 or XP
- 2GHz or faster Pentium processor (3GHz recommended)
- 512 MB or more RAM
- 500 MB of available hard disk space
- An available IEEE 1394 (FireWire) socket or an available PCI slot for the IEEE 1394 card provided with the MicronTracker.

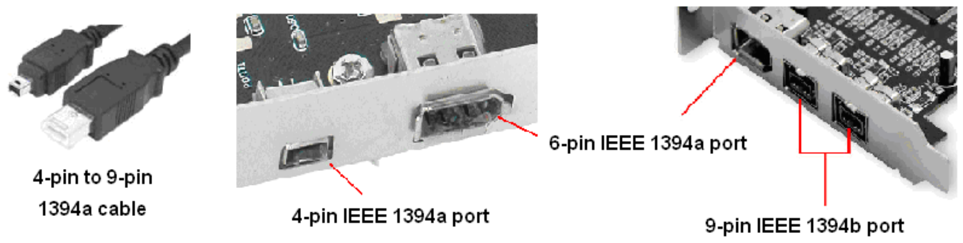
To be able to run and modify the source code of the demonstration programs:

- Display of 1024x768 or higher resolution
- 500 MB of hard disk space and 512 MB of memory
- Visual Basic 6, C++ or Microsoft .Net development environment

2.3 Enabling the IEEE 1394 Link

2.3.1 IEEE 1394 (FireWire) connectors

The MicronTracker camera communicates with the PC host using an IEEE 1394 (also known as FireWire or i.link) digital link. IEEE 1394 is a fast serial link commonly used to connect consumer digital video cameras to computers. There are three types of IEEE 1394 connectors: a wide 6-pin, narrow 4-pin and IEEE 1394b 9-pin. The three types of connectors and their matching sockets are shown below.



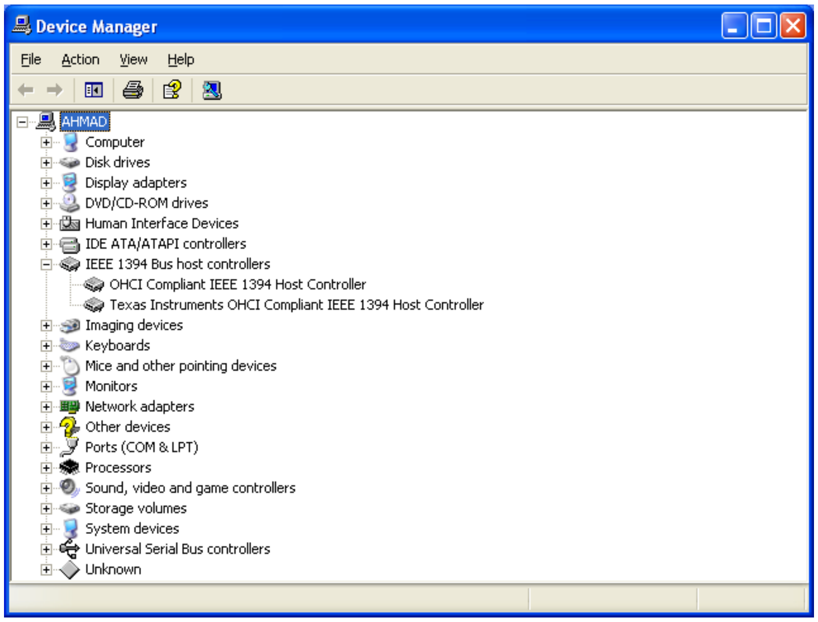
MicronTracker is shipped with a 6-pin to 6-pin or 9-pin to 9-pin and 6-pin to 9-pin cable (depending on the model) and an IEEE 1394a PCI board (which has two 6-pin IEEE 1394 ports and one 4-pin IEEE 1394 port) or IEEE 1394b PCI board (which has two 9-pin IEEE 1394b ports and one 6-pin IEEE 1394a port) to use with stationary PCs that do not already have a proper socket available.

2.3.2 Connecting the camera to an unpowered socket

The MicronTracker camera obtains its power through the IEEE 1394 cable. The 4-pin socket, which is commonly found on laptops, and some 6-pin sockets on mobile computers, does not provide power. A powered IEEE 1394 hub or repeater is, therefore, needed between the camera and the laptop.

NOTE: *If the hub you use does not come with its own power adapter, ensure that you use an adapter with enough power to enable the MicronTracker to function properly (8-30VDC, 2.5W per camera).*

2.3.3 Installing the IEEE 1394 PCI Board



```
AHMAD
  Computer
  Disk drives
  Display adapters
  DVD/CD-ROM drives
  Human Interface Devices
  IDE ATA/ATAPI controllers
  IEEE 1394 Bus host controllers
    OHCI Compliant IEEE 1394 Host Controller
    Texas Instruments OHCI Compliant IEEE 1394 Host Controller
  Imaging devices
  Keyboards
  Mice and other pointing devices
  Monitors
  Network adapters
  Other devices
  Ports (COM & LPT)
  Processors
  Sound, video and game controllers
  Storage volumes
  System devices
  Universal Serial Bus controllers
  Unknown
```

Figure 1: The *Device Manager* dialog, showing that the computer detected the 1394 bus controller and the PCI card is correctly installed.

To install the IEEE 1394 OHCI PCI host adapter 3-port 400 MB/sec you will need complete the following steps:

1. Make sure that the computer is turned off.

2. Remove the side panel from the computer.

3. Place the IEEE 1394 PCI card in an open PCI slot, making sure that it is firmly seated.

4. Replace the side panel on the computer.

5. Turn the computer on.

6. The screen will show a *New Hardware Found* dialog box. Then an *Add New Hardware Wizard* box will appear. Follow the prompts given by the Wizard.

7. To ensure that the PCI card is properly installed, open the Windows Control Panel, click on the *System* icon, then the *Hardware* tab and finally select *Device Manager*. The presence of the *IEEE 1394 Bus host controller* entry as shown in fig. 1 indicates that the PCI card has successfully been installed.

2.4 Removing an older installation

When upgrading an existing MicronTracker installation with a newer version, first it is recommended to uninstall existing cameras. With the camera connected, open the Device Manager and look under Imaging Devices for a device named "PGR Bumblebee". Right click on the device and select uninstall. After uninstalling the device, unplug the camera. Proceed to uninstall the previous MicronTracker software version by navigating to Control Panel → Add or Remove Programs → MicronTracker.

Since the device driver may have been loaded in memory, it is recommended to reboot the machine before starting to use the new MicronTracker version.

2.5 Internet downloadable upgrades

When upgrading an existing MicronTracker installation (version 2.0 or higher) with an installer downloaded from the Internet, it is not required to uninstall the previous version. Simply follows the four steps of the installer in sequence, as detailed below. When installing the MicronTracker package, select the "Repair" option when asked. This different procedure is necessary since the camera specific calibration file is not included in the downloadable upgrade, and therefore the existing installation needs to be already in place. Alternatively the camera calibration file (BumbleBee_xxxxxx.calib, where xxxxxx represent the serial number) can be copied in the folder [*INSTALLER_ROOT*]\ *Claron Technology\MicronTracker\CalibrationFiles* to transform the update installer in a complete camera specific installer.

2.6 Installing the MicronTracker Camera Device Drivers and SDK

Once you have the IEEE 1394 6-pin powered connector ready, you need to run the MicronTracker Installer available on the MicronTracker CD. It will install the necessary camera drivers, a set of complementary libraries from Point Grey Research, the MicronTracker interface software and several demonstration applications and their source code.

<u>NOTE</u>: *Since the procedure involves installing device drivers, the user performing the installation needs to have administrative privileges. Otherwise, the installation may fail.*

<u>NOTE</u>: *Do not connect the camera to your computer until you have completed the software installation steps*.

2.6.1 Installer Steps

The installer provided on the MicronTracker CD-ROM includes some basic installation instructions, and references this document (available on the CD-ROM before the installation is performed) for any further information.

The MicronTracker distribution contains Digiclops and FlyCapture SDKs, libraries developed by Point Grey Research (see *www.ptgrey.com/products/index.html* for details). Their installation is required for MicronTracker to function properly.

A basic installation or an upgrade should go through the three steps of installing the MicronTracker, Digiclops and FlyCapture and in this specified order (fig. 2).

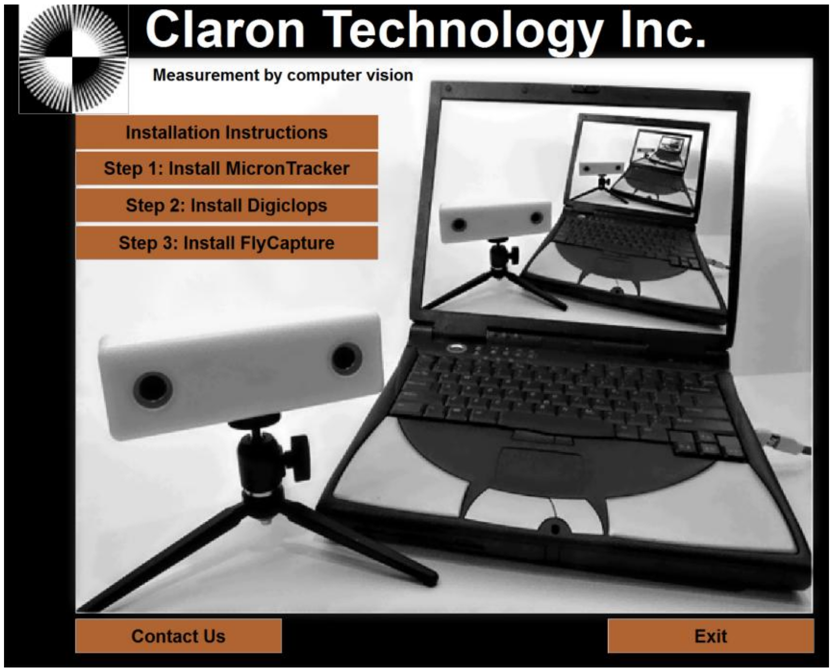

Claron Technology Inc.
Measurement by computer vision
Installation Instructions
Step 1: Install MicronTracker
Step 2: Install Digiclops
Step 3: Install FlyCapture
Contact Us
Exit

Figure 2: The Micron Tracker installation interface, there are three steps to install MicronTracker

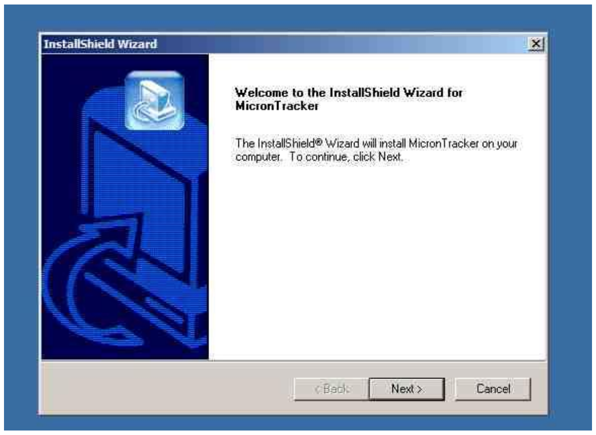
Digiclops and FlyCapture SDKs are the low level video libraries and they also install all the required drivers.

Simply click on the appropriate menus on the main installer and then follow the instructions of the individual installers.

<u>NOTE</u>: *If you plan to experiment with development using Digiclops and FlyCapture, PointGrey recommends accepting the default installation location, since otherwise the example programs project files may need to be modified.*

2.6.2 Installing MicronTracker

Click on the menu "Install MicronTracker".

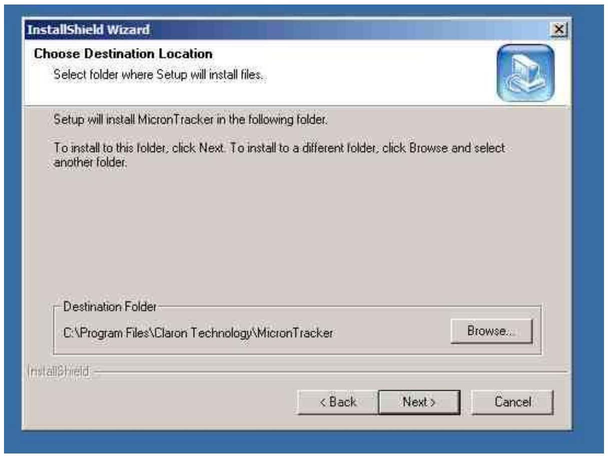


Welcome to the InstallShield Wizard for MicronTracker

The InstallShield® Wizard will install MicronTracker on your computer. To continue, click Next.

Figure 3: The installShield Wizard for MicronTracker

After the *Welcome* screen a dialog to select where the application folder should be located is presented to the user.


InstallShield Wizard

Choose Destination Location

Select folder where Setup will install files.

Setup will install MicronTracker in the following folder.

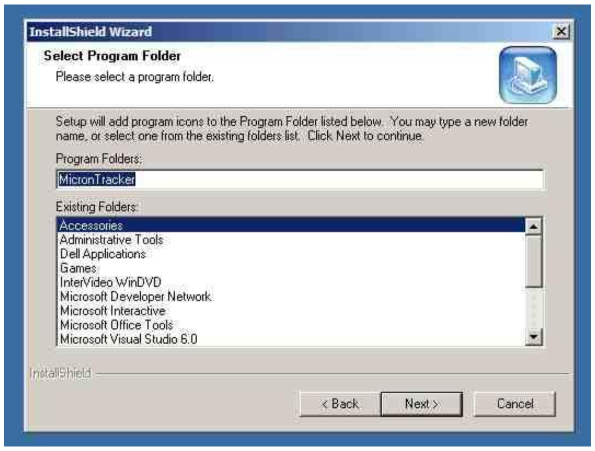
To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

Destination Folder

C:\Program Files\Claron Technology\MicronTracker

Figure 4: Select the MicronTracker's installation directory

Then a dialog to select the program folder will be presented.


InstallShield Wizard

Select Program Folder

Please select a program folder.

Setup will add program icons to the Program Folder listed below. You may type a new folder name, or select one from the existing folders list. Click Next to continue.

Program Folders:

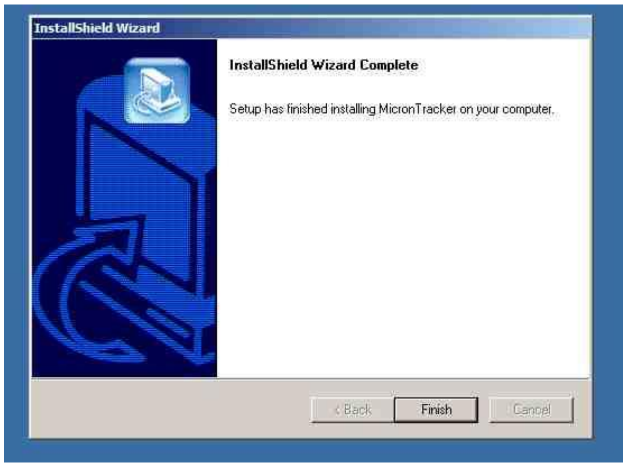
MicronTracker

Existing Folders:

Accessories
Administrative Tools
Dell Applications
Games
InterVideo WinDVD
Microsoft Developer Network
Microsoft Interactive
Microsoft Office Tools
Microsoft Visual Studio 6.0

Figure 5: Select the MicronTracker's program folder

After that, the installer will start transferring files. When terminated, the installer will display the *Wizard Complete* dialog (fig. 6).



InstallShield Wizard Complete

Setup has finished installing MicronTracker on your computer.

Figure 6: MicronTracker has been installed successfully

2.6.3 Connecting the Camera

To connect the camera to the computer, perform the following steps:

1. Plug one end of the cable into one of the sockets on the newly installed PCI card or IEEE 1394 hub (it does not matter which port you plug it into).

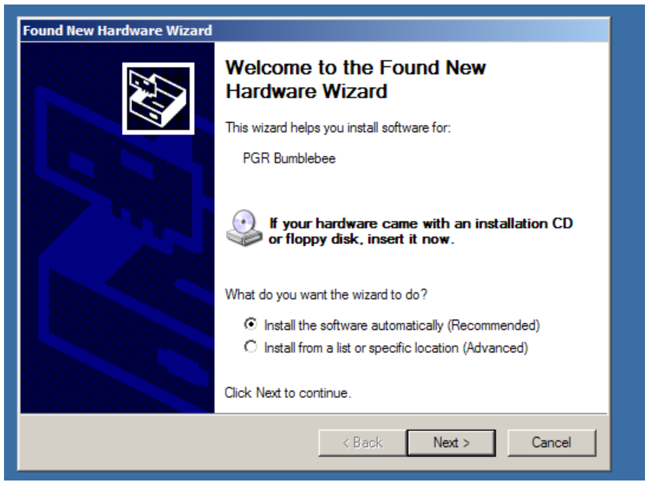
2. Plug the other end of the cable into the socket on the back of the MicronTracker camera.

When the camera is connected and powered, the green LED at the front of the MicronTracker camera will glow. You may need to wait a minute or two before being prompted to install the camera driver.

2.6.4 Installing the Camera Driver

When you plug the MicronTracker camera into an available IEEE-1394 port for the first time, you will be automatically presented with the following dialogs to install the camera driver.

When presented with the *Welcome to the Found New Hardware Wizard* dialog click *Next*.



Welcome to the Found New Hardware Wizard

This wizard helps you install software for:

PGR Bumblebee

If your hardware came with an installation CD or floppy disk, insert it now.

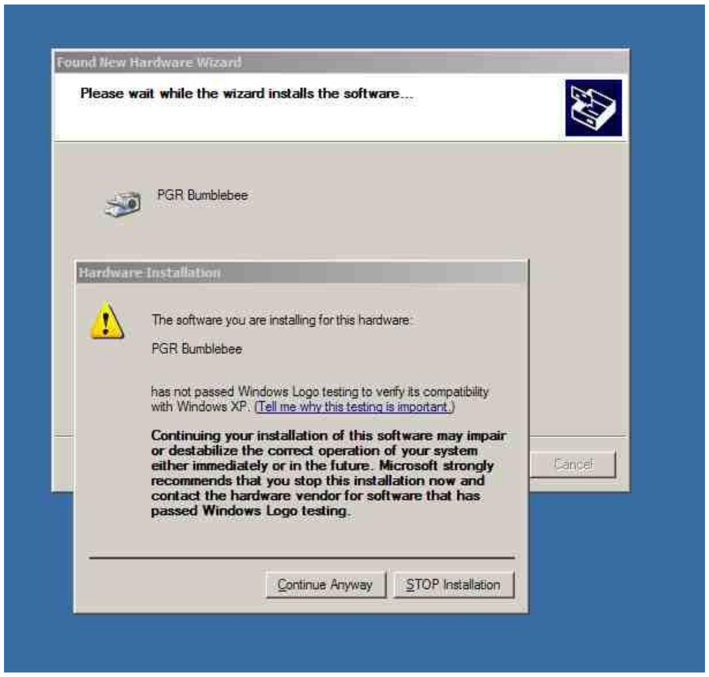
What do you want the wizard to do?

(●) Install the software automatically (Recommended)
( ) Install from a list or specific location (Advanced)

Click Next to continue.

Figure 7: *Found New Hardware* Dialog box, click *Next* to continue

In Windows XP, you will then be presented with a Warning dialog informing you that the camera driver is not Microsoft certified. Click *"Continue Anyway"* and ignore the warning. (fig. 8).



Found New Hardware Wizard

Please wait while the wizard installs the software...

PGR Bumblebee

Hardware Installation

The software you are installing for this hardware:

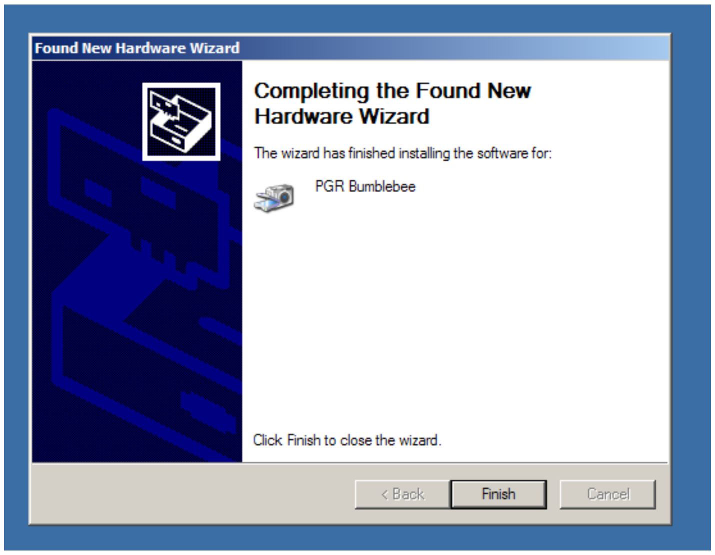
PGR Bumblebee

has not passed Windows Logo testing to verify its compatibility with Windows XP. (Tell me why this testing is important.)

Continuing your installation of this software may impair or destabilize the correct operation of your system either immediately or in the future. Microsoft strongly recommends that you stop this installation now and contact the hardware vendor for software that has passed Windows Logo testing.

Figure 8: *Microsoft uncertified driver warning, just ignore it and click Continue Anyway*

Complete the process by clicking *Finish* on the *Completing Found New Hardware Wizard* dialog.



Found New Hardware Wizard

Completing the Found New Hardware Wizard

The wizard has finished installing the software for:

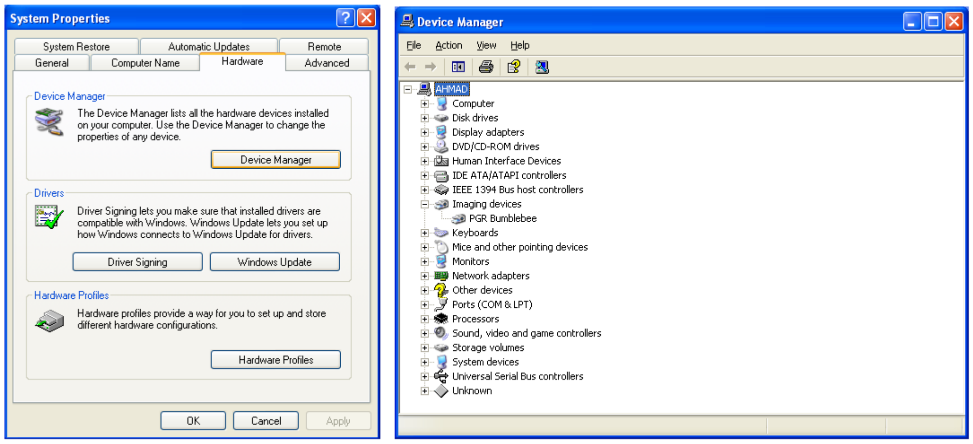
PGR Bumblebee

Click Finish to close the wizard.

