An Overview of NifTK for IGI

(with a focus on calibration stuff)

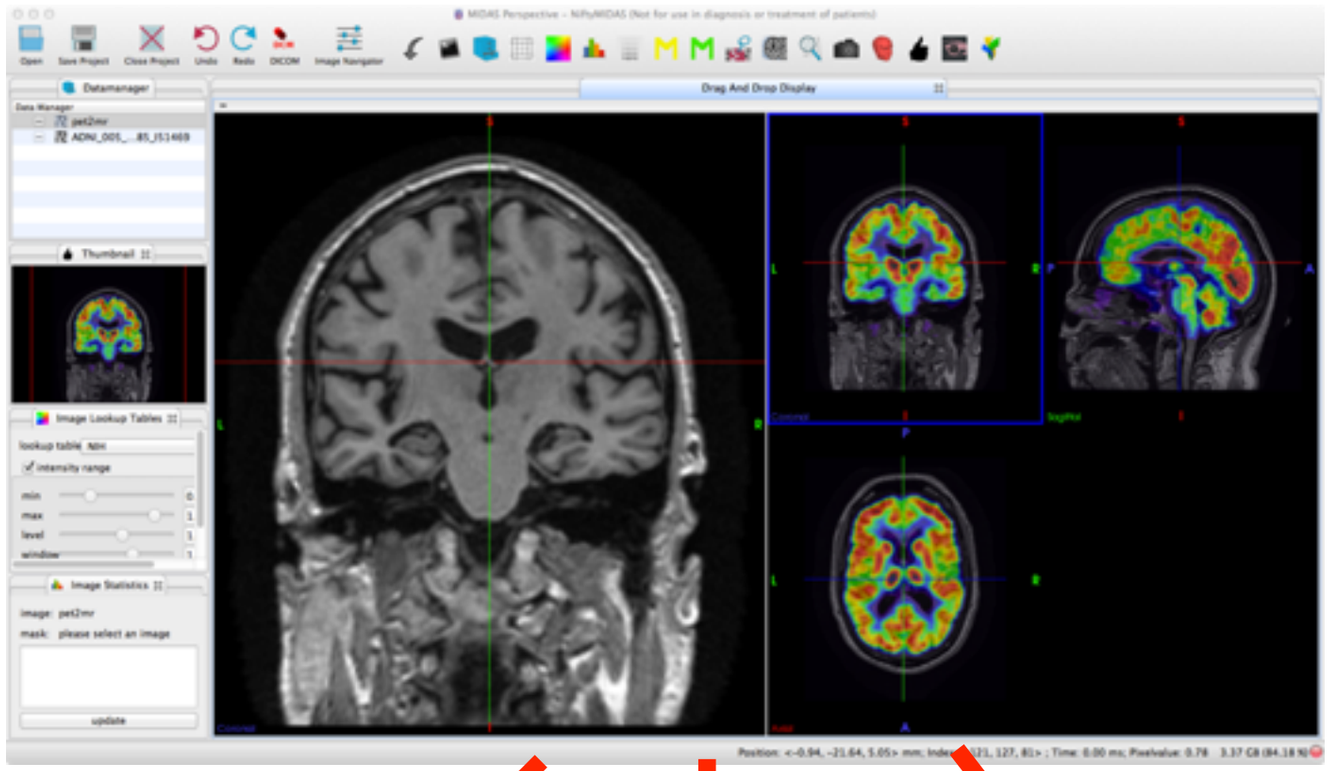
Matt Clarkson, Steve Thompson, Guofang Xiao

Translational Imaging Group
Centre for Medical Image Computing

- HICF Liver project ended - good time to review
- WTC starting - more support for core infrastructure

- Aim:
  - History of NifTK / NiftyIGI / Smart Liver
  - Overview of platform
  - Focus on some algorithms
  - How to contribute

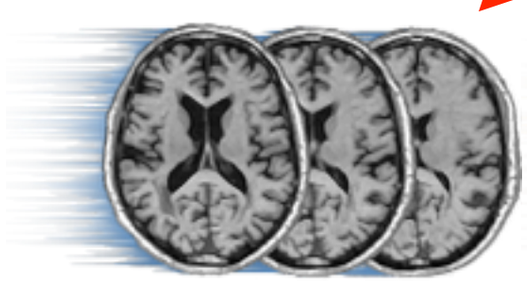
- Matt - NifTK = http://cmiclab…../NifTK.git
- Seb - NifTK = All his Software



2010 UCLToolkit

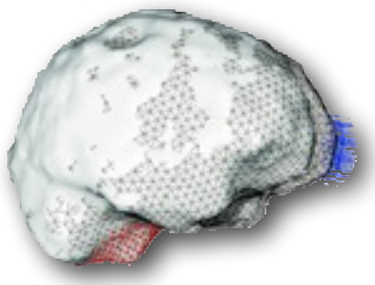
NifTK

includes



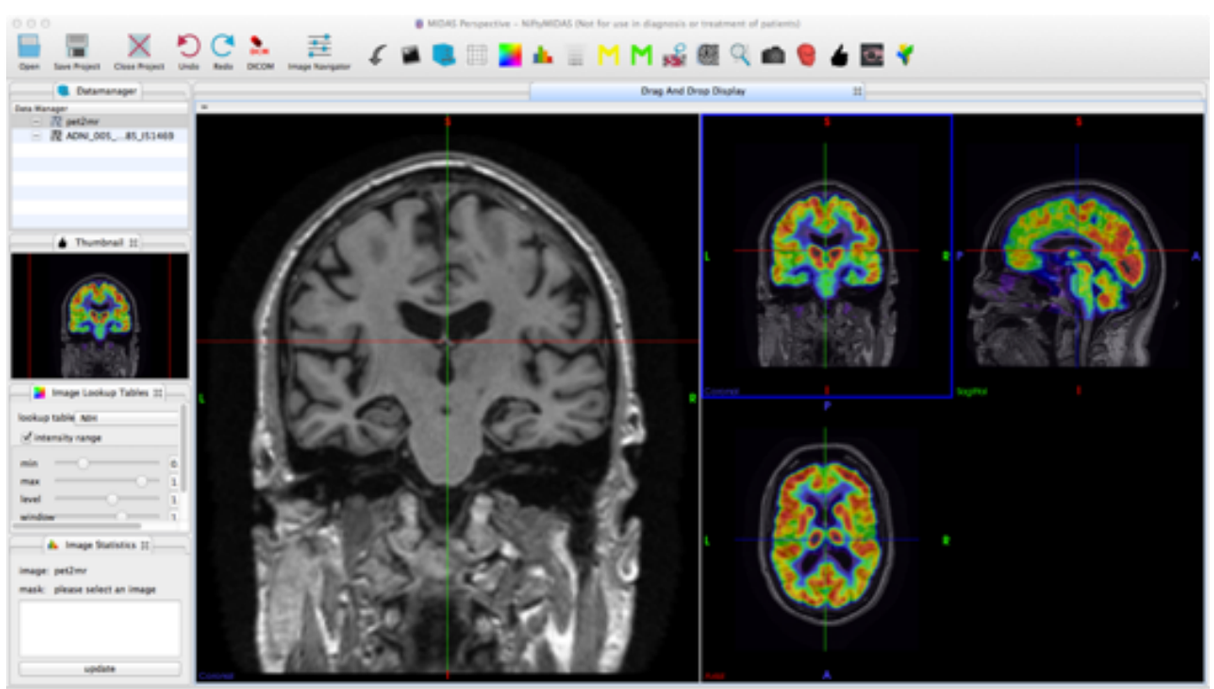
NiftyReg

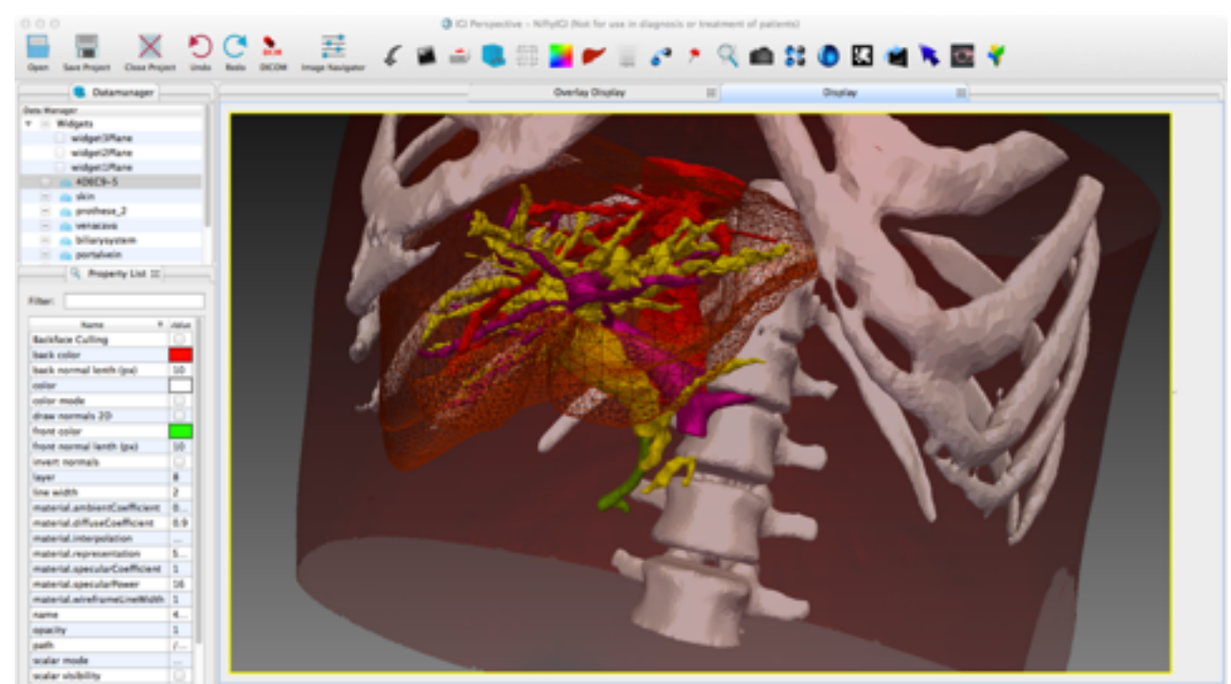
NiftySeg



NiftySim

- Aim 2012
  - Create GUI for clinical research
  - NeuroImaging (NiftyMIDAS)
    - 3D, 4D, multiple volumes, XNAT, LUT, viewer, image processing