Figure 9: The camera driver has been installed successfully

2.7 Installation Verification

After the MicronTracker installation is complete, open the *Device Manager* and verify that under *Imaging Devices* a device named "PGR Bumblebee" is listed.



Device Manager

The Device Manager lists all the hardware devices installed on your computer. Use the Device Manager to change the properties of any device.

Drivers

Driver Signing lets you make sure that installed drivers are compatible with Windows. Windows Update lets you set up how Windows connects to Windows Update for drivers.

Hardware Profiles

Hardware profiles provide a way for you to set up and store different hardware configurations.

- AHMAD
  - Computer
  - Disk drives
  - Display adapters
  - DVD/CD-ROM drives
  - Human Interface Devices
  - IDE ATA/ATAPI controllers
  - IEEE 1394 Bus host controllers
  - Imaging devices
    - PGR Bumblebee
  - Keyboards
  - Mice and other pointing devices
  - Monitors
  - Network adapters
  - Other devices
  - Ports (COM & LPT)
  - Processors
  - Sound, video and game controllers
  - Storage volumes
  - System devices
  - Universal Serial Bus controllers
  - Unknown

Figure 10: Device Manager, the imaging device has been installed properly

3 Linux Installation

3.1 PC Host Configuration

To be able to install and properly run the MicronTracker software, the following system configuration is required:

- A Linux distribution with kernel 2.4 or higher including gcc 3.2 or higher. Previous versions of the kernel had inconsistent FireWire support.
- 2 GHz or faster Pentium processor (3GHz recommended)
- 512 MB or more RAM
- 200MB of available hard disk space
- An available IEEE 1394 (FireWire) socket or an available PCI slot for the IEEE 1394 card provided with the MicronTracker.
- Display of 1024x768 or higher resolution

3.2 Linux Installation

The MicronTracker SDK on Linux is distributed as a tgz file in conjunction with the Digiclops and Triclops tgz files. The MicronTracker SDK also contains a multi-platform demo. This demo is not as complete as the Windows VB demo, but it shows how to perform the fundamental MicronTracker tasks:

- Loading pre-existing marker templates,
- Visually generating new templates,
- Recognizing markers,
- Displaying positions and distances between markers.

The UI is has been implemented using FLTK (www.fltk.org) a public domain multi platform toolkit. The video and text display is done using OpenGl or FLTK depending on configuration parameters. Note that the display of images and overlays is just an example implementation. Cross-platform compatibility was given the highest priority.

3.2.1 Dependencies

NOTE: *to operate the MicronTracker the ieee1394, raw1394, video1394 modules should be loaded.*

You can check if the module is loaded by issuing the command:

- /sbin/lsmod

If necessary you can then load the modules with:

- /sbin/modprobe ieee1394
- /sbin/modprobe ohci1394
- /sbin/modprobe video1394

- /sbin/modprobe raw1394

Note that this should be done as root, or appropriate permissions should be setup.

It is possible to load the module on demand (i.e. when anything is plugged in the /dev/raw1394 device), by having the right entries in the /etc/modules.conf file.

The relevant lines are:

- alias ieee1394-controller ohci1394
- alias char-major-171 raw1394
- options ohci1394 attempt_root=1
- above ohci1394 raw1394
- above ohci1394 video1394

Make sure the raw1394 and the video1394 device files exist on the system under /dev. If not create them with the following command.

raw1394:

```
mknod -m 666 /dev/raw1394 c 171 0
```

video1394:

```
mknod /dev/video1394/0 c 171 16
```

In case of multiple camera and buses, create a FireWire bus device (one for each bus) in the dev folder

```
mkdir /dev/video1394
mknod /dev/video1394/0 c 171 16
mknod /dev/video1394/1 c 171 17
...
```

To test if raw1394 module is installed properly, run:

```
/testlibraw
```

It should list all the buses and the connected cameras.

If testlibraw gives no error, you can add those scripts to the login process. This will load the modules automatically every time you login. To load the modules automatically login as root, edit the file /etc/rc.d/rc.local and append the following commands to the file:

```
/sbin/modprobe ieee1394
/sbin/modprobe ohci1394
/sbin/modprobe video1394
/sbin/modprobe raw1394
mknod -m 666 /dev/raw1394 c 171 0
mkdir /dev/video1394
mknod /dev/video1394/0 c 171 16
mknod /dev/video1394/1 c 171 17
...
```

NOTE: *The commands to be added depend on your specific HW configuration, as the number of FireWire buses currently installed.*

Please pay attention to avoid modifying any other parts of the rc.local file, if you are not knowledgeable of the Linux internals, or you may be unable to start up again your workstation.

The Linux release comes in the form of a gzip tar archive. After extracting the sources and libraries to the filesystem you need to:

- Set the MTHome environment variable to where you have extracted the tarball. An example environment.sh file is included for your convenience.
- You will also find an empty CalibrationFiles directory at the higher level where you extract the tar file. You need to copy the calibration file of your camera from the CD into this directory. The directory where you'll find it on the CD is:

  [*CD_ROOT*]*\Claron\CalibrationFiles*

  And the file will be named:

  BumbleBee_xxxxxx.calib

  Where xxxxxx will be the serial number of your MicronTracker.

The release includes a precompiled demo in the directory MTDemoCPP, but you can invoke make at the higher level to compile everything from scratch.

Complete releases of the PointGrey toolkits Digiclops and Triclops are included in the Tarballs directory, together with the full release of FLTK. They are not required for using MicronTracker, but you may find them useful when experimenting with other uses for the stereo video input, such as augmented reality.

4 Mac OSX Installation

4.1 PC Host Configuration

To be able to install and properly run the MicronTracker software, the following system configuration is required:

- A Mac OSX 10.4.11 distribution with Xcode version 2.4.1 or higher including gcc 3.2 or higher.
- 2 GHz or faster Pentium processor
- 512 MB or more RAM
- 200MB of available hard disk space
- An available IEEE 1394 (FireWire) socket or an available PCI slot for the IEEE 1394 card provided with the MicronTracker.
- Display of 1024x768 or higher resolution

4.2 Mac OSX Installation

The Mac release comes in the form of a package file and has been tested on Mac OSX 10.4.11. After extracting the source and libraries you need to:

- Set the MTHome environment variable to where the package has been extracted. An example environment.plist file is included for your convenience.
- You will also find an empty CalibrationFiles directory at the higher level where you extract the package file. You need to copy the calibration file of your camera from the CD into this directory. The directory where you'll find it on the CD is:

  [CD_ROOT]\Claron\CalibrationFiles

  and the file will be named:

  BumbleBee_xxxxxx.calib

  FlyCapture_xxxxxx.calib

  where xxxxxx will be the serial number of your MicronTracker.
- The release includes a precompiled demo in the directory MTDemoCPP, but you can invoke make at the higher level to compile everything from scratch.
- The release also includes a precompiled Xcode project of the MTDemoCPP project.

NOTE:

- MTDemoCPP invokes MTHome; an environment variable; to load the calibrationfile and the Markers file.
- The Xcode sample project has been tested on Xcode version 2.4.1. If you are using different versions make sure that all of the required dependencies are assigned properly.

5 Product Overview

5.1 Camera Models and Specifications

MicronTracker cameras are customized versions of PointGrey's class-leading Bumblebee stereo cameras. Five models are currently offered, H3-60, Sx60, S60, Hx40 and Hx60 (Hx40 and Hx60 are developed as a drop-in replacement for the original H40 and H60 models and feature higher frame rate and GPIO connector for external trigger and strobe functionality). The following table describes and compares their key characteristics.

Specification \ Model	H3-60	Sx60	S60	H40/Hx40	H60/Hx60
FOM, spherical section (radius x width x height)	240x200x160 cm	115x70x55 cm	115x70x55 cm	120x120x90 cm	200x130x100 cm
Measurement rate	16 Hz	48 Hz	30 Hz	15Hz / 20Hz	15Hz / 20Hz
Calibration accuracy[1]	0.20 mm RMS	0.25 mm RMS	0.25 mm RMS	0.20 mm RMS	0.35 mm RMS
Jitter, static target[2]	0.007mm RMS	0.007mm RMS	0.007mm RMS	0.015mm RMS	0.015mm RMS
Jitter, moving target[2]	0.07 mm RMS	0.07 mm RMS	0.07 mm RMS	0.14 mm RMS	0.14 mm RMS
Processing time/frame (3GHz CPU, 6 markers)	30~35 ms	10-12 ms	10-12 ms	15-20 ms	15-20 ms
Lag	~100 ms	~30 ms	~45 ms	~85 ms	~85 ms
Sensors resolution	1280x960	640x480	640x480	1024x768	1024x768
Lenses / H°xV°	6 mm 50°x38°	6 mm 50°x38°	6 mm 50°x38°	4 mm 70°x52°	6 mm 50°x38°
Case dimensions (WxHxD, approx.)	283x43x49 mm	164x43x54 mm	172x57x57 mm	172x57x50/ 164x43x54 mm	172x57x57/ 164x43x54 mm
Weight (approx.)	505gr	342gr	790gr	675gr/342gr	790gr/342gr
Electrical Interface	IEEE-1394b (FireWire), 800 Mbps	IEEE-1394a (FireWire), 400 Mbps			
Mount	1/4" thread tripod mount				
Operating temperature	15-35°C (59-95°F)				
Warm-up period	~30 mins	~15 mins			
Warm-up period Template-based correction enabled	~10secs	~10secs			
Max. # of marker templates	practically unlimited				
Max. # of markers concurrently tracked	100				
Illumination range	50-100,000 Lux (5-400,000Lux in HDR mode)				
Max illumination variation between markers	6:1[3] (1000:1 in HDR mode)				

Certifications	CE and Part 15 Class A of FCC, RoHS and WEEE compliant	Class II medical device: IEC 60601-1, CAN/CSA C22.2, CSA 601.1, UL 60601-1 (Certification pending for Sx60/Hx40/Hx60 and H3-60 models)
Drift: temperature, pressure, light intensity, marker orientation, motion blur, exposure settings	insignificant	
Hazard warnings issued	Marker out of FOM; Thermal instability; Temperature out of range; Shadow over target	
Operating system	MS-Windows 2000/XP/Vista/Windows 7, Linux, Mac OSX – 32bit and 64bit	
Minimal computer hardware	2 GHz CPU, 512MB RAM, 500MB disk space	

<u>NOTES</u>*:*

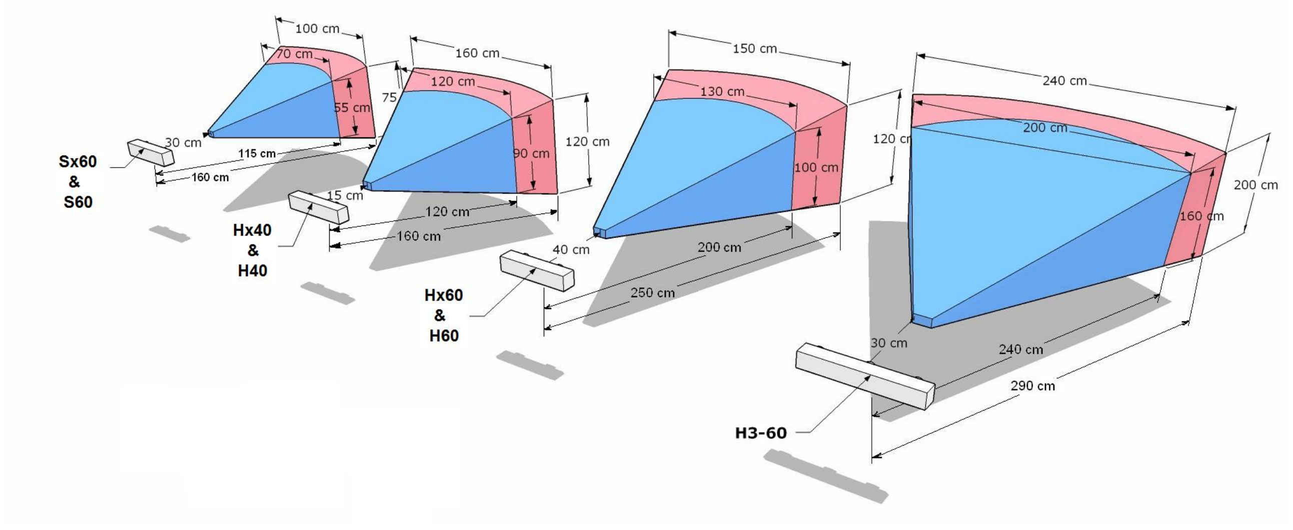
1) *Single target position accuracy. For H40 and Hx40, ~30,000 averaged positions at distances of 40-85cm, for S60 and Sx60, ~10,000 averaged positions at distances of 50-85cm, for H60 and Hx60, ~30,000 averaged positions at distances of 60-155cm, for H3-60, ~50,000 averaged positions at distances of 60-170cm. Calibration accuracy decreases when smaller footprint detection is enabled.*
2) *Single target at a distance of 75cm, 150 Lux, 20ms shutter.*
3) *The illumination variation range can be extended to 20:1 with the use of black/gray markers (see 9.2).*

The camera's optics and electronics are housed in an internal aluminum case and an external plastic shell. The external shell is designed to protect the internal case against electric and physical shock. The camera has been certified for compliance with medical safety standards and carefully calibrated to perform with the external shell.

<u>NOTE</u>: *Removing the protective shell would nullify the certification and degrade the camera's measurement accuracy.*

5.2 Field of Measurement (FOM)

MicronTracker's FOM is defined as a volume in which Xpoint targets of a 12 mm radius can be detected within accuracy specifications. The FOM boundaries are shaped approximately as a rectangular section of a sphere, as shown below. MicronTracker supports both the standard and extended FOM size (shown in blue and red, respectively, in the below diagrams). The size of the FOM can be extended as required, by enabling Xpoint detection with smaller image footprint (details in Section 4.2.1).



4.2.1 Extended FOM and small marker detection

An Xpoint target must cast a minimum projection "footprint" in the left and right camera images, in order to be detected by MicronTracker. MTC API provides means to configure the minimum footprint size to either 11 or 9 pixels in diameter. By reducing the minimum detectable projection of targets, the MicronTracker's field of measurement can be extended, so that Xpoints can remain visible at larger distances, or when tilted at larger angles away from facing the camera. Alternatively, the smaller footprint option can be used to enable detection of Xpoint targets of a smaller physical diameter within the standard FOM. The tradeoff for such an increased target visibility is a slightly lower detection accuracy, since fewer pixels may be used to pinpoint the position of an Xpoint.

All of the demo applications provide the interface for choosing between the two footprint sizes. In particular, "Detect small markers" option in the *Options* dialog box enables the 9 pixel footprint detection. Note that this option is enabled by default.

5.3 Multi-Camera

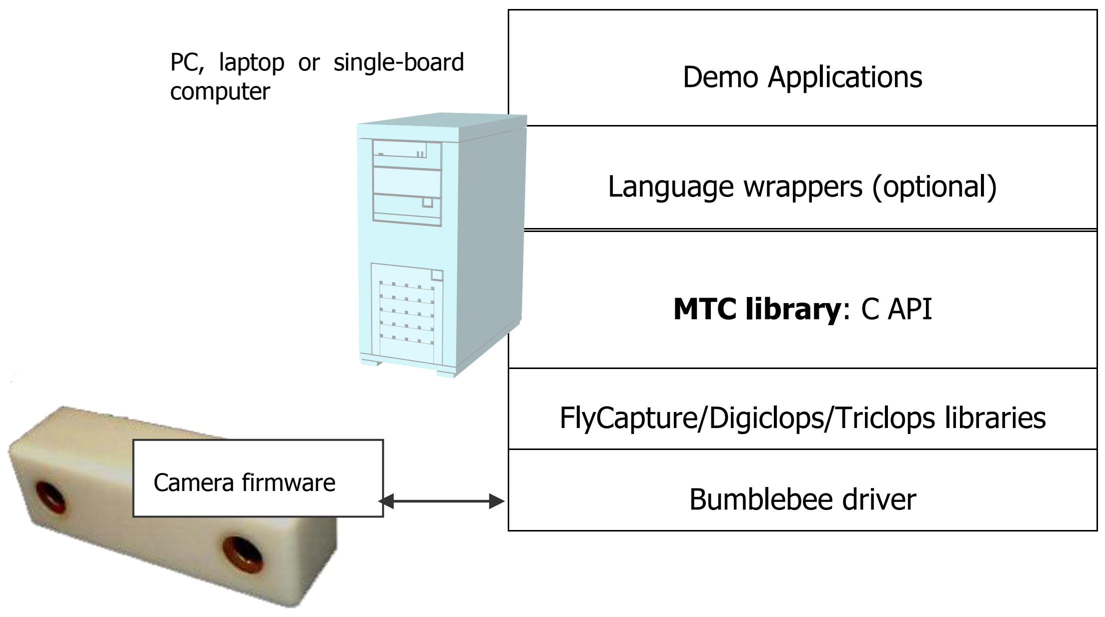
MicronTracker's software is designed to allow interfacing to a multi-camera set-up with ease. When multiple cameras are connected to the same IEEE-1394 bus, they need to share bus bandwidth (400Mbps), leading to reduced frame rate for each camera. The following table shows the maximum shared-bus frame rates obtained with 2 or 3 cameras:

Single-bus configuration	**Max. frame rate, Hz**
S60 + S60	30
S/Sx60+S/Sx60+S/Sx60	15
S/Sx60 + H40/60	7.5
H40/60 + H40/60	7.5
Sx60 + Sx60	24
Hx40/60 + Hx40/60	10

To obtain higher frame rates, the cameras may be connected to the computer using two or more separate buses (e.g., a built-in port and an IEEE-1394 PCI interface card). Cameras on separate buses do not automatically synchronize with each other and a synchronization unit (sold separately) is required. When cameras capture frames in an unsynchronized fashion, measurements of the same marker taken by different cameras are not simultaneous. This is, however, not a significant drawback for most applications. If synchronization is required, there are both HW and software solutions available. Please contact Claron Technology for further information.

5.4 Software

The following diagram shows the various software modules delivered as part of the MicronTracker system and their dependency hierarchy. Modules below the double line are supplied as executable libraries, and above it as example source code.


PC, laptop or single-board computer
Demo Applications
Language wrappers (optional)
MTC library: C API
FlyCapture/Digiclops/Triclops libraries
Camera firmware
Bumblebee driver

5.4.1 Firmware

The camera firmware runs on a dedicated processor inside the camera and communicates with the computer over the IEEE-1394 bus. The firmware controls the sensing and communications hardware and is critical to accurate performance of the product. Firmware version names contain 4 numbers (*major.minor.type.revision*). The camera is calibrated for a specific firmware version and updates to the firmware may require the camera to be recalibrated.

5.4.2 Bumblebee driver

The camera driver is provided by Point Grey Research. This driver depends on the IEEE-1394 driver, which is part of the operating system installation.

5.4.3 FlyCapture/Digiclops/Triclops Libraries

Higher level interfaces to the camera control functions, provided by Point Grey Research. Triclops provides support for unstructured stereo vision capabilities which are not used by the MicronTracker software, but are available to developers (contact Claron Technology for more details).

5.4.4 MTC Library

MTC (MicronTracker's C language interface) implements all the identification and pinpointing algorithms and provides a complete object-oriented application programming interface (API) for:

- Detecting and starting up the cameras connected to the computer, then loading their calibration files;
- Loading Marker templates from files;
- Capturing and processing camera frames on demand to report on detected markers and their poses;
- Creating or modifying marker templates;
- Customizing many aspects of the measurement operations to fit specific application needs.

Applications written in any language may call MTC directly to take full advantage of the MicronTracker's capabilities. MTC completely encapsulates and abstracts the lower-level camera control functions, making it unnecessary for the application to make calls to any of the lower-level modules.

MTC version names contain 3 numbers: *major.minor.revision*. The major version number, currently 2, remains fixed as long as backwards compatibility is maintained, usually over a number of years. The minor version number increments whenever new features are added, usually every few months. The revision, or "patch", number increments with versions containing new bug fixes, but no API changes or feature additions.

5.4.5 Language wrappers

While C libraries can be interfaced directly from any language, programmers developing applications in higher-level languages, such as C++, Visual Basic or Python, find it easier to interface to the MicronTracker using a "native" API. Since MTC is object-oriented, it is relatively easy and straightforward to write a thin "wrapper" around the MTC API, making its objects appear as native objects in the chosen language.

Claron provides a number of example language wrappers, in source code format, which can be used as are, or as examples to assist in developing a wrapper for other languages of the developer's preference. This code is neither documented nor thoroughly tested. Furthermore, it is not guaranteed to provide a full coverage of MTC functionality.

5.4.6 Demo Applications

A number of different demonstration applications are provided in source code format to help people understand how to optimally use MicronTracker: They include:

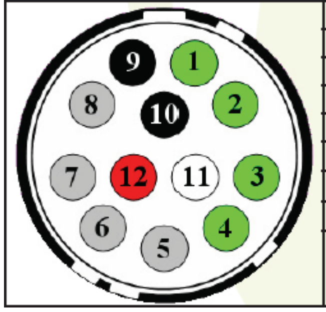
- MTSimpleAppC – a simple, short, C console program that demonstrates the basic MTC functionality.
- MTDemoVB6 – an extensive demonstration of product capabilities, written in Visual Basic 6.
- MTDemoCPP – a multi platform limited demo that supports both Windows and Linux environments and allows performing the basic MicronTracker tasks.

- MTDemoNet – an extensive demonstration of product capabilities, written in Visual Studio .Net 2005. The demonstration project includes a C++ wrapper for the MTC SDK, a VB.net demo and a C# demo.

5.5 General-Purpose digital Input/Output (GPIO)

The MicronTracker Sx60, Hx40, Hx60 and H3-60 cameras come with a 12-pin GPIO connector on the back that allows the camera to interact with external devices. It can be configured to trigger on an external electrical signal or produce a similar signal that allows devices external to the camera to be synchronized to the camera. To configure the GPIO pins, consult the camera's "Frame control/access" section of the MTC documentation. The following pictures show the GPIO connector on the back of the Sx60 camera and its circular push-pull coupling connector.