  - IGI (NiftyIGI and NiftySLS)
    - Surfaces, 2D, video, ultrasound, realtime, interactive

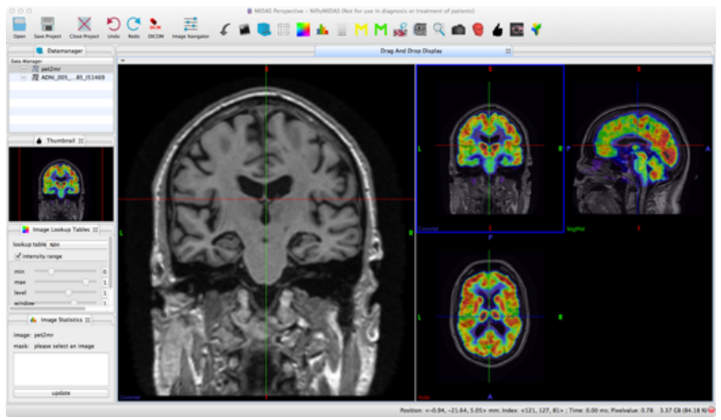




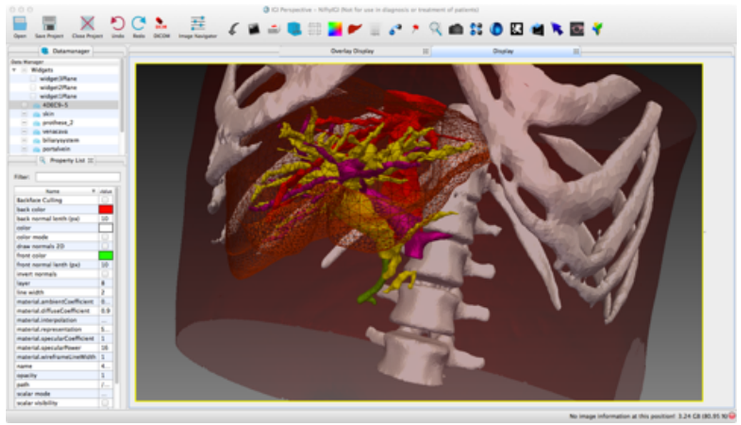
Clarkson et al. IJCARS 2014: http://dx.doi.org/10.1007/s11548-014-1124-7

- Successful??: EpiNav, Smart Liver, Medtronic, WTC
- Today:

NiftyView

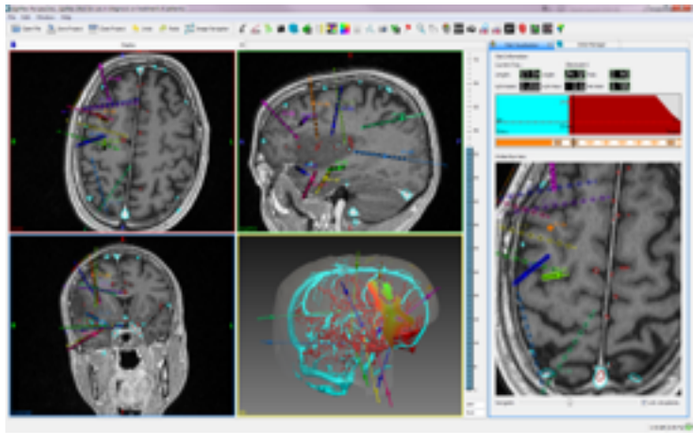


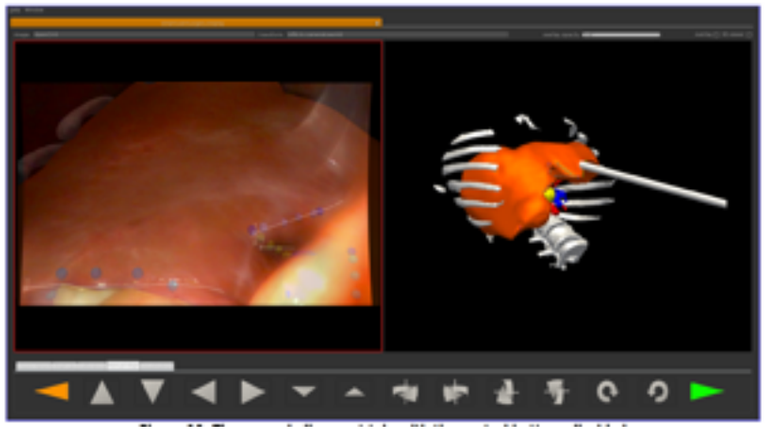
NifTK



NiftyIGI

EpiNav



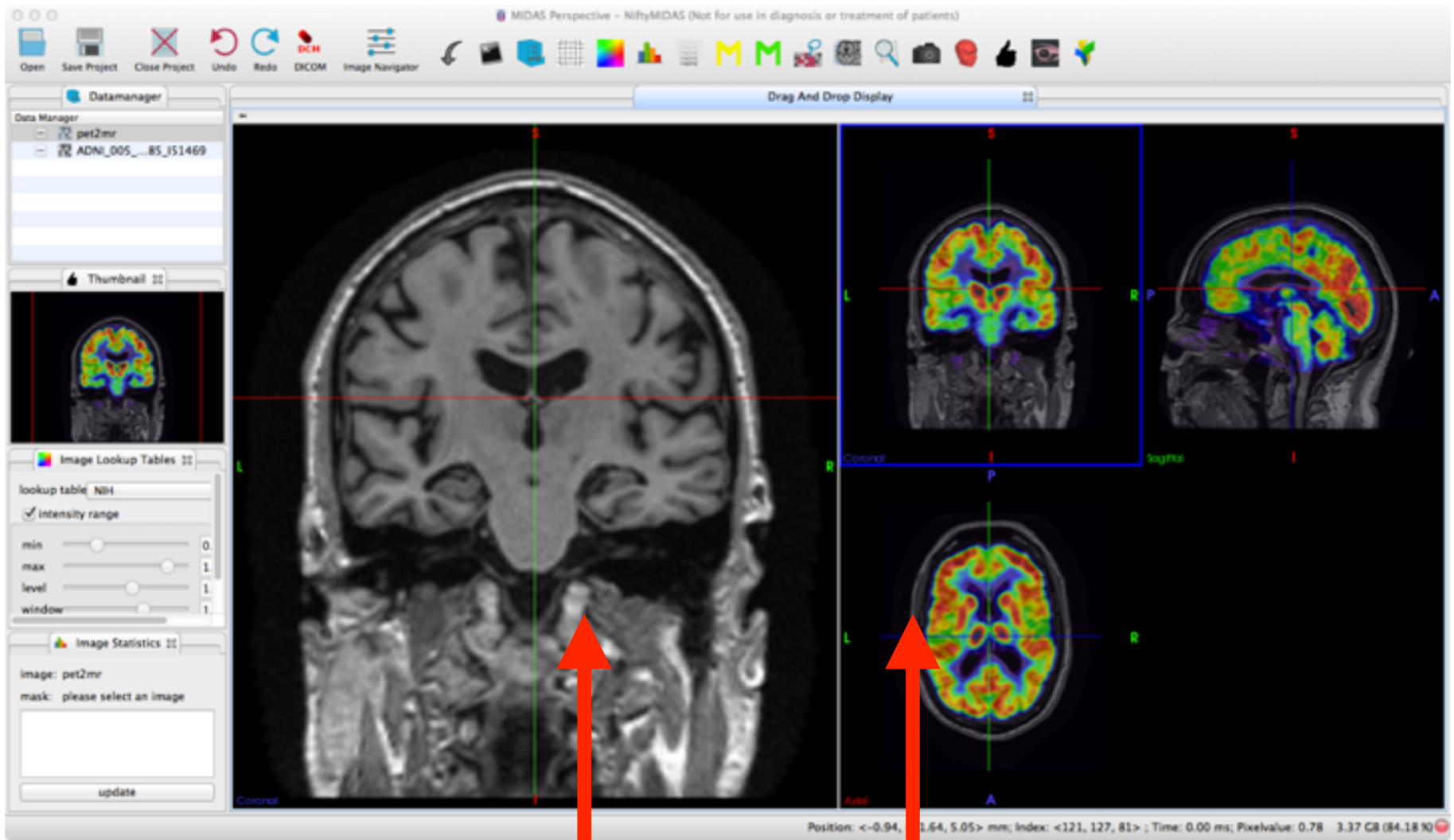