The GPIO pins are TTL 3.3V pins. Inputs can be configured to accept external trigger signals. When configured as inputs, the pins are internally pulled high using weak pull-up resistors to allow easy triggering of the camera by simply shorting the pin to ground (GND). The inputs are protected from both over and under voltage. It is recommended, however, that they only be connected to 5V or 3.3V digital logic signals. Outputs can be configured to send an output signal or strobe pulse. When configured as outputs, each line can sink 10mA of current.

Diagram	Pin	Func-tion	Description
	1	IO0	Input / Output (default Trigger_Src)
	2	IO1	Input / Output
	3	IO2	Input / Output / RS232 Transmit (TX)
	4	IO3	Input / Output / RS232 Receive (RX)
	5, 6, 7, 8	n/a	Not connected
	9, 10	GND	
	11	VEXT	Voltage limit: 8-30V; Current limit: 1A
	12	+3.3V	Power external circuitry up to a total of 150mA
	To Configure the GPIO pins, consult the "Frame control/access" section of the MTC documentation.		

<u>NOTE</u>: *To use the RS232 functionality, a level converter must be used to convert the TTL digital logic levels to RS232 voltage levels. See B&B Electronics (http://www.bb-elec.com/) P/N: 232LPTTL for an example.*

<u>NOTE</u>: *The GPIO connector shipped with the MicronTracker is only an example of the type of connector required. To maintain EMC certification as a medical device, any application that make use of the GPIO port should make sure that a non conductive shell shall be present on the GPIO connector.*

6 Accuracy Performance

6.1 Reporting Terms

Accuracy, or **trueness,** is the closeness of measured values to their expected quantity as defined by some standard. The difference between the true value and the value reported by the instrument is called **the measurement error**. In measuring a spatial position, the measurement error is a 3-vector *(dx, dy, dz)*, where each component is the difference between the measured position and the true position projected onto the corresponding axis. The error value is often reduced to a single number, which is the Euclidean length of the error vector, i.e., $\sqrt{(dx^2 + dy^2 + dz^2)}$.

Due to a variety of uncontrolled factors, repeated measurements of the same quantity under different conditions generate slightly different values. It is, therefore, common to take a statistical view of accuracy, describing it in terms of **error distribution** around the true value.

A number of statistical values may be used to quantify error distribution in a single number. The **maximum error** (or its close relative, the median error) is easy to define, calculate and understand. Unfortunately, it is not a useful term in characterizing measurement performance, since its value critically depends on the number of samples taken and, due to random factors, may vary widely from one experiment to another unless the number of samples is very large, sometimes impractically so. The **average error** is also easy to define and calculate. It does not, however, reflect well enough how "safe" it is to rely on the reported value. In averaging, small errors are assigned the same importance as large errors, while users often consider large errors more important than small ones. The average error, therefore, tends to generate lower values than what users feel is the "true" inaccuracy of the instrument.

Perhaps the most useful statistical error quantification is the **% confidence interval** (CI), defined as the value below which a specified % of the samples fall. For example, the value reported for the 95% confidence interval (95%CI) implies that only 5% of the samples had higher error values. Unfortunately, CI values have not been widely reported in the past, and there is no consensus regarding what specific interval to use.

The most frequently used measure of spatial measurement accuracy is the **Root Mean Squared Error** (**RMSE)**, calculated by averaging the squares of the individual error values, then taking the square root of the average. If all the errors have the same magnitude, RMSE generates the same value as the average error. When some of the errors are larger than others, however, they are given more weight, resulting in the RMSE being larger than the average.

In the many experiments conducted to measure the accuracy of MicronTracker products we found the ratios between RMSE and 95%CI to be very similar across experiments:

95%CI $\approx$ 1.85 x RMSE

From publications, this ratio appears to be approximately valid in other optical tracking products as well. Since 95%CI and RMSE are closely related and RMSE has been widely used in the past, the accuracy characteristics of the MicronTracker are specified using RMSE.

When the accurate value is not known, it is often useful to report on the **repeatability** (or precision) of measurements in terms of their **standard deviation** (STD). Standard deviation is identical to RMSE when the average is the true value, i.e., it is calculated by taking the square root of the average of the squares of deviations from the average value. When the average is off the true value by distance *d*, the following relationship applies (assuming a normal error distribution): RMSE = $\sqrt{(d^2 + \text{STD}^2)}$

6.2 Application-independent accuracy reporting

The end-user of the application-specific system incorporating a MicronTracker product is interested in the measurement accuracy of the overall system under typical usage conditions. This accuracy will be affected by various factors, of which many are either outside the scope of the measurements MicronTracker provides, or would vary widely from one application to another. In particular, system integrators may vary the design of markers and the tools/objects to which they are attached.

Considering that

- Different marker designs would result in varying marker pose accuracy characteristics; and
- Other vendors report on accuracy of single target position measurement;

Claron Technology has decided to characterize MicronTracker's accuracy performance by referring to the 3D position of a single target (Xpoint). While single target positions are not typically used directly by applications, their measurement drives the computation of marker poses and can, therefore, be used to model and predict the marker pose measurement accuracy for a given marker design (see 9.5.2).

6.3 Factors affecting accuracy

The causes for measurement errors in optical pose sensors are numerous and their effects are often complex and interdependent. It is useful to divide them into 3 categories: (1) calibration (2) drift and (3) jitter. **Calibration errors** are introduced in the camera calibration process at the factory, and remain constant for a long time unless the physical structure of the camera changes, e.g., due to a physical shock. **Drift** errors are introduced by variations in the operating environment, e.g., temperature, lighting, or marker position/orientation. **Jitter** is a momentary deviation caused by random optical or electrical noise in the image capture and analog-to-digital conversion circuitry.

6.4 Calibration

During calibration, a sequence of steps is executed to extract individual camera characteristics, which are later used in converting the input pixel values into 3D target positions. In the most critical calibration step coefficients are extracted for the equations that convert pixel locations in the image into the corresponding projection rays in camera coordinate space (see 1.3). To obtain those coefficients, a 3D grid of targets at nominal spatial relationships to each other is presented to the camera, and the projection equations' coefficients are tuned to obtain the closest match possible between the nominal spatial positions of the targets and the equation representing the location of their projections in each image. Over 10,000 grid positions are used in the calibration of each MicronTracker camera. Thus, many grid positions contribute to the calculation of each projection coefficient. The effects of drift and jitter are reduced by controlling the environment in which the calibration is made and by averaging the results of repeated measurements of the same grid position.

Errors may be introduced during the calibration itself as a result of:

- Residual drift and jitter errors;
- Interpolation and grid registration errors;
- Deviations between nominal target grid positions and true positions.

Due to the large amount of redundancy in the process, the accumulated effect of these errors expresses itself as a vector between each nominal grid position and its positions as calculated from the pairs of images acquired during the calibration. The calibration accuracy is calculated from the magnitude of these error vectors. The RMSE of the calibration accuracy has been adopted by suppliers of competing pose tracking products and is, therefore, particularly useful in obtaining a simple comparative accuracy yardstick.

A detailed record of the calibration process, including the calibration error statistics and the range of coordinates over which it was measured, is stored in the calibration file of each camera, and can be easily obtained through the API (see 7.3). Claron Technology guarantees that the calibration error of each camera shipped is lower than specified.

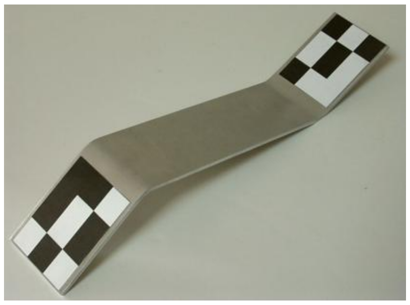
Grid registration errors are minimal at the center of the volume over which the calibration is performed, increasing towards its edges and beyond. To alert users to the degraded accuracy, an "outside calibrated volume" hazard warning is appended to the reported marker's pose when the depth (z coordinate) exceeds the farthest end of the FOM (1200/1150/2000/2400mm for H40,Hx40/S60, Sx60/H60,Hx60/H3-60 models).

The specified maximum calibration error for MicronTracker is 0.20mm/0.25mm/0.35mm/0.20mm RMS for H40, Hx40/S60, Sx60/H60, Hx60/H3-60. The actual calibration error for a specific camera is recorded in the calibration file header and can be viewed using the demonstration application (<**Calibration info**> in the <**Display**> menu), or accessed programmatically through the MTC API (`Camera_CalibrationInfo` or `Camera_CalibrationFileInfo`).

6.4.1 Calibration refinement with R-Fine

The MicronTracker camera is specifically designed to preserve the high accuracy of its calibration performed at the factory. But, like any other optical pose tracker, it can suffer from undesired hardware changes for a variety of reasons, such as environmental stresses or rough handling. These modifications, however slight, may in time render the camera's factory calibration less accurate than originally specified.

To address the issue of a potential degradation of calibration accuracy, **R-Fine** software application has been added to the MicronTracker kit. R-Fine provides a fast and intuitive method to *evaluate* the camera's calibration accuracy and the possibility to *refine* the calibration, if required. Calibration refinement in the field is a convenient alternative to shipping the camera back for full re-calibration. The R-Fine procedure is recommended to be performed every few months or if the camera has been subjected to physical stresses.



R-Fine software detects a specially designed **AccBar** marker, which is supplied as part of the MicronTracker kit. The software provides visual guidance to place the AccBar at specific positions and orientations in the camera's view. The goal is to collect a representative set of AccBar samples, detected over the camera's FOM. Each sample contains an estimate of the AccBar's length, computed from its image projections using the camera's calibrated projection rays.

The *variability* of the AccBar's length (measured as its STD) over the camera's FOM is used to quantify a characteristic non-rigid distortion of the camera's calibrated space due to likely physical changes in the hardware. R-Fine uses the collected samples to optimally adjust the camera's calibration model in order to lower the length variability and thus eliminate the distortion effect. Typical variability results produced by an accurate calibration are considered to be within 0.20-0.30 mm range for H40/Hx40, 0.25-0.35 mm range for S60/Sx60 and 0.35-0.40 mm range for H60/Hx60 and 0.20-0.35 mm for H3-60.

In addition to the length variability, R-Fine reports a *projection ray convergence* value before and after the refinement. Ray convergence describes the precision of intersection of the camera's projection rays, where they meet to form a 3D point from its left and right image projections. Ray convergence is expressed as an RMS shortest distance, or intersection gap, between the left and right rays corresponding to a triangulated point. The RMS is computed over all pairs of projection rays associated with the collected AccBar samples. The lower this value, the closer the calibration resembles an ideal camera model, in which the rays intersect at unique points in space.

NOTE: *Because accuracy requirements on the MicronTracker camera are application-dependent, MicronTracker users are encouraged to apply their own methods of evaluating the camera's accuracy in combination with the R-Fine protocol. Please contact us if the camera's tracking accuracy remains unsatisfactory after the refinement procedure.*

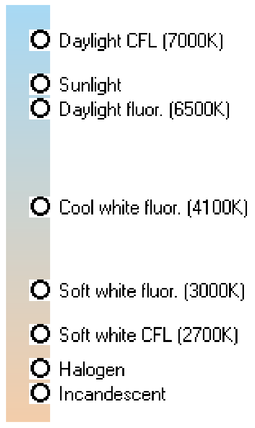
6.5 Drift Factors

Much effort in the development of the MicronTracker has been invested in detecting, characterizing and mitigating drift errors. As a result, MicronTracker is largely unaffected by external measurement conditions. In some cases, where the drift is hard or impossible to eliminate, a decision was made to implement hazard condition detection and associate a drift hazard warning with the affected measurements. The following table summarizes how various potential drift factors impact the performance of the MicronTracker:

Drift factor	**Impact**
Rapid internal temperature change in camera	A thermal instability hazard is added to measurements until camera becomes thermally stable (≈15 minutes for binocular models and 30 minutes for H3-60 model following power-up or 2 minutes following activation, whichever is later).
Ambient temperature	Camera automatically corrects thermal drift within operating temperature range (18-30°C or 66-86°F). A hazard warning is issued when temperature falls outside the range.
Atmospheric pressure	Insignificant
Light intensity or direction	Insignificant
Light type/spectrum	Insignificant when the *light coolness* property (see next) is correctly set by the user. Incorrect setting may affect the Z coordinate of measured positions by up to ±0.25% (although error rarely exceeds ±0.1% in most common in-doors circumstances).
Xpoint rotation and tilt angle relative to camera	Insignificant*.
Image contrast around Xpoint	Insignificant.
Partial occlusion or staining of Xpoint	Insignificant
Uneven lighting on Xpoint	Small drift may result when part of an Xpoint is shadowed at certain orientations. This condition is detected and a "Shadow over XP" hazard is appended to the marker's measurement.
Motion blur	Insignificant.
Power supply voltage variations	Insignificant.
Camera mount and orientation vs. gravity	Insignificant
Camera exposure settings	Insignificant

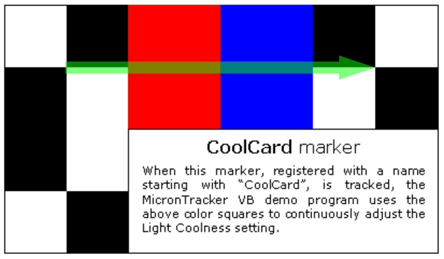
*) While Xpoint orientation by itself does not cause drift, a small sampling drift is introduced when the Xlines are aligned with the image X and Y axes.

6.5.1 Light Coolness



The spectral contents of the light (the mix of component colors) reflected off Xpoints affects the rays' refraction path through the lenses and, therefore, the exact location where each feature point appears on the sensor. The relevant property is called, in the MicronTracker's API, the "*light coolness*", representing the overall balance between the intensities of the light rays along the continuum between the blue (cool) and red (hot) ends of the visible spectrum. The more the balance tips towards the blue end, the light is cooler. Light coolness is closely related to the "light temperature" scale often used in photography, with higher temperatures representing a cooler color mix. The chart on the left shows the positions of commonly found light sources on the light coolness scale, some of them with their light temperature rating (CFL stands for "compact fluorescent light").

The MicronTracker software contains correction algorithms that automatically compensate for changes in light coolness, eliminating the drift that would otherwise result. The internal coolness scale has been calibrated such that 0.0 represents incandescent light and 1.0 represents Daylight CFL, and the allowed value range is –0.5 to 1.5.



When this marker, registered with a name starting with "CoolCard", is tracked, the MicronTracker VB demo program uses the above color squares to continuously adjust the Light Coolness setting.

Setting the light coolness correctly can be done either automatically or manually. Both methods are demonstrated in the VB6 demo provided as part of the SDK. The automatic method uses a "color vector" (show in transparent green in the figure on right) containing patches of red and blue at known positions between its base and head. When it appears in the FOM, the vector is identified by the application, e.g., by being the long vector of a marker with a known name, and submitted to MTC to be used for auto-adjustment. For reliable results, it is very important that the marker surface would be highly non-reflective (matte), since light reflections negatively affect the coolness calibration's accuracy. It is also important to use long-lasting, stable, colors which will not fade (such pigment-based dyes).

Automatic light coolness adjustment can be either performed once at the beginning of a measurement session by flashing the "CoolCard" provided as part of the MicronTracker kit, or continuously, by leaving the CoolCard visible in the FOM, or by embedding a coolness tuning "color vector" within markers in use for other purposes, such a coordinates reference marker. MTC supports the concept of "color profiles", identifying a specific combination of color patch sequence, ink, paper and printer. Currently, only a single profile, associated with the CoolCards issued by Claron, is supported. The VB6 demo identifies the long vector of the first facet of any marker whose name starts with the letters "cool" as a color vector.

<u>NOTE:</u> *Starting with the MicronTracker release 2.5 the "CoolCard" marker shipped with the kit is made of the adhesive Duramark material, instead of paper, for longer lasting durability. Since releases 2.5 and higher no longer support*

paper profile, the owners of a paper CoolCard are advised to replace it with the Duramark version when upgrading their release, in order for automatic light coolness adjustment to work properly.

Manually setting the light coolness is straightforward, and is demonstrated in the "Options" dialog of the VB6 demo (see Section 7.10).

6.6 Jitter

Measurement jitter is the direct result of image noise. Image noise, in turn, is affected by the camera's gain setting. MicronTracker's automatic exposure algorithms have been designed to optimize exposure settings for minimal measurement jitter. Proprietary jitter-filtering algorithms further reduce measurement jitter when markers are momentarily static. The jitter filtering has no negative impact on measurement accuracy or response lag.

The following table shows the H40 jitter's standard deviation under a selection of typical measurement conditions. In general, jitter for static targets, such as for reference markers, is very low under all conditions. Jitter magnitude for moving targets is proportional to the root of the gain factor and the distance.

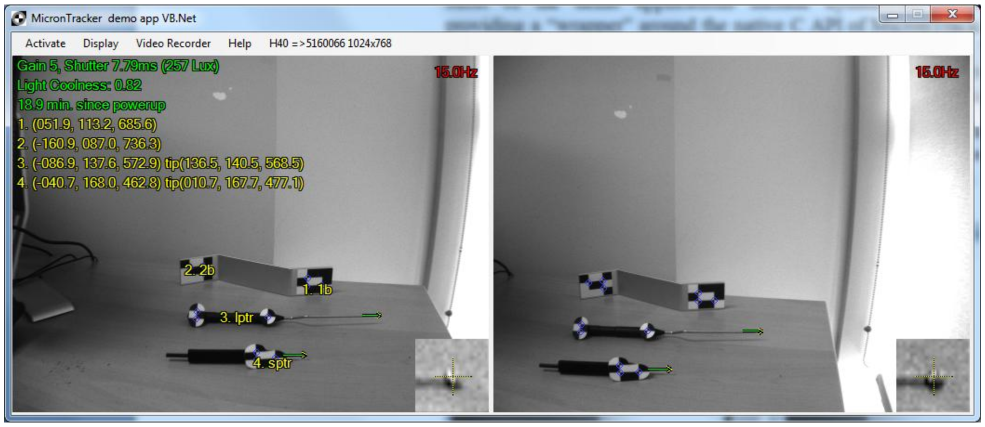
Condition	**S60, STD mm**	**H40, STD mm**
Static target, z=750mm, 150 Lux, GainF = 2.9	<0.01	<0.02
Static target, z=750mm, 500 Lux, GainF = 1.0	<0.01	<0.02
Moving target, z=750mm, 150 Lux, GainF = 2.9	0.11	0.22
Moving target, z=750mm, 500 Lux, GainF = 1.0	0.07	0.14

7 The VB.Net Demo Application

MicronTracker ships with a few example applications written in Visual Basic .Net, Visual C#, CPP and VB6 in both executable and source formats. By default the application is installed under *Program Files\Claron Technology\MicronTracker*. A shortcut to this application is placed on the desktop and it is also available through *Start* → *Programs* → *Claron Technology* → *MicronTracker*. The applications demonstrate usage of most of the MicronTracker API, and enable accomplishing many simple measurement operations. A user can register markers by showing them to the camera, record distance and angle measurements between markers as well as accuracy statistics, register multiple cameras to create a combined FOM, take frame snapshots, and produce an AVI time-lapse video of a measurement session. A user may also modify various camera settings and immediately view their effect.

Most of the demo applications include dynamically-loaded library (DLL) providing a "wrapper" around the native C API of MicronTracker (MTC.dll). For instance MTInterfacedotnet.dll is used both by VB.net and Visual C# demo applications and the VB6 demo application includes an ActiveX class library "wrapper" around the MTC.dll. On overview of the API is provided in section 10.

The VB.net and Visual C# applications are the most developed demo applications. The source code is installed under *Program Files\Claron Technology\MicronTracker\MTDemosNet.* In the rest of this chapter the main features of the VB.net demo application are discussed.

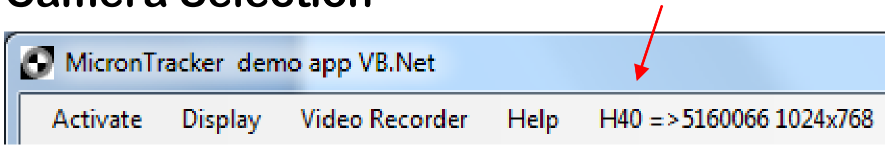


Gain 5, Shutter 7.79ms (257 Lux)
Light Coolness: 0.82
18.9 min. since powerup
1. (051.9, 113.2, 685.6)
2. (-160.9, 087.0, 736.3)
3. (-086.9, 137.6, 572.9) tip(136.5, 140.5, 568.5)
4. (-040.7, 168.0, 462.8) tip(010.7, 167.7, 477.1)

Snapshot of the MicronTracker VB.net Demo application

NOTE: *The VB6 Demo application code is used internally by Claron during development and quality control, which sometimes requires access to low-level or advanced features not supported as part of the public API. The development-only parts of the code are conditionally compiled, depending on the value of the "dev" pre-processor variable (i.e., they are bracketed between "#if dev then" and "#end if").*

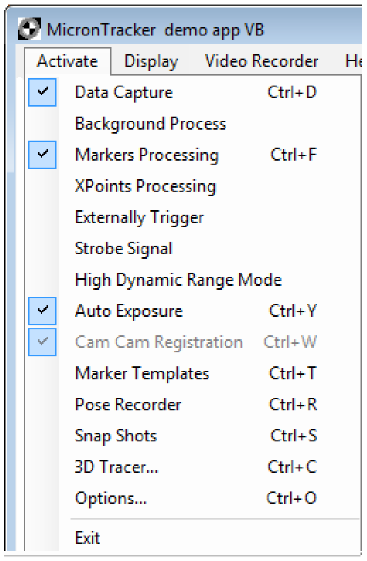
7.1 Camera Selection



On startup, the demo application automatically attempts to activate all the MicronTracker cameras connected to the computer. Once activated, tracking and display commence immediately. The activated cameras are listed by the MicronTracker model name, serial number and resolution on the right side of the menu bar (red arrows in the above figure). If there is more than one camera, selection of the "current" camera is available by clicking on the camera's listing in the menu bar. The current camera is indicated by an arrow on the left side of its listing.

The current camera selection determines the images shown and the marker pose measurements. Pose measurements made by other cameras registered to the current one is shown as if made by the current camera, even if the current camera does not see them.

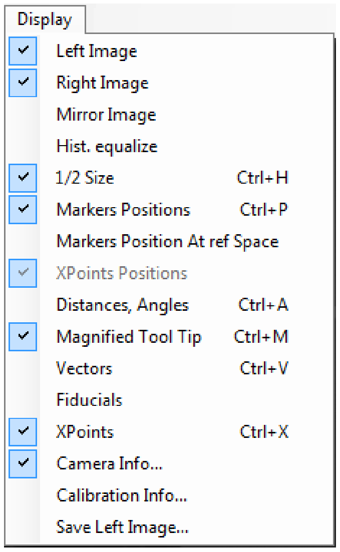
7.2 The Activate Menu

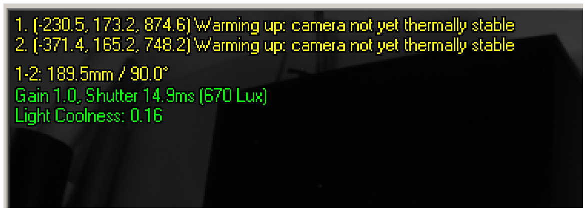


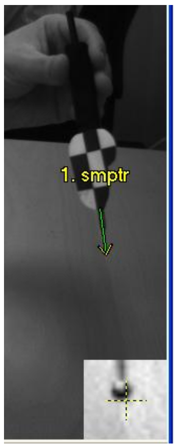
The Activate menu allows starting/stopping the data capture or the frame processing. The frame update rate and the data capture status are displayed at the top right of the images. <**Externally Trigger**> and < **Strobe Signal**> are available for Sx60, Hx40, Hx60 and H3-60 cameras. Enabling <**Externally trigger**> will stop the normal capturing mode and will put the camera in different mode. In this mode the camera, does capturing and frame processing right after receiving an external signal trough GPIO connector. The <**Strobe Signal**> doesn't interfere the normal execution of the application and just transmits electrical signals when the camera captures new images. Toggling on <Background Process> makes the MicronTracker grab and process frames automatically in a background thread and retrieves the most recently processed pose data. This allows faster processing and it increases the performance of the capture process about 30%. Toggling off <**Auto Exposure**> enables the user to manually control the camera's shutter and gain settings using controls which appear on the right of the top image. The <High Dynamics Range Mode> toggle enables/disables HDR mode for the camera, a feature that enables the camera to detect markers is a wider range of illumination like under OR light. The <**Cam-cam registration**> toggle enables/disables the automatic update of the camera-to-camera registration transform whenever the same marker is measured by multiple cameras. The <**Marker Templates…**> selection invokes a dialog to register new markers and manage the MicronTracker markers template database (see 7.6). The <**Pose Recorder…**> entry activates a dialog to record pose measurements for statistical analysis (see 7.7). The <**Snapshots…**> activates a dialog to save and restore complete measurement frames (see 7.8). The <**3D Tracer…**> dialog supports creating an augmented-reality overlay trace (see 7.9). The <**Options…**> dialog allows modifying some general tracker settings (see 7.10). For convenience, activating most menu options is also possible through keyboard accelerator keys.

NOTE: *The automatic exposure controller needs a constant stream of frames. Enabling the <Externally trigger> mode, particularly with a significant time delay between two frames, may cause the auto exposure to perform incorrectly. It is advised to use manual exposure when <**Externally trigger**> mode is enabled.*

7.3 The Display Menu




```
1. (-230.5, 173.2, 874.6) Warning up: camera not yet thermally stable
2. (-371.4, 165.2, 748.2) Warning up: camera not yet thermally stable
1-2: 189.5mm / 90.0°
Gain 1.0, Shutter 14.9ms (670 Lux)
Light Coolness: 0.16
```



The Display menu allows selecting various options for image and overlay display. Each option can be toggled on or off.

The <**Hist. Equalize**> option activates a special algorithm (a variation of histogram-equalization) to maintain a visually consistent image despite the constant manipulation of the exposure settings by the MicronTracker. This is a particularly useful option when image overlays are being used to mark locations and objects of interest ("augmented reality").

Enable <**Markers Position at ref Space**> to display the pose data at the Reference space. Both marker and the tooltip pose data will be displayed at the reference space. By disabling this option it will display the pose data at the "Camera-Space".

When <**Magnified tooltip**> is activated and a marker with a corresponding tooltip is detected, the regions around the tooltip in both left and right images are magnified 4X (from the original video image), contrast-enhanced, and displayed as insets within the video images. A cross mark shows the assumed location of the tooltip, as extrapolated from the marker location. This provides an excellent indication of the exact location actually being reported as being at the tip of the tool, and provides a simple visual inspection to detect problems with tooltip registration (see 7.6.4), marker attachment or tooltip step bending.

When the <**Vectors**> option is on, the vectors of identified facets are shown in cyan (the longer vector, defining the X axis) and magenta (the shorter one). Vectors that were not matched to a known facet are shown in yellow.

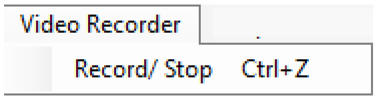
When exposure parameters are displayed, the demo also shows an estimate of the amount of illumination, in Lux, falling on the brightest marker tracked.

The demo includes support for identifying a variable number of vectors used for marking a variable number of locations, and not associated with any specific marker. These locations are called "fiducials" and are assumed to be marked by vectors with length of 10mm or less (this parameter can be programmatically modified by changing the value of the `FiducialVectorMaxLength` property of the main form). When the <**Fiducials**> option is turned on, such vectors are marked in the image by a green circle and the letter "f".

The calibration record of your camera can be viewed by selecting <**Calibration info>.**

<**Save left image…**> saves the left (top) picture, exactly as it appears on the screen (not as captured), compressed using JPEG, in a file.

7.4 The Video Recorder Menu



The Video Recorder menu demonstrates the creation of a time lapse AVI file documenting a procedure in video. When recording is activated, the user is prompted to select the output AVI file path and the video compressor to use. The time interval between recorded frames is determined by the setting of the `tmrAviSaver.Interval` property of the main form (500 milliseconds by default). To minimize interruptions to the pose measurement cycle on slower computers, an uncompressed format is recommended. A blinking red dot appears at the top right of the top image when recording is in progress.

To eliminate image brightness oscillations during the video recording, you should ensure that the <**Hist. Equalize**> option in the Display menu is checked.

7.5 Help

<**MTC API Documentation**> opens a web browser to show the home page of the API documentation. The <**About…**> dialog box provides information on version

number of the executable files being used, and allows access to complete system information.

7.6 The Marker Templates Dialog

<u>NOTE</u>: *For optimal performance, new marker templates should be registered only when no hazards are associated with the measurements.*

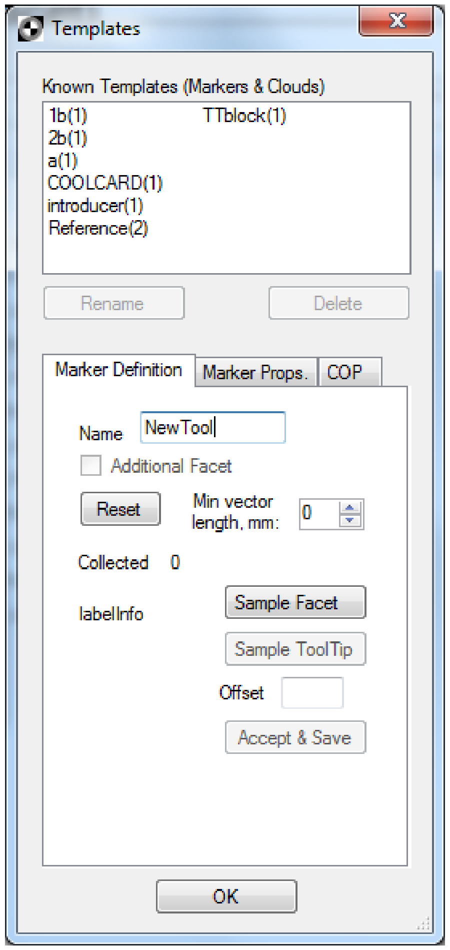
7.6.1 Registering a new marker template (single facet)

To register a new marker template:

- Select <**Templates…**> from the Activate menu (or press Ctrl+T). The Templates dialog is displayed. A list of the currently registered *marker* or *cloud* templates is shown. The number of facets in each marker template appears in parentheses next to its name.

- To register a *marker* select the *Marker Definition* tab and type the new marker's name in the Name field. The name needs to be a valid MS-Windows file name, and different from any other registered template's name. Otherwise, the <**Sample facet**> button will be disabled.

- Make sure that no other unregistered markers are visible to the camera. Place the new marker in front of the camera such that it will be fully seen by both lenses and click on <**Sample Facet**> to toggle sampling on. When exactly two unidentified vectors are seen by the camera, the "Collected" counter should increment with each frame. Otherwise, the reason a sample cannot be collected appears below the "Collected" line. If the marker contains short vectors that should be ignored (often the case when two XPs share B/W regions), you can set a minimum vector length to be recognized. Once you have a substantial number of samples (>30) you can stop the recording by clicking again on the <**Sample Facet**> button to toggle it off. If you are unsure of the quality of the sampling, you can click on <**Reset**> to discard the samples collected so far.

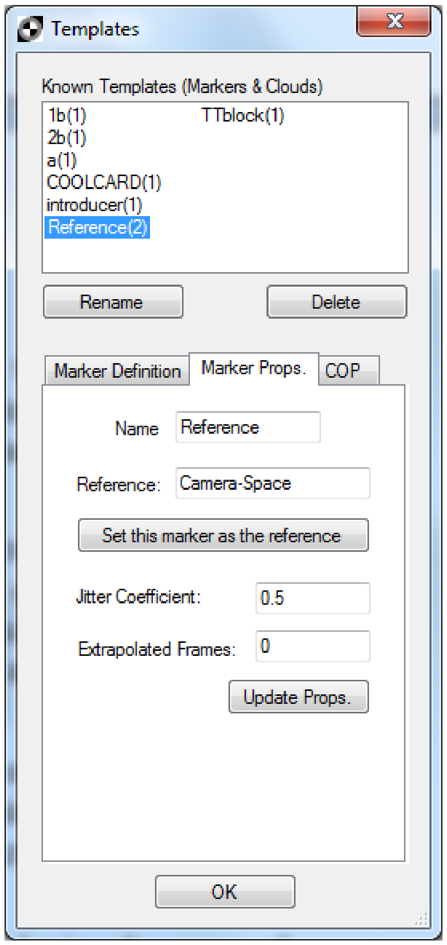


Known Templates (Markers & Clouds)
1b(1)
2b(1)
a(1)
COOLCARD(1)
introducer(1)
Reference(2)
TTblock(1)

Name: NewTool
Additional Facet: ☐
Min vector length, mm: 0
Collected 0
labelInfo
Offset:

- Click on <**Accept & Save**>. The marker is now in the database. Its name should appear next to the marker in the upper (left) image whenever the marker is in the FOM.

After saving the template, the database is automatically checked to see if the new marker's facet is too similar to other facets in the database, given the current setting of the template matching tolerance (see 11.3). If a similarity warning appears, you can choose to reduce the tolerance (Options menu), delete the similar marker(s) from the database, delete, redesign and re-register the new marker, or ignore the warning (taking the risk of possible misidentification of markers).



Known Templates (Markers & Clouds)

1b(1)
2b(1)
a(1)
COOLCARD(1)
introducer(1)
Reference(2)
TTblock(1)

Name: Reference

Reference: Camera-Space

Jitter Coefficient: 0.5

Extrapolated Frames: 0

Markers are stored in text files (.ini format) in the "Markers" folder, which is automatically created in the directory where the application's executable resides. As necessary, they can be manually edited, e.g., by using "Notepad" or "Wordpad". They may also be copied to other computers and used with any MicronTracker model.

The *Templates* form also enables to define a reference marker. Select the *Marker properties* Tab and from the *known Templates* list select a marker and then click on "Set this as the reference" button. From this moment the selected marker will be recognized as the reference marker for all other known markers.

By clicking on each known marker from the list, it will be displaying the name of reference marker for that particular marker. Note that if you select the reference marker from the list it will show "Camera-Space" as a reference for the reference marker.

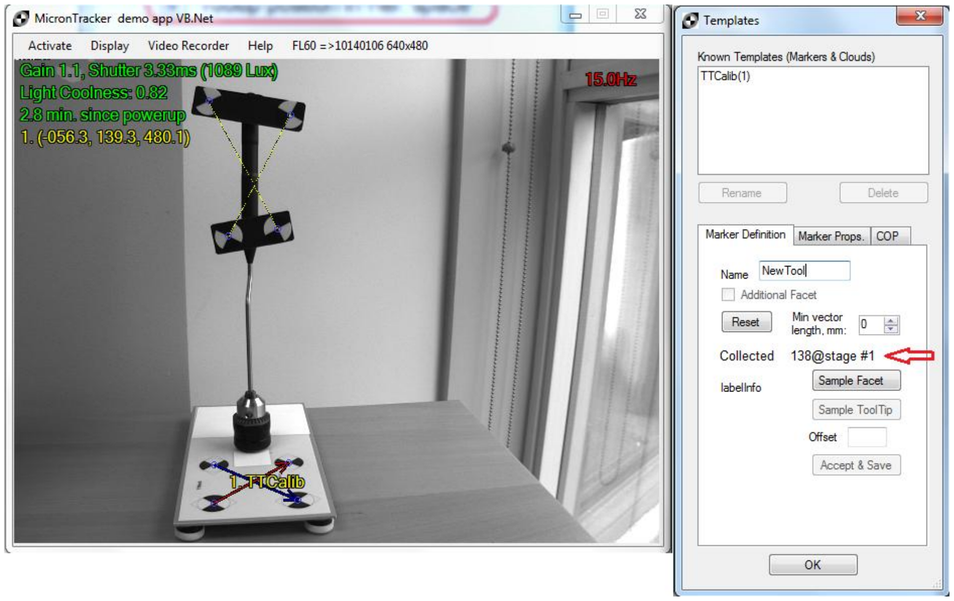
NOTE: *to remove incompatibility problems, the decimal dot in numbers stored in marker files represented by a period, regardless of the OS regional number format settings. Commas inserted manually would be interpreted as dots when the marker files are loaded.*

7.6.2 Registering a new marker template using UniTracker

To register a new marker template using UniTracker:

- Select <**Marker Templates…**> from the Activate menu (or press Ctrl+T). The Marker Templates dialog is displayed. A list of the currently registered marker templates is shown. The number of facets in each template appears in parentheses next to its name.
- Type the new marker's name in the Name field. The name needs to be a valid MS-Windows file name, and different from any other registered template's name. Otherwise, the "Sample facet" button is disabled.
- Make sure that no other unregistered markers are visible to the camera. Mount the new tool to the TTCalib marker rigidly and place them in front of the camera such that it will be fully seen by the camera and click on <**Sample Facet**> to toggle sampling on. Note that while collecting samples both the camera and the new marker and TTCalib must be in static condition. When exactly two

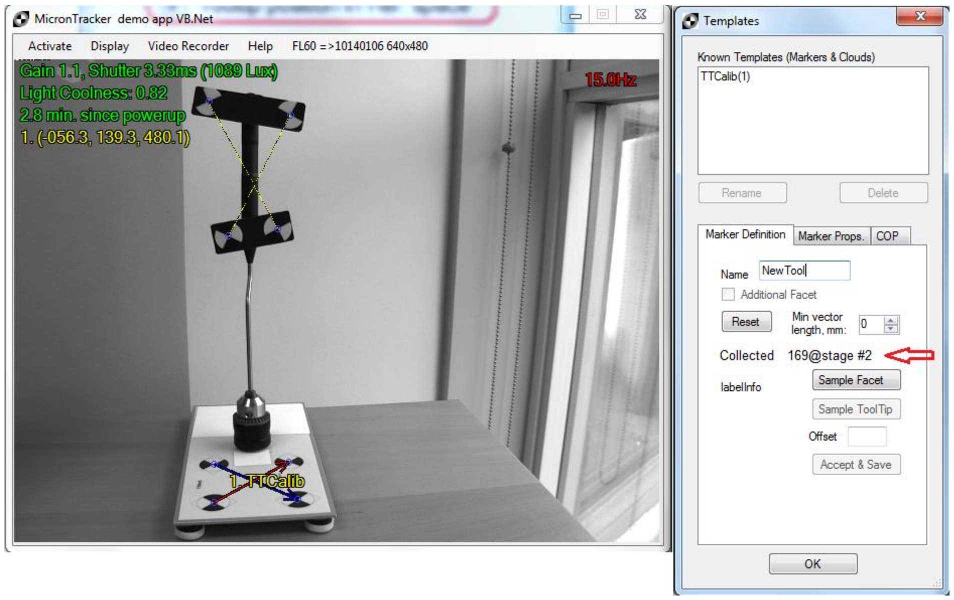
unidentified vectors and the TTCalib marker are seen by the camera, the "Collected" counter should increment with each frame. Otherwise, the reason a sample cannot be collected appears below the "Collected" line. Once you have a substantial number of samples (>50) you can stop the recording by clicking again on the <**Sample Facet**> button to toggle it off. If you are unsure of the quality of the sampling, you can click on <**Reset**> to discard the samples collected so far.



Gain 1.1, Shutter 3.33ms (1089 Lux)
Light Coolness: 0.82
2.8 min. since powerup
1. (-056.3, 139.3, 480.1)

Known Templates (Markers & Clouds): TTCalib(1)
Name: NewTool
Additional Facet
Min vector length, mm: 0
Collected 138@stage #1
labelInfo
Offset

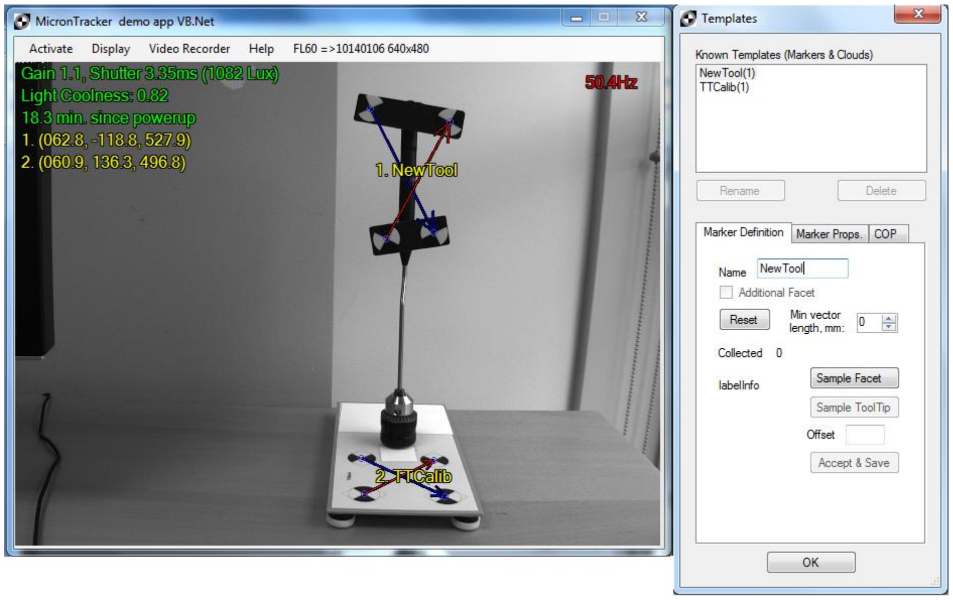
- Without changing the relative position of the new tool with respect to the camera, reposition the camera and collect another set of samples by clicking on <**Sample Facet**> to toggle sampling on.



Gain 1.1, Shutter 3.33ms (1089 Lux)
Light Coolness: 0.82
2.8 min. since powerup
1. (-056.3, 139.3, 480.1)

Known Templates (Markers & Clouds): TTCalib(1)
Name: NewTool
Additional Facet
Min vector length, mm: 0
Collected 169@stage #2
labelInfo
Offset

- Click on <**Accept & Save**>. The marker is now in the database. Its name should appear next to the marker in the upper (left) image whenever the marker is in the FOM.



```
Gain 1.1, Shutter 3.35ms (1082 Lux)
Light Coolness: 0.82
18.3 min. since powerup
1. (062.8, -118.8, 527.9)
2. (060.9, 136.3, 496.8)
```

Known Templates (Markers & Clouds)
NewTool(1)
TTCalib(1)

Name: NewTool
Additional Facet
Min vector length, mm: 0
Collected 0
labelInfo
Offset

After saving the template, the database is automatically checked to see if the new marker's facet is too similar to other facets in the database, given the current setting of the template matching tolerance (see 11.3). If a similarity warning appears, you can choose to reduce the tolerance (Options menu), delete the similar marker(s) from the database, delete, redesign and re-register the new marker, or ignore the warning (taking the risk of possible misidentification of markers).

Markers are stored in text files (.ini format) in the "Markers" folder, which is automatically created in the directory where the application's executable resides. As necessary, they can be manually edited, e.g., by using "Notepad" or "Wordpad". They may also be copied to other computers and used with any MicronTracker model.

7.6.3 Registering a multi-facet marker template

When the marker contains more than one facet, start by registering the marker with just a single facet as described in the previous section (7.6.1). If the multiple facets face in the same direction, you may need to cover the centers of Xpoints of all facets other than the one you are registering (e.g., by using the sticky part of a post-it note). If the facets are facing in different directions, make sure only one of them is facing the camera during the sampling step.

To add facets to the marker's template:

- Make sure the template's name appears in the "Name" box, and check <**Additional facet**> on.
- Position the marker (or uncover hidden Xpoints) such that both at least one registered facet and the facet you are adding are seen by the camera concurrently.
- Click the <**Sample Facet**> button to toggle sampling on.
- Collect samples of the additional facet together with the known facet (so that the MicronTracker software will register their relative spatial positions).

- Once a sufficient number of samples has been collected (>30), click the <**Sample Facet**> button to toggle it off, then click <**Accept and Save**>. The facet count in parenthesis next to the marker's name in the list should increment by one.

Repeat the above steps for each additional facet.

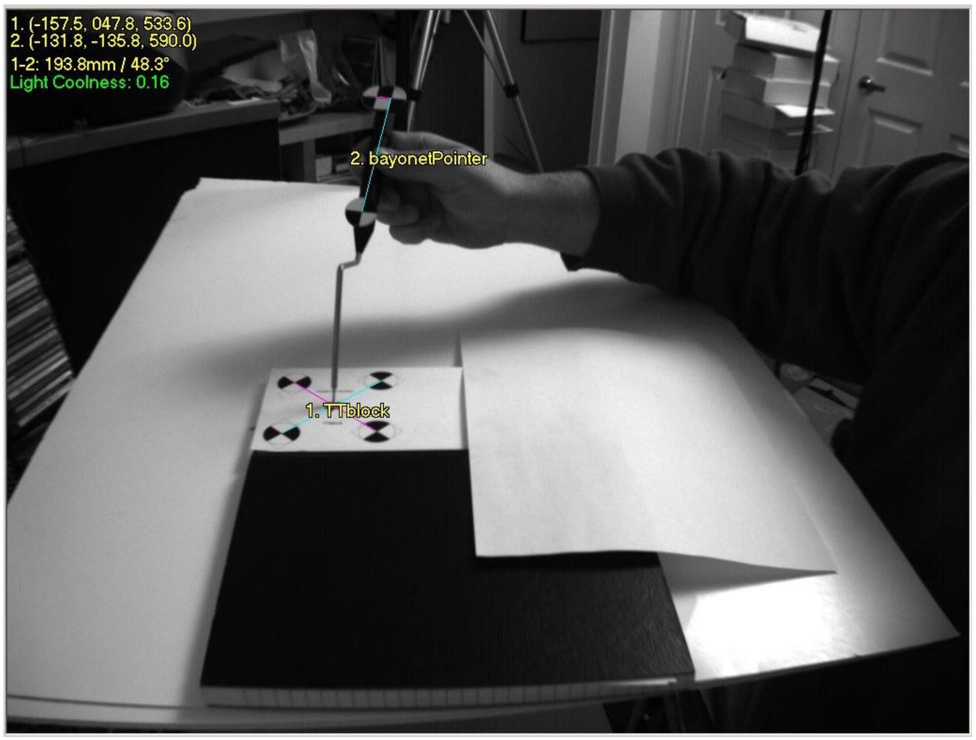
In some cases the additional facet is placed such that it cannot be observed by the camera concurrently with any previously recorded facet. For example, this is the case when the marker has only two facets oriented back-to-back. Several options are available for handling this case:

- Use an additional temporary facet during the template registration process. That facet should be positioned such that it can be observed together with both the registered and the unregistered facets. It is then registered relative to the first facet and subsequently used to register the additional facet. After the registration has been performed, this additional facet can be removed and ignored (you can manually edit the template file to remove it, or make sure the same facet it not registered with any other template).
- Use a multi-camera setting where the two facets can be simultaneously observed by two or more mutually-registered cameras.
- Treat each facet as a separate marker, and make the application aware that they refer to the same object, e.g. by assigning them marker names conforming to some convention (eg. [objectname].L and [objectname].R for left and right facets correspondingly).

7.6.4 Registering a tool tip

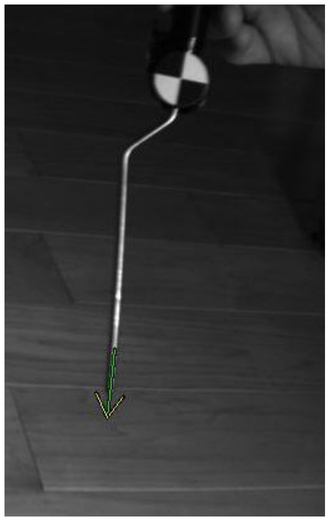
The demo provides a simple method for registering the position and orientation of the tip of a tool, such as a pointer, simply by placing it at the origin of another marker with a known name. An example of such marker, named "TTblock" (for Tool-Tip calibration Block) is provided as part of the MicronTracker SDK. Once a tooltip is registered, its position is reported on the screen and can be recorded using the Pose Recorder dialog. Furthermore, an arrow representing the predicted location and orientation of the tip in camera space is overlaid on the video images, allowing verification of the tip's calibration procedure.

The picture below shows a suggested setup for performing the tooltip registration using the example TTblock marker shipped with your MicronTracker unit. A sheet of paper covers the second marker on the plate, and a heavy notebook or book is placed in front of the plate to prevent it from sliding when pressure is applied against the small notch at the TTblock's origin. For maximum accuracy, the plate should be placed at a distance of 40-50cm from the camera.



NOTE: *The mechanical structure of the TTblock example does not provide highly accurate results (typical errors can be up to 1mm in position and 20 degrees in orientation). For more accurate results you can either order pre-calibrated tools from Claron, or use a more robust and accurate tip calibration setup/procedure of your own.*

To perform the registration, ensure that only the tool's marker and the "TTblock" marker are detected. Place the tip of the tool in the small notch on the board, touching the origin point of TTblock (the middle of the long vector), with the tip's stem at an elevation of 90 degrees directly above the orientation line pointing at the marker origin. Click on the <**sample tooltip**> button to begin sampling. Once a sufficient number of samples was collected (e.g., 50), click on the button again to complete the sampling and save the result by clicking on <**Accept and Save**>.



You should now observe the tooltip arrow overlaid on the video images. The tip of the arrow points at the positive Z direction in the tooltip coordinate system, and its side lobes lie in the tip's XZ plane.

7.6.5 Renaming and Deleting Markers

To delete or rename a marker, simply select the marker by clicking on its name in the list, then click on the <**Delete**> or <**Rename**> button.

7.7 The Pose Recorder

The pose recorder's purpose is to allow users to conveniently export a large number of pose measurements to a file for off-line analysis. The format in which the measurements are exported allows the file to be easily imported into spreadsheet programs to perform various calculations and statistical analyses on the measurements. To open the file in MS-Excel, for example, right-click on the file's name, then select *Open With → Microsoft Office Excel* (or open the file in Excel).

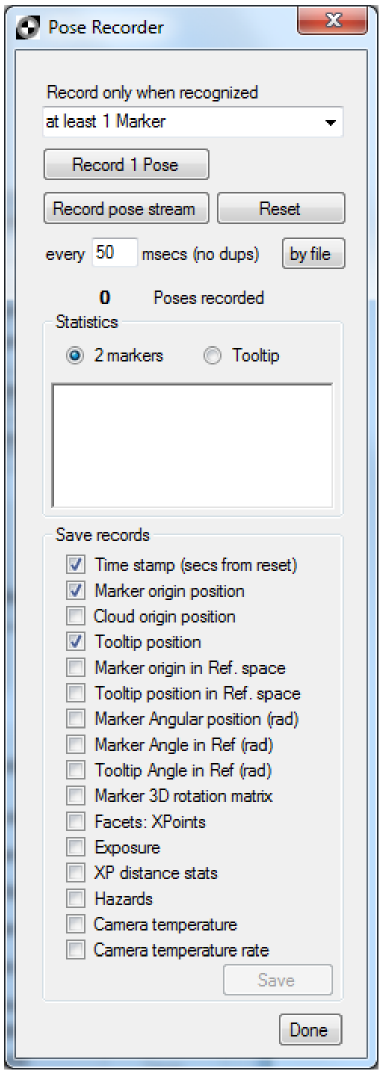
The data at each sample is collected in an internal record. Each data item in the record can optionally appear in the formatted text by toggling it on/off using the provided checkboxes.

NOTE: *The state of the checkboxes has no effect on what is being collected, only on what is being written to the output file. After the data was collected, you can save it multiple times to different files with different options enabled/disabled.*

Other than the various automatic and manual options controlling the data capture timing, it is possible to use the <**by file**> toggle to enable triggering record capture from another program with access to the computer's file system (i.e., a local

program or on another computer in a local area network with proper access permissions). When placed in the "by file" mode, the demo program checks for the existence of the file "*C:\MTmeasure*". When it exists, a record is captured, the file is erased to signal completion, and the program waits for it to appear again. The frequency in which the file's existence is checked is controlled by `tmrCheckFile.Interval` property of `frmPositionRecorder`. You can also easily change the name and/or location of the file by modifying the function `tmrCheckFile_Timer`.

When <**2 markers**> statistics is chosen, the program shows distance and angle statistics relating to records containing exactly two markers (only). Otherwise, it shows statistics relating to the tooltip position of the first marker in each record (which is identical to the marker's origin position if a tooltip is not registered for that marker).


Pose Recorder

Record only when recognized: at least 1 Marker

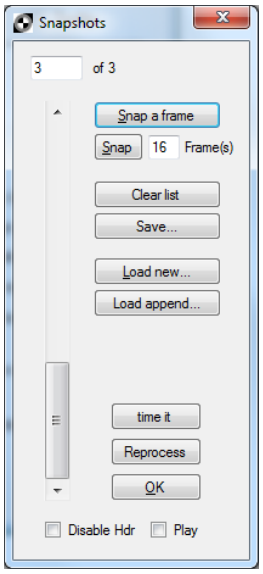
every 50 msecs (no dups)

0 Poses recorded

Statistics: [✓ 2 markers] Tooltip

Save records
- [✓] Time stamp (secs from reset)
- [✓] Marker origin position
- [ ] Cloud origin position
- [✓] Tooltip position
- [ ] Marker origin in Ref. space
- [ ] Tooltip position in Ref. space
- [ ] Marker Angular position (rad)
- [ ] Marker Angle in Ref (rad)
- [ ] Tooltip Angle in Ref (rad)
- [ ] Marker 3D rotation matrix
- [ ] Facets: XPoints
- [ ] Exposure
- [ ] XP distance stats
- [ ] Hazards
- [ ] Camera temperature
- [ ] Camera temperature rate

7.8 The Snapshots Dialog

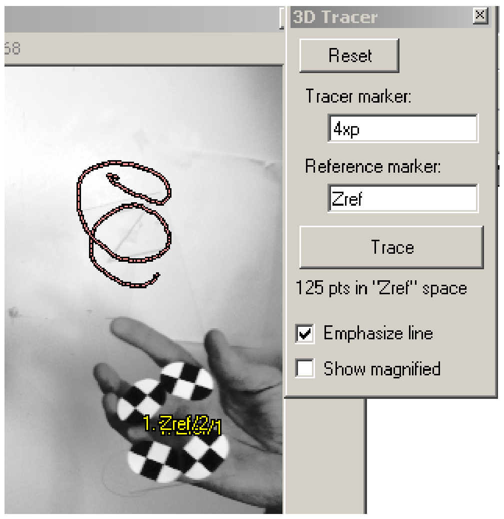


One or more snapshots of measurement frames (left and right image uncompressed) can be recorded by clicking the <**Snap frame**> button. Each snapshot is appended to the end of the snapshots list. Using the <**Save…**> button, the list may be saved in a special "stereo snapshots" files (.ssnap extension), from which it can retrieved later by using <**Load new…**> or <**Load append…**>. Scrolling through the list, using the slider or the frame number box above it, feeds the snapshots to the MicronTracker's interface software as if they were newly captured frames from the camera. Regular data capture is disabled, and can be enabled again by clicking the <**Data Capture**> option in the Activate menu (or by pressing Ctrl+D).

NOTE: *correctly performing measurements from snapshots requires that the calibration file loaded will be the same one used when the snapshots were taken. Calibration files are unique to each individual camera. A warning is, therefore, issued when a snapshots file is loaded while the currently connected camera is different than the one used when the snapshots were taken.*

7.9 The 3D Tracer Dialog

The 3D Tracer feature demonstrates how to use the MicronTracker's API to generate and display 3D geometry projected and overlaid on the image while "anchored" to a (reference) marker. The demonstration uses manual tracing to generate the geometry. The manual tracing may be performed, for example, to mark the location of visible boundary or feature points of an object assumed to be fixed in space relative to the reference marker during measurements. If the object accidentally shifts, a mismatch between the trace and the locations it marks would appear.



Tracer marker: 4xp
Reference marker: Zref
125 pts in "Zref" space
[✓] Emphasize line
[ ] Show magnified

To trace, you need to enter the name of the reference marker, the name of the tracer marker and click on the <**Trace**> button. When both markers are detected, line segments are added to the trace. The trace, if visible, is shown overlaid over both the left and the right pictures.

If no reference marker is entered, the segments are recorded in the camera space, and will thus be fixed to the camera. Otherwise, the line is anchored to the reference marker and would move with it when the marker or the camera is moved.

When tracing is activated, it is often convenient to start/stop the tracing by hiding/exposing the tracer marker.

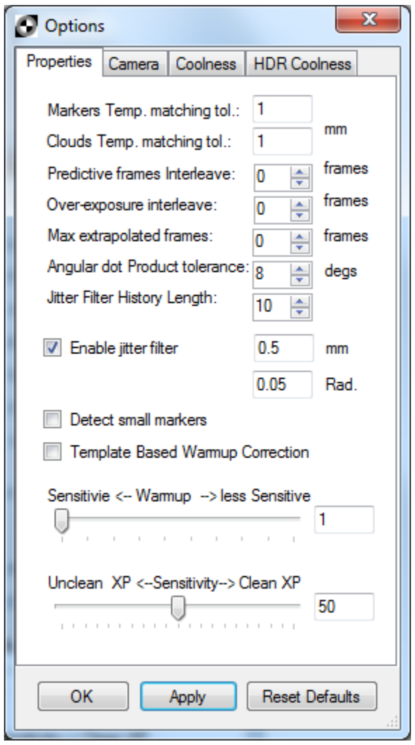
To better examine of the match between underlying feature locations in the image and the trace, the region around the center of the trace may be shown magnified by checking the <**Show magnified**> box.

7.10 The Options Dialog

This form allows modifying various attributes that control the performance of the MicronTracker. Entering a value for an option other than its currently set one causes it to be highlighted in red. To actually set the MicronTracker's value, click the <**Apply**> button.

The meanings of most of the attributes controlled by this dialog are described in detail in sections 11 and 6.5.1 (light coolness).

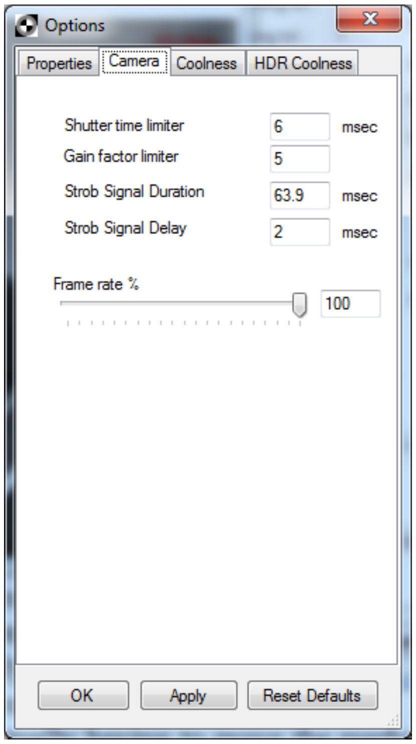
The *Properties* tab provides options to control Marker and Cloud detection behavior. For instance the *jitter filter* provides measurement noise reduction for static markers by averaging multiple measurements taken at the same location. It automatically shuts off at the slightest movement, and does not, therefore, have an adverse effect on dynamic measurements. It should, therefore, remain enabled during normal operation. You may want to disable it temporarily, however, in order to quantify the jitter level during dynamic measurements without actually having to move the marker being measured.



Markers Temp. matching tol.: 1 mm
Clouds Temp. matching tol.: 1
Predictive frames Interleave: 0 frames
Over-exposure interleave: 0 frames
Max extrapolated frames: 0 frames
Angular dot Product tolerance: 8 degs
Jitter Filter History Length: 10
[✓] Enable jitter filter 0.5 mm
0.05 Rad.
[ ] Detect small markers
[ ] Template Based Warmup Correction
Sensitivie <-- Warmup --> less Sensitive 1
Unclean XP <--Sensitivity--> Clean XP 50

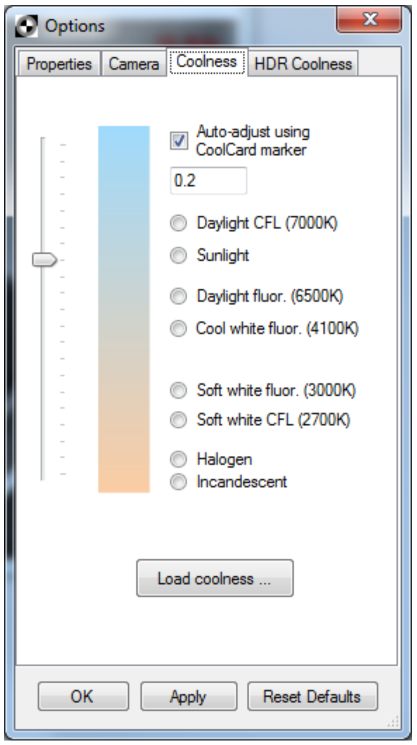
Selecting *Detect small markers* option will change the minimum Xpoint image projection, required for detection, from 11 to 9 pixels. This option can be used to extend the visibility of markers, enabling detection of smaller Xpoint targets, or targets farther away from the camera.

The Template Based Warm-up Correction compensates measured pose data using geometrical information stored in the marker's template file. If the marker is damaged and the template file does not correlate with the measured data, MicronTracker will ignore correction and will issue a warning message.



Shutter time limiter 6 msec
Gain factor limiter 5
Strob Signal Duration 63.9 msec
Strob Signal Delay 2 msec
Frame rate % 100

Setting the number of *maximum extrapolated frames* to a non-zero value can be used to reduce flickering of tracked targets due to momentary detection failure. If a marker is no longer detected, its predicted location can be used instead, extrapolated from its last successfully detected position. The number of extrapolated frames (ranging from 0 to 10) is the number of consecutive frames in which the extrapolated location of the undetected marker will be reported. This option is most effective for static or nearly static markers, as rapidly moving targets may result in their predicted locations lagging behind their actual ones, if the number of extrapolated frames is set too high.



☑ Auto-adjust using CoolCard marker

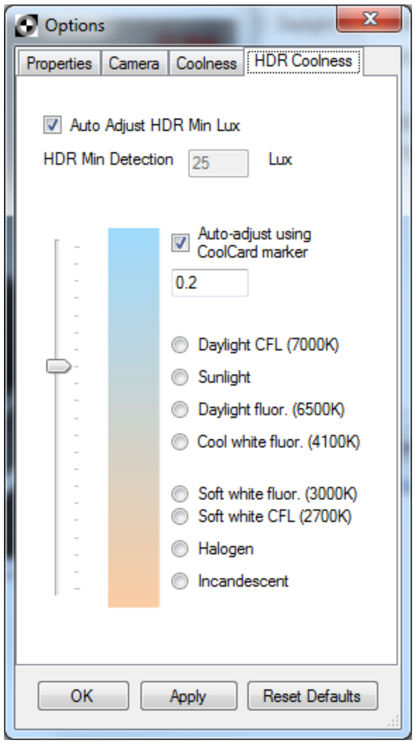
0.2

The Cameras tab provides options to control camera behavior like *Shutter* and *Gain limiter* and *strobe signal* properties.

The *Frame Rate* attribute enables to grab images with any desired frame rates. Its default value is the camera's maximum supported frame rate. You may change the frame rate either by moving the horizontal scroll bar or by typing numerical values in the corresponding text box.

The Coolness and HDR Coolness tabs provide options to control coolness parameter. Section 6.5.1 has more details about coolness.

Enabling *Auto Adjust HDR Min Lux* allows MicronTracker to automatically adjust the Min value for the High Dynamic Range mode. MicronTracker uses *HDR Min Detection* value only if Auto Adjust HDR Min Lux is disabled.



☑ Auto Adjust HDR Min Lux

HDR Min Detection 25 Lux

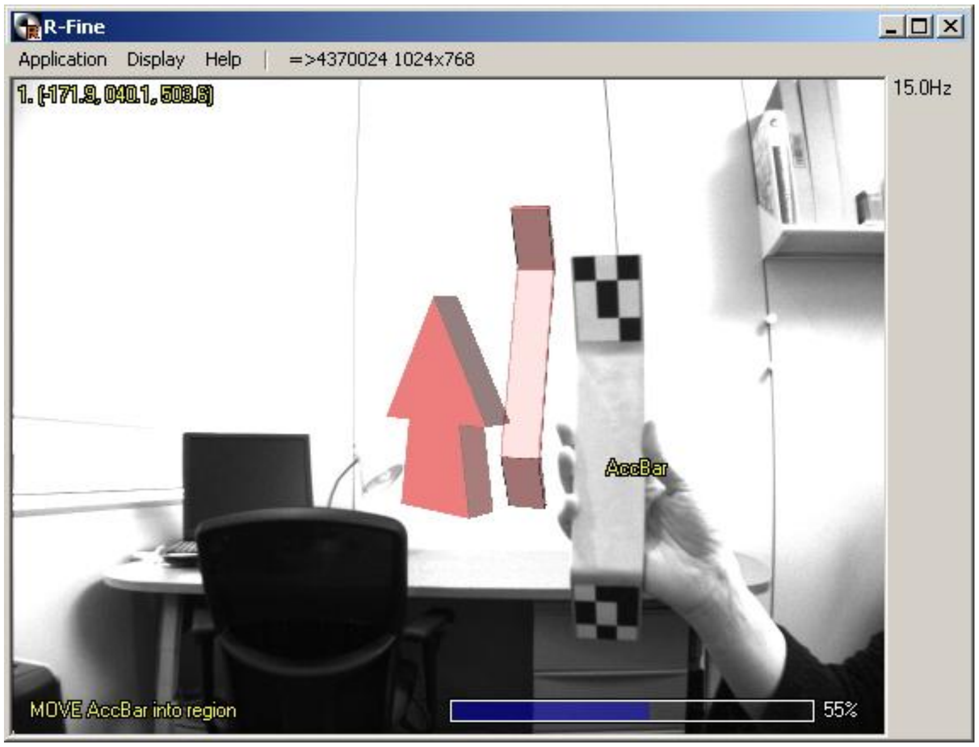
☑ Auto-adjust using CoolCard marker

0.2

8 The R-Fine Application

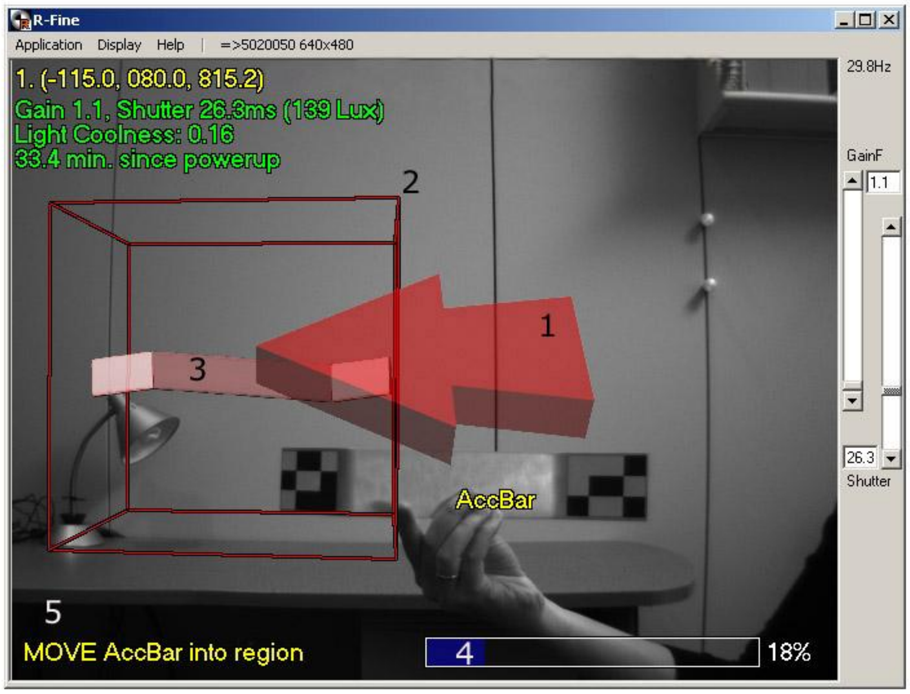
The R-Fine application comes as part of the MicronTracker kit and can be used for both verifying and restoring the accuracy of the camera's calibration (see 6.4.1). The executable is installed by default under *Program Files\Claron Technology\MicronTracker*. Its shortcut is placed on the desktop and is also available through the Start menu.

R-Fine operates by detecting a special two-faceted AccBar marker, mounted on a metal S-shaped tool. The tool must be placed by the user in a number of different regions and orientations, as guided by the software using semi-transparent image overlays. Once correctly positioned, the AccBar's computed 3D coordinates are sampled to be used for calibration accuracy analysis.



8.1 Main screen

To correctly position the AccBar tool in the camera's field of measurement, R-Fine uses the following visual guides:



1. 3D arrow (at the center of the screen), pointing in the direction the AccBar must be moved to reach its required location.
2. (Optional) outline of the FOM region inside which the AccBar must be placed.
3. Virtual AccBar icon, showing how the tool must be oriented in the camera's view (aligned with either X, Y or Z axis of the camera). Aligning the actual tool with its virtual image will correctly place the AccBar inside the current sampling region.
4. A status bar in the bottom right corner, displaying the percentage of AccBar samples collected so far.
5. A status message (bottom left corner) reporting the progress of sample collection. The message can be one of the following:
    - *AccBar not found* (the marker is not detected by MTC);
    - *Sampling disabled: reduced accuracy* (a hazard warning is issued);
    - *Low light* (the operating environment is not sufficiently illuminated);

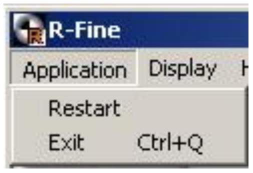
- *Move AccBar into region* (the marker is outside of the sampling region);
- *Orient AccBar as shown* (the marker is not oriented properly);
- *Collecting samples: N* (N is the number of samples collected in the current region for the current orientation).

The remaining components of the R-Fine's main screen, such as camera's information or hazard warnings, are identical to those of the VB6 Demo application.

NOTE**:** *R-Fine does not collect samples while a hazard warning is issued by the MTC, to avoid possible drift errors. This implies that the R-Fine procedure cannot be performed while the camera is still warming up after being powered*

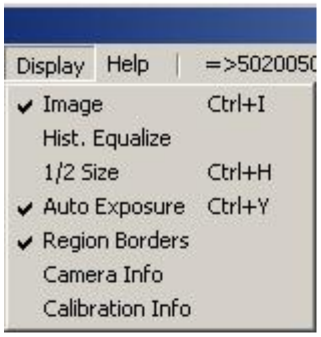
NOTE: *R-Fine has a built-in check to discard duplicate samples, i.e. a minimum displacement between two consecutive samples is required. If the sampling has slowed down or paused, move the bar slightly while maintaining its orientation within the region.*

8.2 The Application menu



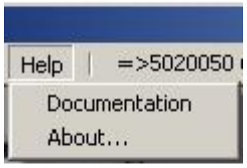
The Application menu provides the option of restarting the R-Fine's sample collection procedure, or exiting the application.

8.3 The Display menu



Much of the selections in the Display menu are the same as those for the VB6 Demo (see 7.3). In addition, <**Region Borders**> option is provided to toggle on or off the display of the 3D outline, highlighting the location in the FOM where the AccBar must be placed.

8.4 The Help menu

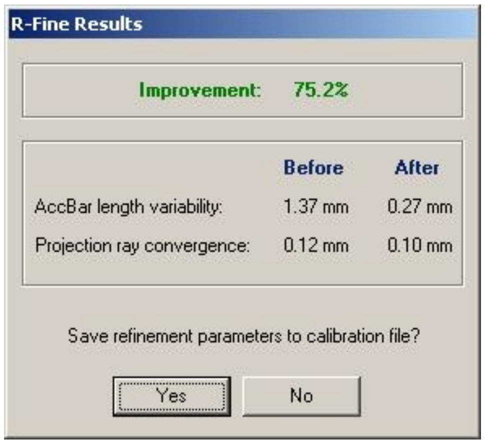


The <**Documentation**> opens a default web browser to the home page of the R-Fine documentation. The <**About…**> dialog box shows version numbers of the executable and DLL files, and allows access to complete system information.

8.5 The Camera

Similarly to the VB6 Demo, on startup R-Fine automatically detects any MicronTracker cameras connected to the computer via FireWire. However, R-Fine allows only a single camera to be activated at a time of its execution (otherwise an error message is issued). The serial number and resolution of the connected camera are displayed on the right side of the menu bar.

8.6 The Results window



R-Fine Results

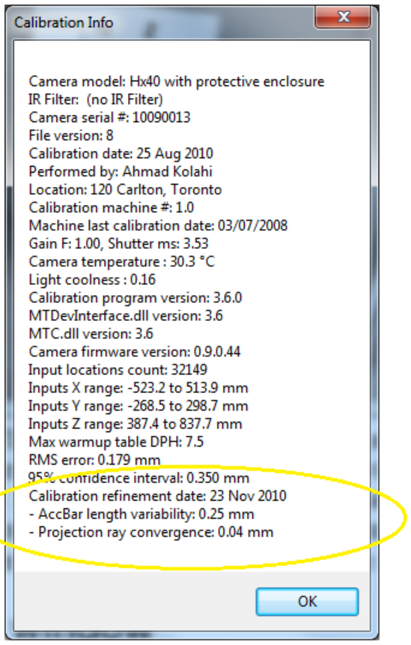
Improvement: 75.2%

	Before	After
AccBar length variability:	1.37 mm	0.27 mm
Projection ray convergence:	0.12 mm	0.10 mm

Save refinement parameters to calibration file?

Once all of the required samples have been collected, R-Fine performs their statistical analysis and automatically executes a refinement algorithm, which adjusts the calibration to best fit the data. The improvement in accuracy resulting from the refinement is measured as a percentage of a combined decrease in the AccBar length variability and projection ray convergence values (see 6.4.1).

Upon successful refinement (resulting in a non-zero improvement), the application provides a choice of permanently modifying the calibration by updating the camera's calibration file. Once saved, the modifications to the camera's calibration will be visible to the MTC library. A brief record of calibration refinement can be viewed by selecting <**Display/Calibration info**> in the VB6 Demo:


Calibration Info

Camera model: Hx40 with protective enclosure
IR Filter: (no IR Filter)
Camera serial #: 10090013
File version: 8
Calibration date: 25 Aug 2010
Performed by: Ahmad Kolahi
Location: 120 Carlton, Toronto
Calibration machine #: 1.0
Machine last calibration date: 03/07/2008
Gain F: 1.00, Shutter ms: 3.53
Camera temperature : 30.3 °C
Light coolness : 0.16
Calibration program version: 3.6.0
MTDevInterface.dll version: 3.6
MTC.dll version: 3.6
Camera firmware version: 0.9.0.44
Input locations count: 32149
Inputs X range: -523.2 to 513.9 mm
Inputs Y range: -268.5 to 298.7 mm
Inputs Z range: 387.4 to 837.7 mm
Max warmup table DPH: 7.5
RMS error: 0.179 mm
95% confidence interval: 0.350 mm
Calibration refinement date: 23 Nov 2010
- AccBar length variability: 0.25 mm
- Projection ray convergence: 0.04 mm

If an attempt to refine the calibration failed, the user is provided with an option to retry the procedure by collecting a new set of AccBar samples (this will restart R-Fine), or exit the application. Refinement failure is usually an indication that the camera's calibration hasn't yet undergone a significant degradation in accuracy. This outcome should also be supported by a low measure of length variability before and after the refinement (typical variability results for a recently calibrated camera are in 0.25-0.35 mm range for S60 and Sx60, 0.20-0.30 mm range for H40 and Hx40, 0.35-0.40 mm range for H60 and Hx60, 0.20-0.35 mm for H3-60).

8.7 The log file

During its execution, R-Fine generates a log file (serial#_mm_dd_yy.log), written to the same directory as the executable. The log records some additional information that can be used for more detailed calibration analysis as well as troubleshooting. The contents of the log file include:

- date of the R-Fine procedure,
- camera's calibration information, including its serial number and model,
- coordinates of FOM regions,
- number of collected samples, per region and per orientation,
- average AccBar length, computed over the sample set, before and after the refinement,
- standard deviation of AccBar length, before and after the refinement,
- projection ray convergence (intersection "gap"), before and after the refinement,
- length average and standard deviation values per orientation, before and after the refinement,
- internal parameter values used by the refinement algorithm.

9 Marker Design

9.1 Introduction

A marker is made of one or more patterned surfaces rigidly attached to, or printed directly on, an object, usually either a reference body or a tool of some sort.

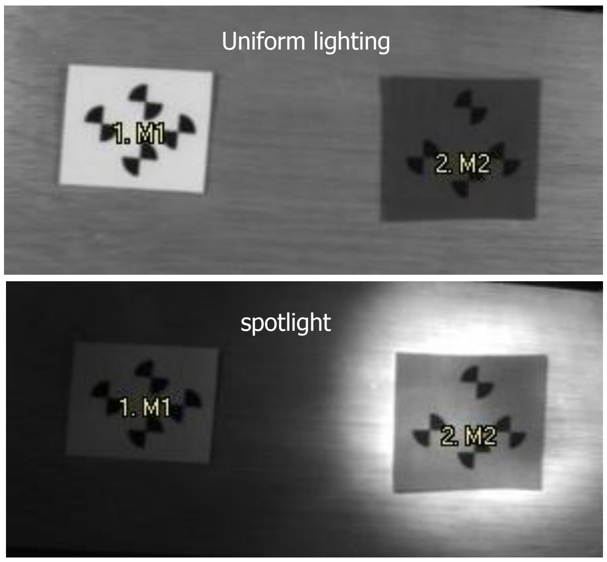
Marker design is an art. While there are rules that limit, to some degree, what designs will be identified and measured by the MicronTracker, there is much room left for creativity in finding the best design for a particular object used in a particular application setting. The purpose of this section is to guide the marker design process. Terms used in this section are defined in section 1.4 above.

9.2 XP contrast

For best visibility and accuracy under typical lighting conditions, the pattern making up each XP should be printed using the highest possible contrast between the dark and white regions, at the highest possible resolution. The surface should be as non-reflecting (matte) as possible. The brightness of the surface should not be easily affected by contact with human hands. If it would be used for along period, it needs to be easy to clean and, as necessary, sterilize.

NOTE: *Non-reflective XPs with deep black and bright white regions not only provide the best identification reliability, but also improve accuracy by ensuring each region appears highly uniform in the video images.*

Under intense or mixed lighting conditions, Xpoint contrast may be reduced to allow its detection in a situation in which it may otherwise be over-exposed ("washed-out"). For example, XPs printed in black on a 33% gray background (1/3 the brightness of white) would still be detected concurrently with a white-background marker even when it is illuminated by a light that is 15 times more intense (see pictures below).



NOTE: *Slightly larger black/gray Xpoint regions are required to obtain the same maximum detection range as black/white Xpoint regions.*

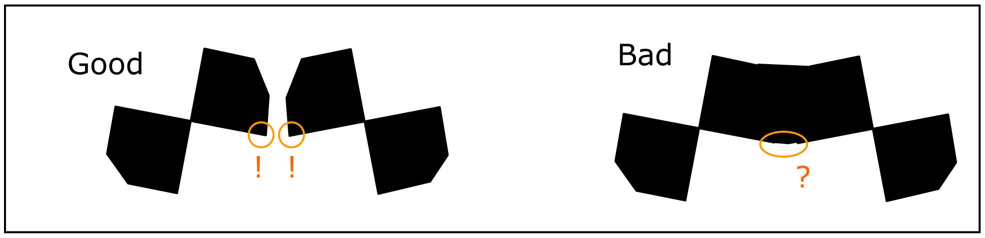
9.3 XP flatness

The surface on which XPs are printed should be flat. A slightly curved Xpoint region would still be detected, but may produce position reports that will vary somewhat depending on the orientation of the Xnormal relative to the camera. This small drift may still be acceptable for some applications.

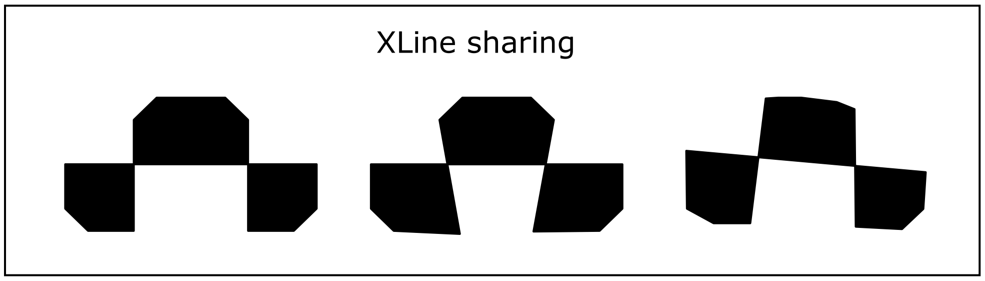
9.4 Xlines

The two Xlines do not need to be perfectly perpendicular to one another, but this is desirable for maximum angular detection range.

To eliminate measurement errors resulting from incorrect determination of the extent of the Xpoint region, Xlines should preferably terminate at a sharp angle. This is especially important when two Xpoint regions are placed next to each other at an angle.



If the neighboring Xpoints are a mirror image of each other, it is often useful to have them share an Xline as shown below.

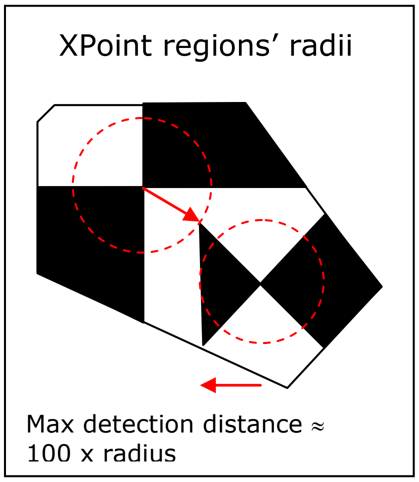


9.4.1 Size and shape

Maximum Xpoint detection distance is linearly proportional to the radius of the Xpoint's region. This radius is defined as the shortest distance between the Xpoint itself and the nearest point on the external edge of the uniformly white or black regions creating it. Increasing Xpoint region size therefore increases the field of measurement (FOM). While the shape with minimal area for a given radius is a circle, it is perfectly acceptable for the Xpoint region to be square or any other shape.

As a rule of thumb, <u>the maximum detection distance of an Xpoint is roughly 100 times its radius</u>. For example, a 5mm radius would allow detection up to 50cm, and a 12mm radius would provide detection up to the far limit of the FOM.

Xpoint position measurement errors are also linearly proportional to the distance from the camera. Beyond a certain distance, those errors become larger than the marker template matching tolerance (controlled by the property *TemplateMatchToleranceMM* of the *Markers* object, see 11.3) making marker identification unreliable.



NOTE: *While it may be tempting in some situations to increase the FOM depth by using larger XPs, keep in mind that at depths larger than the specified FOM, measurement errors grow rapidly, roughly proportionately to the square of the distance.*

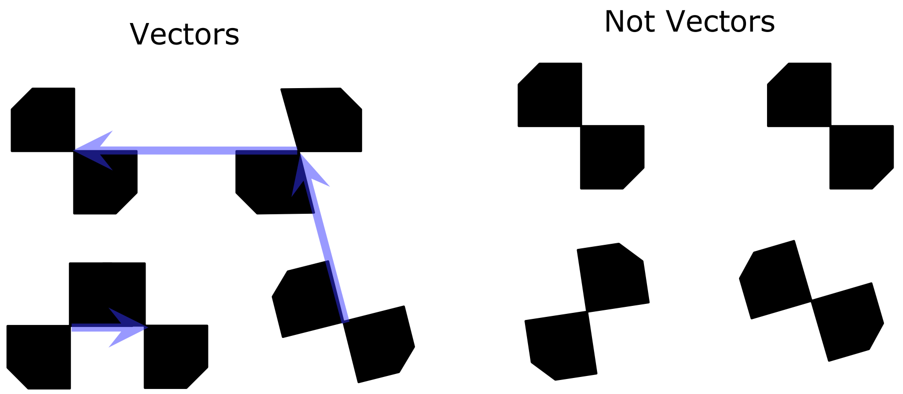
9.5 Facet Design

9.5.1 Design rules

To be recognized as a unique facet, the facet pattern design must comply with 4 simple rules:

- A facet contains exactly two vectors.
- The angle between the vectors is between 8 and 172 degrees.
- One of the facet's vectors is longer than the other by more than twice the value of `Markers_TemplateMatchToleranceMM` (see 11.3). The default factory value of this property is 1mm, which implies a difference of more than 2mm is required).
- The position of at least one end of its two vectors (in the facet's own coordinates) needs to be different than the corresponding end position in all other facets registered in the marker templates database by at least twice the value of `Markers_TemplateMatchToleranceMM`.

NOTE: *By definition, to form a vector, one of the Xlines of one Xpoint must be aligned with one of the Xlines of the other Xpoint, with black facing black and white facing white. The Xpoint regions at both ends should lie on the same plane. The MicronTracker software uses a tolerance of 5 degrees between the Xline and the line connecting the two Xpoints in recognizing Xline alignment in the Xpoints' projection in an image.*



NOTE: *Two XPs sharing an Xline automatically form a vector. This vector, however, does not necessarily have to participate in the facet to which the XPs belong. The marker templates dialog of the VB demo provides a simple way to ignore such vectors when registering a template by setting a minimal vector length threshold (see 7.6.1).*

The facet pose (coordinate transform) reported by the MicronTracker software is the one that minimizes RMS error between the facet template positioned at the reported pose and the actual measured positions of each Xpoint. Therefore, all else being equal, the more Xpoints in a facet, the lower the overall pose measurement error. Since two vectors can either share or not share a single Xpoint, the number of Xpoints in a single facet is either 3 or 4. It is possible, however, to construct a flat marker with any number of Xpoints higher than 3 (except 5) by combining multiple facets into a multi-facet marker (see 9.6.2 below).

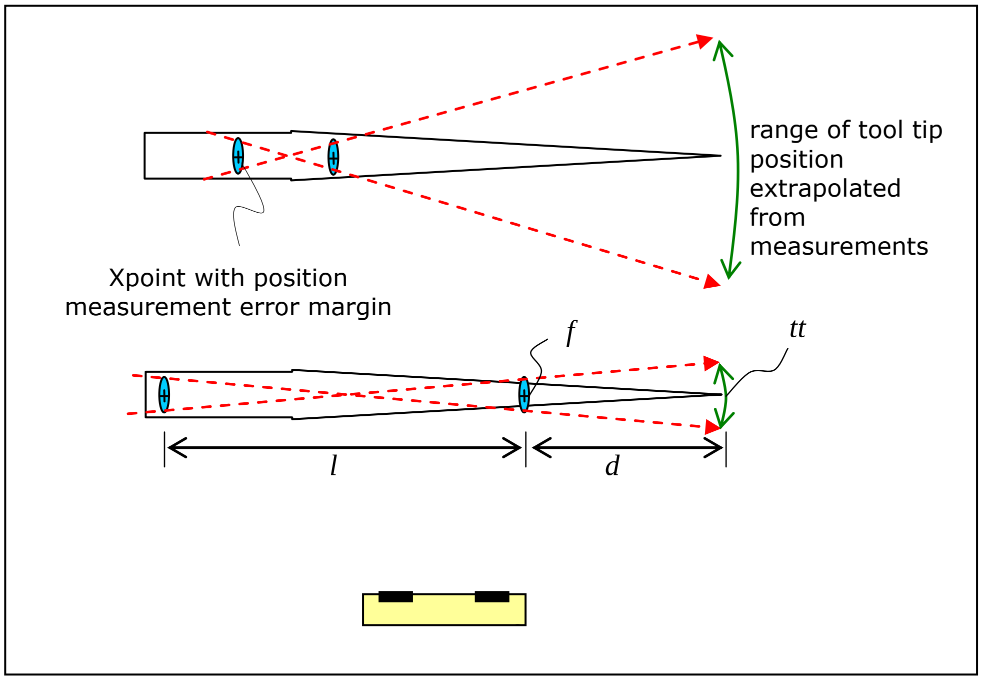
9.5.2 Tooltip accuracy

Due to the lever effect, the farther apart the Xpoints at the ends of a vector, the lower the error in measuring the vector's orientation in space, and, therefore, the lower the error in extrapolating object positions of interest, such as a tool's tip. The drawing below demonstrates this effect by showing a tool tip position error range for two different placements of Xpoints. To minimize extrapolation error, one end of the longer vector should be placed as close to the tooltip as possible, and the other as further away from it as possible.

Assuming the tool tip is approximately aligned with the long vector of the marker, and the vector's length is *l* (refer to the diagram below), and given a position RMS error at each marker Xpoint *e(f)* and a distance *d* between f and the tooltip point tt, the tooltip RMS error *e(tt)* is approximately:

$e(tt) \approx e(f) + 1.5 * e(f) * d/l$

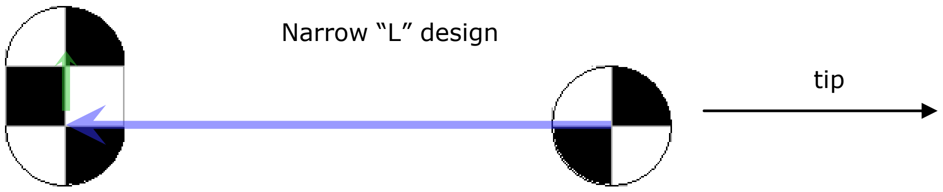
If *d* is small relative to *l*, the tip error is dominated by the Xpoint error (roughly, calibration error + jitter, typically in the range 0.2mm-0.4mm RMS). If *d* is large relative to *l*, the error is dominated by the ratio *d/l*. As a rule of thumb, to ensure that RMS error at the tip is sub-millimeter, keep $l \geq d$.

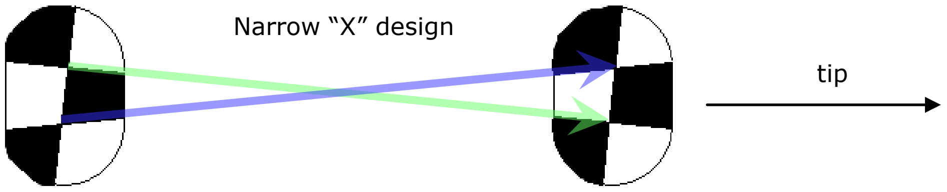

range of tool tip position extrapolated from measurements
Xpoint with position measurement error margin
f
tt
l
d

9.5.3 Facet design alternatives

9.5.3.1 For tooltip sensing

When the marker's purpose is to provide the tooltip sensing, and the orientation is unimportant or secondary, two simple designs, shown below, should be considered.



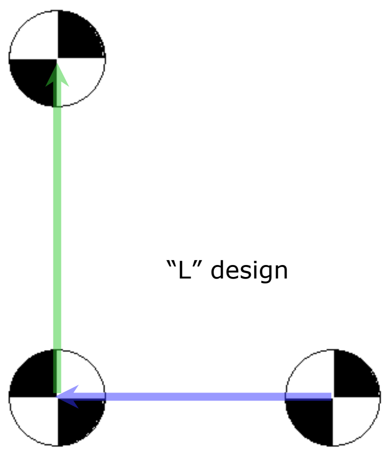


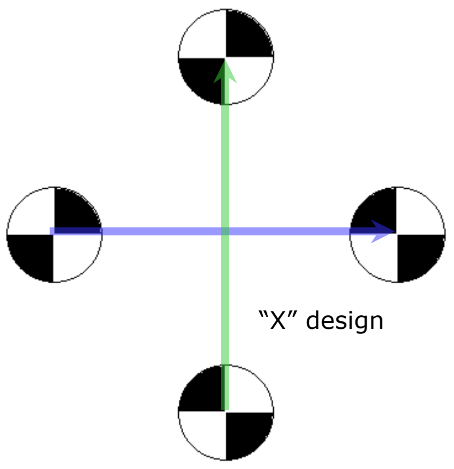
The L design provides the most efficient use of marker area, and is thus preferable when minimizing marker size is paramount. Accuracy degrades quite rapidly away from the X axis direction (the blue arrow), and it is thus very important to place the marker such that the X axis will pass through, or very near, the tip.

The narrow X design requires a larger area at the front, but reduces tip position RMS error, by 10-20% when the tip is placed along the X axis, and by up to 50% when the tip is off-axis. It also improves the roll angle sensing accuracy. Note that the angle between the two vectors needs to exceed 8 degrees.

9.5.3.2 For general pose sensing

When angular measurement accuracy is equally important in all directions, the following two simple designs should be considered.





9.6 Multi-facet Marker

A marker may contain an arbitrary number of facets. Unlike vectors within a single facet, different facets cannot share Xpoints (and, therefore, vectors).

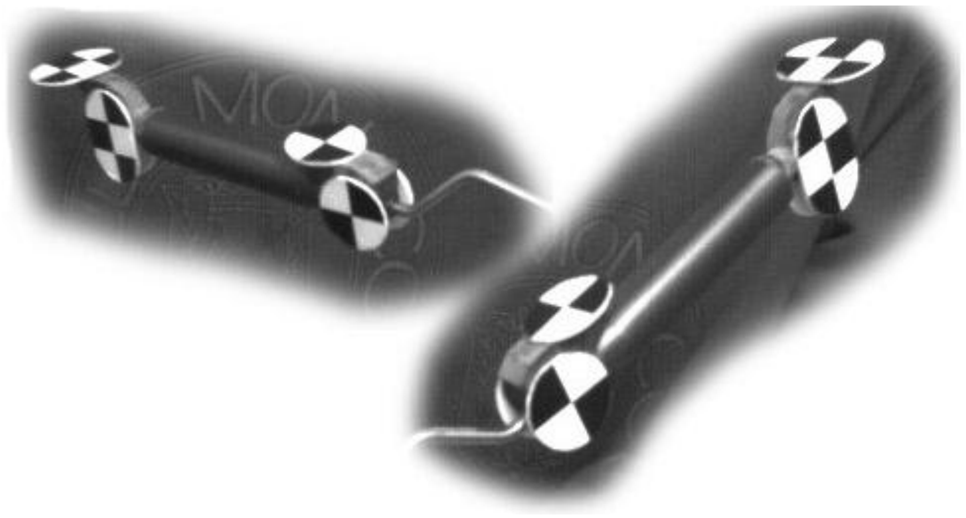
There may be two good reasons for designing a marker with more than one facet:

- Increasing the detection angle range by placing facets in different orientations.
- Increasing pose measurement accuracy by increasing the number of Xpoints contributing to the pose calculation.

NOTE: *While Xpoints may be shared by vectors of the same facet, they cannot be shared by vectors of different facets. Each facet requires its own set of Xpoints.*

9.6.1 Increased detection angle

When the Xnormal is tilted away from aiming at the lens, the Xpoint region's projection on the corresponding sensor shrinks by the cosine of the tilt angle. As the Xpoint region's projected image size falls below the minimum needed for detection (11 pixel units diameter), MicronTracker cannot continue tracking that Xpoint. Once this happens to one of the facet's Xpoints in one of the two sensors, tracking of that facet it lost, limiting the angular detection range of that facet. Adding more facets to the marker, each facing in a different direction, increases the angular detection range to the combined angular ranges of all its facets. For example, placing mirror images of a facet on the right and left sides of a tool allows it to be used with the camera either on its left or on its right. Adding another facet on top allows detection from above as well, providing the full angular detection range needed in practice. The picture below shows two views of an example of such an arrangement. The three facets contain 3 Xpoints each.



9.6.2 Increasing pose measurement accuracy

Due to the error distribution algorithm the MicronTracker uses in matching measured XP positions to their corresponding marker template positions, increasing the number of XPs tracked in a marker increases the marker's pose measurement accuracy. A single facet can contain a maximum of 4 XPs. By placing multiple facets on the same surface it is possible to create surfaces with an arbitrary number of XPs (except 5). This is especially useful when creating a reference marker, which defines the coordinate system for other markers nearby.

Error distribution is approximately normal in shape. In theory, RMS error magnitude should thus decrease proportionally to the square root of the number of XPs, e.g., a position measurement derived from 9 XPs should be roughly twice as accurate as one calculated from 3 XPs. This calculation, however, implicitly assumes that all XPs have a similar influence on the position's calculation. In practice, some XPs have a much greater influence than others, and, therefore, increasing the number of XPs may not necessarily have the expected effect. For example, when extrapolating to a position away from the marker, e.g., a tool tip position, XPs located farther away from the position, or at the ends of a vector perpendicular to the extrapolation direction, have a smaller impact on that measurement.

9.7 Reference marker design

A reference marker is used to establish a coordinate system anchored to an object being measured, so that tooltip measurements would be made relative to that object, rather than relative to the camera. This eliminates the potentially large measurement errors caused by either object or camera movements during a measurement session. For example, attaching a reference marker to the head during a sinus surgery procedure allows the surgeon to conveniently move the head during the procedure without requiring a re-registration. Furthermore, if line-of-sight interferences are detected during the procedure, the surgeon is free to reposition the camera, again without the need for a re-registration.

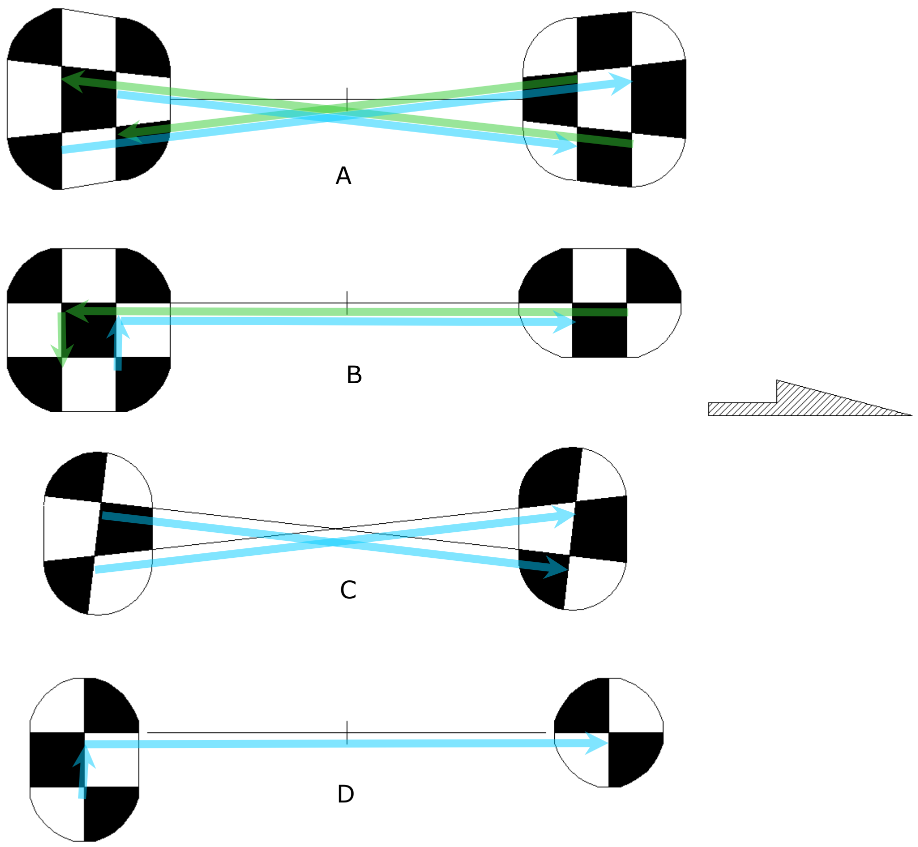
When considering how to design a reference marker, it is useful to think of the region of interest (ROI) for measurements (e.g., the sinuses in the abovementioned example) as the "tooltip" and of the region where the reference would be attached as the tool "handle". The guidelines and formulas for tool marker design would then similarly apply. For example the formula for computing accuracy at the tooltip (see 9.5.2) could be used to compute accuracy at the region of interest.

Some general guidelines for obtaining maximum accuracy in a reference marker design:

- Interpolation is much more accurate than extrapolation, so placing the XPs of the reference marker *around* the ROI provides the highest possible accuracy.
- If placing XPs around the ROI is not possible:
    - Try to place as many XPs as possible as close as possible to the ROI.
    - At least two XPs should be placed as far away as possible from the ROI and from each other to maximize angular accuracy.

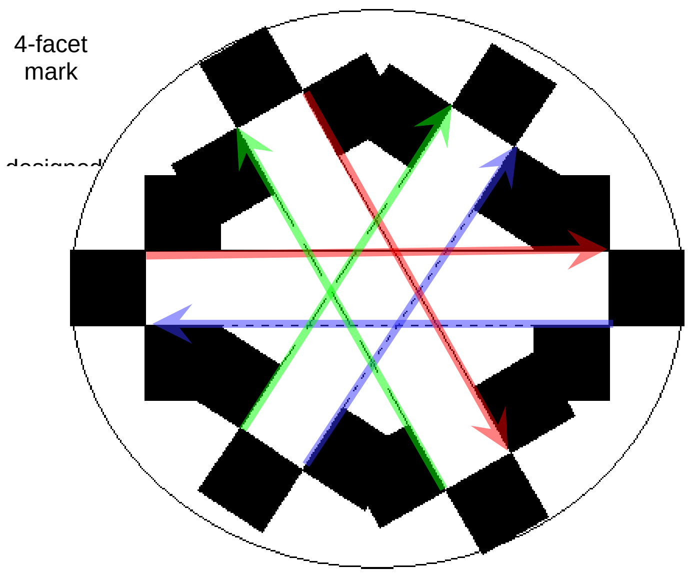
9.8 Design examples

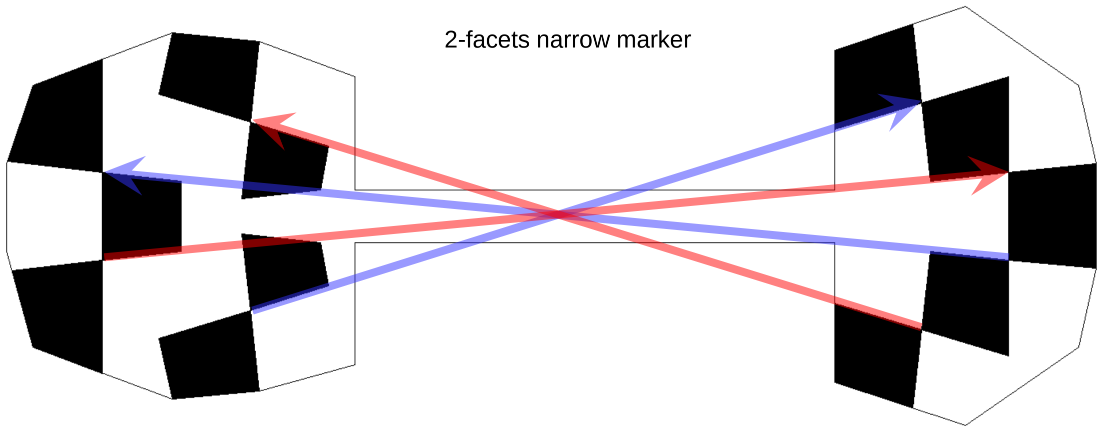
Below are shown 4 different patterns for a 2-part marker designed to be mounted along the handle of a tool and provide maximum accuracy at the tip of the tool, which is extended along the narrow dimension of the marker (see arrow). A and B use two facets (8 and 6 XPs), while C and D use only a single facet (4 and 3 XPs). The two vectors of each facet are marked with similar-colored arrow overlays. Compared to design D, the smallest and simplest, RMS error at the tip of the other designs was measured to be about 55%, 65% and 75% for A, B and C correspondingly.



Next are two more examples of multi-facet markers to be used to provide a reference coordinate system for other markers. The top one is a circular marker containing 12 Xpoints organized into three facets. The vector directions are distributed such that accuracy would be nearly equal in all directions. When visually registering this marker (see 7.6), initially partially cover the XPs of all the facets except one, and then add the other two facets, one at a time, by incrementally removing the covers. Set a vector length threshold to hide the short vectors along the shared Xlines.

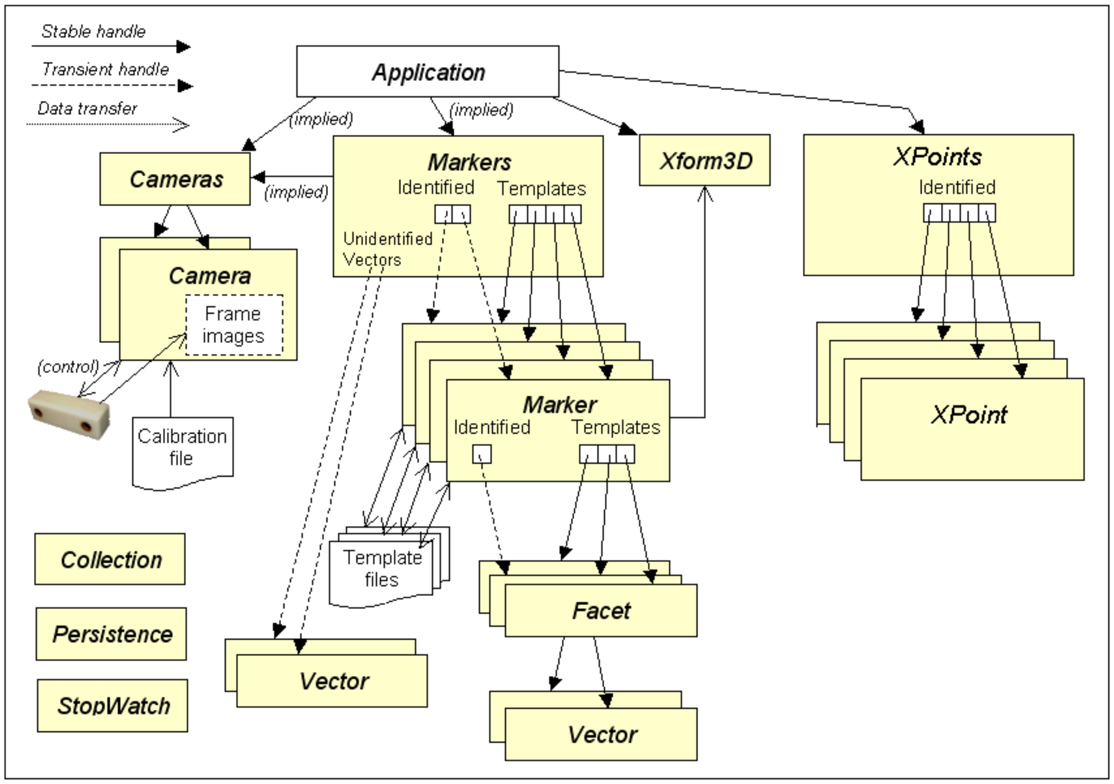
The design below is intended for a marker to be placed to one side of the region in which measurements are performed, and is, therefore, designed to provide increased angular accuracy in the direction of that region.





10 Programming Interface

10.1 Programming Model



The figure above shows the object classes used in controlling the sensor and obtaining pose measurements. One object of each of three classes, *Cameras*, *Markers* and *XPoints*, is automatically created when the interface library starts running. The application refers to those instances directly by their class name without the use of an instance handle.

The Cameras object represents the array of MicronTracker cameras performing pose measurements. Calling `Cameras_AttachAvailableCameras` establishes the communication link with all the cameras physically connected to the computer and loads the proper calibration file for each of the cameras. Handles to individual camera objects representing attached cameras may be obtained using `Cameras_ItemGet`. Each camera can then be individually accessed and controlled. However, the most common operations relating to cameras can be called collectively through the *Cameras* interface. Upon terminating the use of cameras, `Cameras_Detach` must be called to signal the driver to disassociate itself from the application process.

The *Markers* object represents all the markers being potentially tracked by all cameras, i.e., markers with a registered template. Each marker is represented by a separate *Marker* object, which, in turn, maintains a collection of Facet objects, one per facet, in its template. Each *Facet* object maintains two *Vector* objects. A

Marker object also optionally maintains a `Tooltip2MarkerXf` *Xform3D* object describing the pose of a tool-tip in the marker coordinates.

The *XPoints* object represents an array of all the XPoints being detected by all cameras. Each XPoint is represented by a separate *XPoint* object and can be individually accessed and provides 3D spatial position and all 2D information of detected XPoint in the left and right images.

The *Collection* class represents a variable-length 1-based array of numbers, usually object handles. It was designed to be roughly compatible with the ActiveX class of the same name. The *Persistence* class represents a collection of variables (name-value pairs) stored in a file. The file format is with the used for .INI files in MS-Windows, and uses plain text, which can be easily manually edited. The *StopWatch* class provides a convenient and accurate way of measuring elapsed time.

Marker templates are stored in *Persistence* files, typically one per marker. The *Markers* object creates and maintains an array of marker objects, each representing a template. Templates may be added individually to the collection by the client, or a single method, `Markers_LoadTemplates`, may be used to load all templates in a given directory.

There are two different measurement cycles, *Markers Processing* and *XPoints processing*:

1. *Markers Processing*: A measurement cycle for Markers Processing includes the following steps:

- Calling `Cameras_GrabFrame` to transfer the latest frame to the MTC's buffers;
- Calling `Markers_ProcessFrame` to extract measurements from the frame buffers;
- Calling `Markers_IdentifiedMarkersGet` to obtain a collection of handles to the identified markers;
- For each identified marker in the collection, calling `Marker_Marker2CameraXfGet` to set the attributes of an application-owned Xform3D object to the pose of the marker. Optionally, the tooltip pose, obtained via `Marker_Tooltip2MarkerXfGet`, may be concatenated to the marker's pose to obtain the pose of the tooltip in the camera's coordinates.
- Accessing the *Xform3D* attributes to perform application-specific operations.

NOTE: *To speed up frame grabbing and markers processing, MicronTracker can also do all the computation in the background when enabled with Markers_BackGroundProcessSet().*

This will make MicronTracker grab and process frames automatically in a background thread and return to the user the most recently processed pose data when requested.

2. *XPoints Processing*: A measurement cycle for XPoints Processing includes the following steps:

- Calling `Cameras_GrabFrame` to transfer the latest frame to the MTC's buffers;
- Calling `XPoints_ProcessFrame` to detect each individual XPoints from the frame buffers;
- Calling `XPoints_DetectedXPointsGet` to obtain a collection of handles to the detected XPoints;
- For each detected XPoints in the collection, calling `XPoint_2DPositionGet` and `XPoint_3DPositionGet` provide 3D and 2D position data of the XPoint.
- Vectors which were recognized in the frame(s) but not identified as being part of a marker (i.e., did not match any template), can also be obtained by calling `Markers_UnidentifiedVectorsGet`.

<u>NOTE:</u> *It is critical that `Cameras_Detach` is called before the application program terminates. Otherwise, the camera driver may keep the application process running for a few more minutes before releasing it, preventing access to the camera by any other process. In an interactive development environment, such as VB, terminating the application program without calling `Cameras_Detach` would frequently crash the development environment process.*

10.2 Detailed C API Description

MTC supports over 200 function calls, enabling the application to tightly control and modify many aspects of MicronTracker's hardware and software operations. Those functions are documented in a set of hyperlinked web pages that can be accessed from Windows task bar (*Start → Programs → Claron Technology → MicronTracker SDK documentation*) or from the VB6Demo menu (Help → MTC API documentation). The "home" page is in *{$MTHome}\doc\html\index.html*, where {*$MTHome*} stands for the path where the MicronTracker software was installed (by default, *C:\Program Files\Claron Technology\MicronTracker*).

10.3 Source Code Examples

Following installation, the selected installation folder should contain a number of demonstration applications, each in its own sub-folder:

- **MT_SimpleDemoC**: A barebones example that shows a basic of operations for obtaining measurements from a single or a multi- camera set.
- **MTDemoCPP**: A multi-platform demo for both Win32, Linux and Mac OSX. It is not as complete as the MicronTracker VB.Net demo, but it shows how to perform the fundamental MicronTracker tasks, including
    1. loading pre-existing marker templates,
    2. interactively generating new templates,
    3. recognizing markers, and

4. displaying positions and distances between markers.

The UI has been implemented using FLTK (www.fltk.org) a public domain multi platform toolkit. The video and text display is done using OpenGL or FLTK depending on configuration parameters. Note that the display of images and overlays is just an example implementation where being cross-platform, rather than top performance, was given the highest priority.

- **MTDemosNet**: Uses VB.Net and C# .Net to show how to use MicronTracker functionality in Microsoft .Net environment. MTC is accessed via a wrapper library (MTInterfaceDotNet.dll). The wrapper library is included, as a separate reusable project, in source format.

The MTDemosNet VB.net demo functionality is described in section 7. Since Vb.Net is relatively straightforward to read and understand, it is intended to serve as a general-purpose presentation of the manner in which application programmers should use the MicronTracker API in other environments as well.

- **MTDemoVB6**: Demonstrates a wide range of MicronTracker functionality using Visual Basic 6 code. MTC is accessed via a wrapper ActiveX library (MTinterface.dll). The wrapper library is included, as a separate reusable project, in source format. The ActiveX API is very similar to the one provided in earlier product versions.

10.4 The SimpleDemoC Application

This application demonstrates the very basics steps required to use a MicronTracker to detect and report positions. It is a Windows console application, which loads pre-saved markers templates, initializes the camera and then loops for a few seconds reporting the position of the recognized markers on the output text window. It implements some of the basic steps described in the introduction to the Programming model paragraph.

On **initialization**, the application

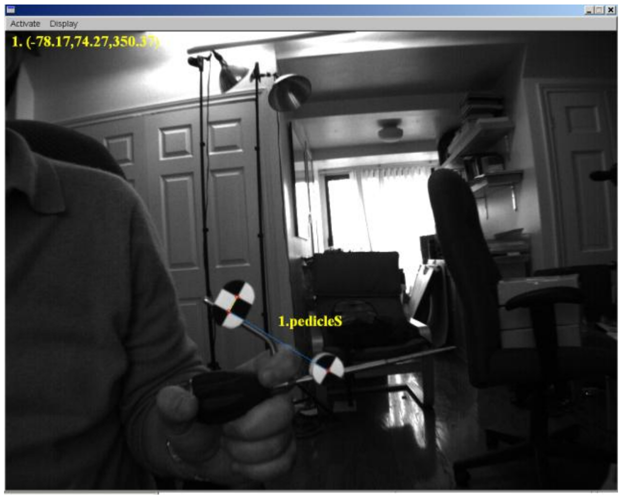
- Calls `Cameras_AttachAvailableCameras` to activate the cameras and load their calibration files from disk; and
- Calls `Markers_LoadTemplates to l`oad all the marker template files.

To obtain a **measurement**, the application

- Calls `Cameras_GrabFrame` to obtain the most recent frame images from the camera(s);
- Calls `Markers_ProcessFrame` to update the collection of Identified markers.
- Obtains a collection of the handles of the identified markers by calling `Markers_IdentifiedMarkersGet;`
- Accesses the pose of each identified marker by using `Marker_Marker2CameraXfGet` to set the properties of a *Xform3D* object accordingly, then prints the name, position and rotation of the marker using properties of the marker and the *Xform3D* object.

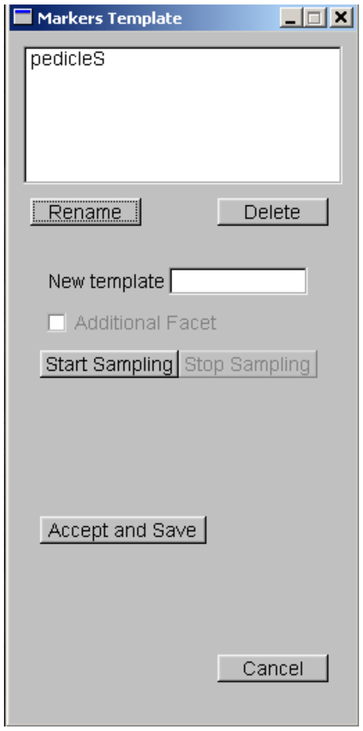
On exit, the application calls `Cameras_Detach` to release the cameras and allow the process to terminate.

10.5 The MTDemoCPP Application



MicronTracker ships with a C++ example application that is available both in executable and in source format. By default the application is installed under *Program Files\Claron Technology\MicronTracker \MTDemoCPP*. A shortcut to this application is placed on the desktop and it is also available through *Start* → *Programs* → *Claron Technology* → *MicronTracker*. This application is multi-platform and supports both Windows and Linux. Its feature coverage is not as complete as the VB6 demo, but it shows how to perform the fundamental MicronTracker tasks:

- loading pre-existing marker templates,
- interactively generating new templates,
- recognizing markers,
- displaying positions and distances between markers.

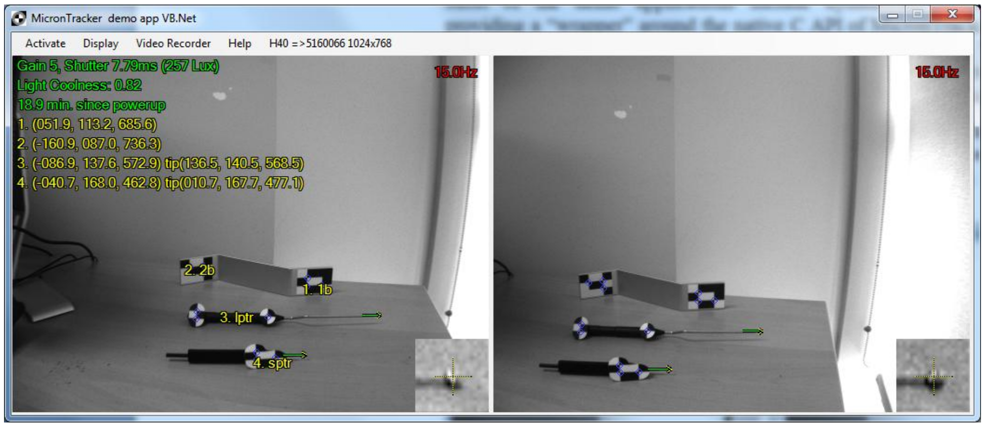


Aside from the main window where video images and detected markers are displayed, it also has a panel that can be used to record new marker templates. The existing templates are listed in the panel and can be deleted or renamed.

Of particular interest is that the application is implemented using a C++ wrapper over the MTC "native" API, making it easier to integrate into a C++ development environment. The wrapper is not complete, but it covers sufficient functionality for most basic applications and can be easily extended following the established patterns.

10.6 The MTDemosNet Application

This application uses Microsoft .Net environment to demonstrate a wide range of MicronTracker functionality. The applications are written in VB.Net and C#. By default the applications are installed under *Program Files\Claron Technology\MicronTracker*. A shortcut to the VB.Net application is placed on the desktop and it is also available through *Start* → *Programs* → *Claron Technology* → *MicronTracker*. This application demonstrates usage of most of the MicronTracker API, and enables performing many measurement operations. A user can register markers by showing them to the camera, record distance and angle measurements between markers as well as collect accuracy statistics, register multiple cameras to create a combined FOM, take frame snapshots, and produce an AVI time-lapse video of a measurement session. The user can also modify various camera settings and immediately view their effect.



```
Gain 5, Shutter 7.79ms (257 Lux)
Light Coolness: 0.82
18.9 min. since powerup
1. (051.9, 113.2, 685.6)
2. (-160.9, 087.0, 736.3)
3. (-086.9, 137.6, 572.9) tip(136.5, 140.5, 568.5)
4. (-040.7, 168.0, 462.8) tip(010.7, 167.7, 477.1)
```

11 Advanced Topics

11.1 Camera Exposure Controls

11.1.1 Exposure, Gain and Shutter

To enable detecting and correctly pinpointing an Xpoint in the image, a minimal contrast of about 20 pixels values between its dark/bright regions is necessary. Furthermore, its bright regions should not be over-exposed, i.e., the pixel values there should fall below the maximum pixel value (255). The range of light intensity encountered during operation is extremely wide (50-100,000 Lux), making it necessary for the MicronTracker software to frequently adjusting the camera exposure settings, which is represented by the *Exposure* property of the *Camera* class.

The *Exposure* property value is actually a product of two other properties, *GainF* (gain factor) and *ShutterMsecs*. *ShutterMsecs* corresponds to the length of the period, in milliseconds, in which each pixel in the sensor is exposed to photons before a measurement is taken (in the range 0.13 to 33 ms for the S60, 0.17 to 66 ms for the H40, 0.17 to 67 ms for the H60 model or 0.01 to 67 ms for H3-60 and Sx60 on 15fps mode) 0.01-50 ms for Hx40 and Hx60. The *GainF* parameter controls the amount of amplification the analog signal received from each pixel undergoes before being converted into a digital value (in the range of 1-20 for H40, 1-32 for S60, 1-24 for Sx60, Hx40 and Hx60 and 1-22 for H60). Gain is similar to the ISO rating of a film. A higher gain increases the sensitivity of the camera, but also increases the noise level. Noise level, and with it measurement jitter, increases proportionally to the square root of the gain factor (e.g., a gain factor of 4 will produce roughly twice as much jitter as a gain of 1).

In a given lighting situation, a range of different combinations of shutter and gain can be used to achieve a similar distribution of gray levels in the image. Lower gain will result in lower jitter. However, under low lighting conditions low gain values will also result in longer shutter periods, in turn blurring moving objects and causing potential loss of tracking during marker movement.

11.1.2 High Dynamic Range Mode (HDR)

High dynamic range mode (HDR) is a technique that allows solid detection in situations of great illumination range. . This feature enables MicronTracker to work reliably under the wide range of intensity levels found in the operating rooms. For instance H3-60 model covers a wide range of 5Lux~400,000Lux illumination.

In this mode MicronTracker assigns a cycle of frames to grab images at different exposure levels that covers the whole range of illumination in the field of view. Based on the range of illumination the cycle might have 1, 2 or 4 frames.

HDR mode might produce a latency of one full cycle in reporting the pose data, although this is generally not significant in image guided application. The additional latency might be noticeable if the camera is operating at lower frame rate. In such circumstance enabling background processing will improve the performance significantly and it will reduce the time lag.

11.1.3 Background Processing

This feature makes the MicronTracker grab and process frames automatically in a background thread. Data is processes as fast as possible and the most recently processed pose data is retrieved on user demand. This approach can result in up to 30% performance improvement in situation where there is significant application processing.

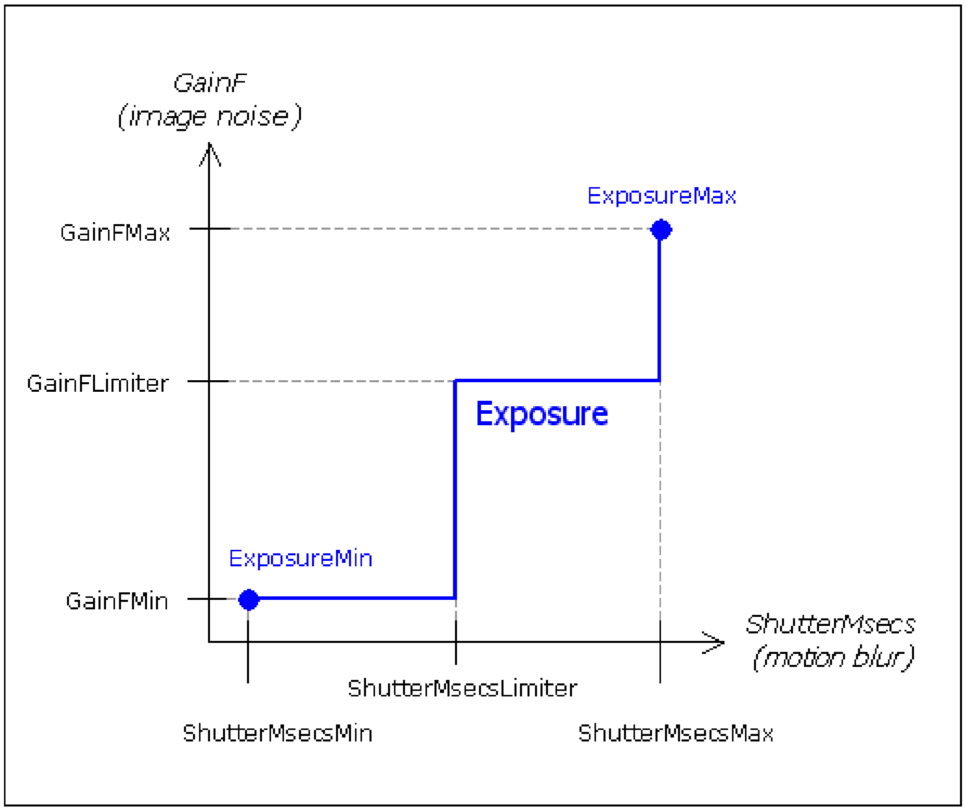
In this mode the MicronTracker does not wait to receive a command to grab and process frames. It simply grabs and processes frames as fast as possible and copies the measured pose data to a separate buffer which is accessible in the application level.

11.1.4 Automatic vs. Manual Exposure Adjustment

When the *AutoAdjustCameraExposure* property of the *Markers* object is set to true (its default value), MicronTracker employs a sophisticated algorithm in adjusting the value of the *Exposure* property based on the contents of recently captured frames. Alternatively, exposure can be adjusted by the application, which may either adjust the *Exposure* property, or go to an even lower level and adjust *GainF* or *ShutterMsecs* directly.

11.1.5 Shutter and Gain Limiters

When the *Exposure* property is set to a given value, its translation into specific settings of each of its two components (shutter and gain) is determined by two other *Camera* properties: *ShutterMsecsLimiter* and *GainFLimiter*. *ShutterMsecsLimiter*. The following diagram shows how shutter and gain are set for a given exposure value (along the blue path).



Since the shutter time controls the amount of motion blur for a given marker movement, *ShutterMsecsLimiter* should be set to the highest value in which, under typical operating conditions, tracking is still maintained. Similarly, *GainFLimiter* should be set to a value corresponding to the maximum measurement jitter acceptable.

When lighting is strong (over 500 Lux), the settings of the limiters should have no noticeable effect on tracking performance. Their effect is more noticeable in medium or dim lighting conditions, where trade-offs are required. Applications that track slow-moving objects can reduce jitter by setting *ShutterMsecsLimiter* to the maximum shutter time (33/66msecs) and *GainFLimiter* to some low value (e.g., 2). Applications that track faster-moving objects should set the shutter limiter at 20 msecs or lower, and the gain limiter to 5 or higher.

The VB6 demo application allows direct experimentation with the effects of different exposure limiters' settings through the Options dialog (see 7.9).

11.2 Overexposure Control Interleave

The automatic exposure adjustment algorithm optimizes settings for tracking markers already detected in previous frames. When new, more strongly illuminated, markers appear, they may be over-exposed ("whitewashed"), resulting in their presence not being detected. To address this situation, MicronTracker deploys a special overexposure detection algorithm to search for regions in the images which may contain over-exposed XPs, and, when they are found, to reduce exposure by an amount appropriate to allow their proper detection and tracking.

By definition, overexposed markers do not have a reliable "signature" in the image and they cannot, therefore, be detected with the level of reliability of properly exposed markers. As a result, the overexposure detection algorithm may occasionally cause unnecessary temporary dimming of the image, resulting in some increase in measurement jitter and periodic oscillations in the brightness of the image presented on the screen.

In most applications, detection of new markers appearing on the scene does not have to be immediate. In such cases, it is acceptable to activate the overexposure detection algorithm only periodically, e.g., once every 10 or 20 frames. The frequency of overexposure detection activation is controlled by the *OverExposureControlInterleave* property of the *Markers* object.

The VB6 demo application allows direct experimentation with the effects of different *OverExposureControlInterleave* settings through the Options dialog (see 7.9).

11.3 Templates Matching Tolerance

Identifying a visible facet involves comparing the measured relative positions of the ends of its vectors to the corresponding vector end positions of each facet of each registered marker template. Identifying a facet, i.e., declaring a positive "match" with a specific facet template, means that the distance between the measured positions and the corresponding template positions is lower than some tolerance. The value of that tolerance (in millimeters) is exposed to the application software through the *TemplateMatchToleranceMM* property of the *Markers* class.

To guarantee that a facet will not be mistakenly identified as a different one, or that multiple visible facets will not be matched to the same template, the difference in length between the two vectors of a facet and between relative vector end positions in different templates needs to be larger than twice the tolerance.

Increasing the tolerance reduces the likelihood that identification will fail due to small position measurement errors or due to bending of the facet. However, it also increases the required minimum length difference between vectors in a single facet and the between the relative positions of Xpoints in different facets (see 9.5). It may be useful to increase that value from its factory default setting (1mm) when measurements need to be performed on large markers at distances larger than 1.5m from the camera (which is outside the calibrated volume!), or when it is important to continue pose tracking despite some distortion of the marker geometry (for example, when the marker is placed directly on a patient's skin). It may be useful to decrease its value when a large number of slightly different markers need to be distinguished, and the measurements are performed at distances shorter than the full depth of the calibrated volume. The Options dialog of the demo application allows experimenting with the tolerance setting (see7.9)

The method *Validate* of the *Markers* class provides a quick and useful way to detect potential misidentification problems in the set of marker templates given the current *TemplateMatchToleranceMM* value. It should be executed whenever changes to either the marker templates or the value of *TemplateMatchToleranceMM* may have been made (see usage examples in the VB6 demo application code).

11.4 Predictive Frames Interleave

In detecting the presence of facets in a frame, MicronTracker uses a combination of two approaches:

- **Comprehensive detection**: The two image of the frame are comprehensively processes to detect any facet's "signature" appearance. Once a facet is detected, it is compared with each facet of each known templates to locate a match.
- **Predictive detection**: The pose of each identified facet in the current frame is predicted from past frames. A search in the current images is then done only in a small area around each of the facet's Xpoints' predicted projection location in that image.

The predictive technique requires fewer calculations to execute per frame, reducing CPU load. It does not, however, allow detection of new facets, and it does not ensure reliable tracking, especially of facets that are moving erratically.

The property *PredictiveFramesInterleave* of the *Markers* class allows the application programmer to interleave comprehensive and predictive frames. A value of 0 (no interleaving) will ensure that each frame is processed comprehensively. This is the recommended setting when sufficient CPU power is available or when measurements are done at a slow rate compared to the movement of the objects it is measuring. Otherwise, a higher value should be set to free up the CPU. For example, a value of 2 will place two predictive frames (P) after each comprehensive frame (C): CPPCPPCPP…

11.5 Reducing CPU Load

In some circumstances, sharing the CPU with the MicronTracker may have an undesirable effect on the capacity of the application to respond to external events in a timely manner. Here, we list a number of strategies that may be used to resolve this problem.

11.5.1 Adjusting Predictive Frames Interleave

Setting the property *PredictiveFramesInterleave* of the *Markers* class (see 11.4) to a value in the range 1-4 may produce satisfactory results, while saving as much as 50% of the CPU power.

11.5.2 On-demand measurement

The demonstration applications obtain measurements as fast as the camera and available CPU power allows. In case your application cannot process the measurements at the maximum rate, requesting a new measurement only when the application is ready to process them would save CPU cycles otherwise taken up by the MicronTracker.

11.5.3 Accessing the Image Buffers

The first call to `Camera_ImagesGet` or `Camera_HalfSizeImagesGet` after a new frame was grabbed requires internal processing within the MicronTracker software to prepare a properly formatted 8-bit image and to copy it to the application's buffers. If the images are not displayed on the screen, the application should avoid making such calls.

If only individual pixels or a small region in the image needs to be accessed, calls to `Camera_PixelsGet` are far more efficient than obtaining access to the full images.

11.5.4 Using a dual-CPU computer

By using a host computer with more than one CPU enables the processing of measurements to be accomplished with no effect on other asynchronous application tasks. Note that MTC is single-threaded and will not occupy more than one CPU to execute each function call.

11.5.5 Using a dedicated single-board computer

An alternative to a dual-CPU configuration is to use a separate SBC.

12 Safety and Maintenance

12.1 Standards compliance certification

MicronTracker products (H40, H60 and S60) have been certified to comply with the following standards:

- EN 60601-1-2:2001 Medical Electrical Equipment-Part 1-2: General Requirements for Safety-2. Collateral Standard: Electromagnetic Compatibility – Requirements and Tests.
- CISPR 11:2003, Class A (Emissions)
- EN 61000-4-2:1995, +A1: 1998 (Electrostatic Discharge, 8kV air; 6kV contact)
- EN 61000-4-3:1996,+A1:1998 (Radiated RF Immunity, 3V/m 80-2500Hz)
- EN 61000-4-4:1995,+A1:2001 (Electrical Fast Transients, ±1kV 5kHz)
- EN 61000-4-6:1996,+A1:2001 (Conducted RF Immunity, 3V/m, 0.15-80 MHz)
- C.F.R. 47, Part 15:2002, Subpart B, Class A

As class 8750 01 and 8750 81 medical electrical equipment class II, no patient applied parts, compliant with the following safety standards:

- IEC 60601-1 edition 2:1998
- CAN/CSA C22.2 No 601.1-M90
- CSA 601.1 supplement 1:1994, and Amendment 2:1998
- UL Std No 60601-1

MicronTracker camera model Sx60 and H3-60 have been certified to comply with the following Standards:

- Complies with CE and Part 15 Class A of FCC regulations
- RoHS and WEEE compliant

NOTE: *that camera enclosure leakage needs to be measured in the end use application.*

12.2 Safety warnings

⚠ WARNING: No Patient Applied Parts

MicronTracker is an electro-medical device of Class II, which is not to be put in contact with the patient (No Patient Applied Parts). MicronTracker has been tested to comply with FCC Part 15, Class A specification and to be conformant with EN 60601-1-2: General Requirements for Safety—2. Collateral Standard: Electromagnetic Compatibility—Requirements and Tests.

⚠ WARNING: Check enclosure electrical current leakage in end use application

Since the MicronTracker can be used with various types of power supplies, camera enclosure electrical current leakage is to be measured in the end use application.

⚠ WARNING: Do not open camera case

MicronTracker camera is optically pre-calibrated at the factory and should not be disassembled or modified in any way. Opening, or attempting to open, the camera case will compromise the measurement performance of the device and invalidate its warranty. An optional recalibration service is available from Claron Technology by contacting support at the above address.

12.3 Care

Handling

MicronTracker is a precise optical instrument and should be treated as such. The case is designed for maximum protection, but when not in use keep the MicronTracker in its foam-padded box and avoid shocks.

Cleaning

Keep the lenses free of dust or stains using a photographic lens cleaning solution and a lint free paper or cloth.

12.4 Symbols on the MicronTracker:

⚠	**Read the Operator's Manual**
⧈	**Class II medical device equipment**