NiftySLS

- Key Architecture Principles - Model-View-Controller

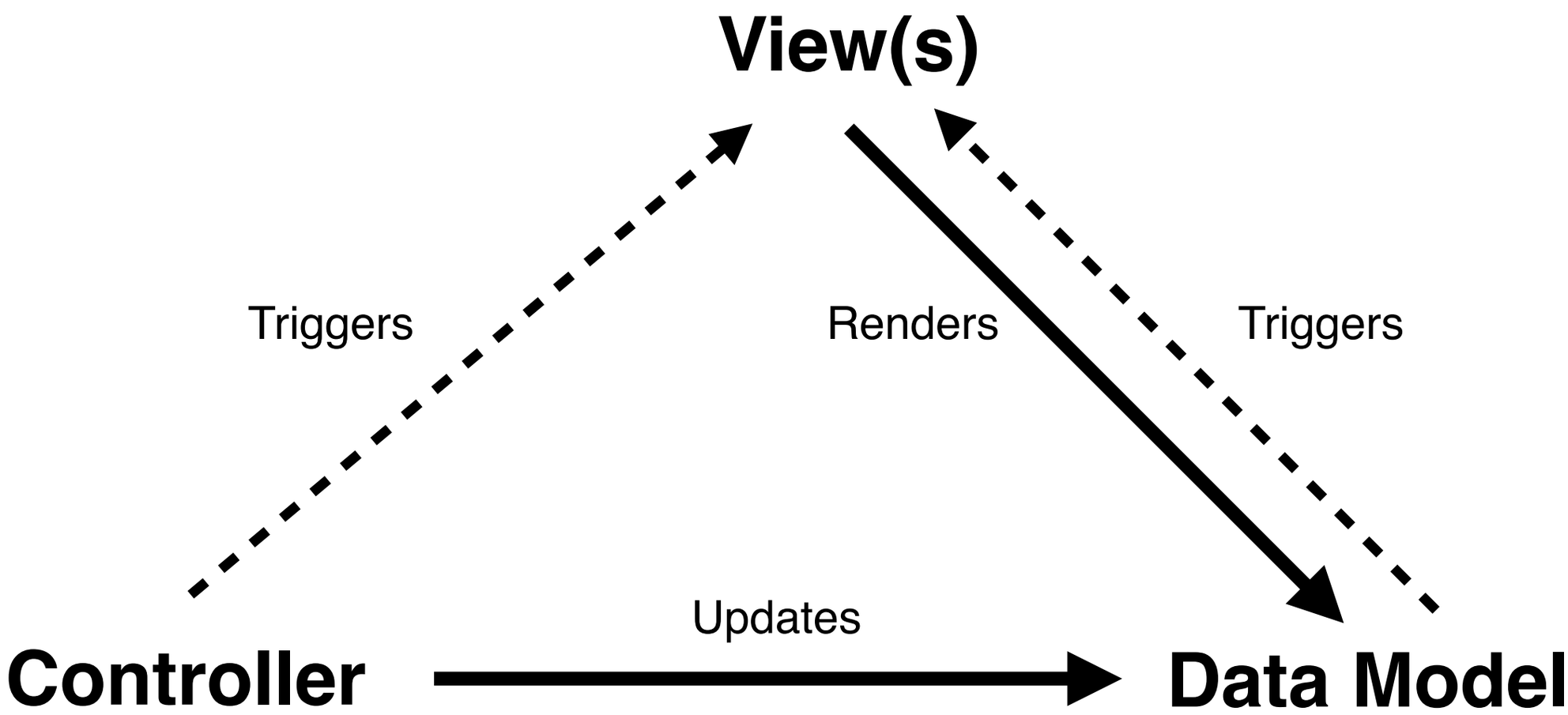
Data Model

Controller



View(s)

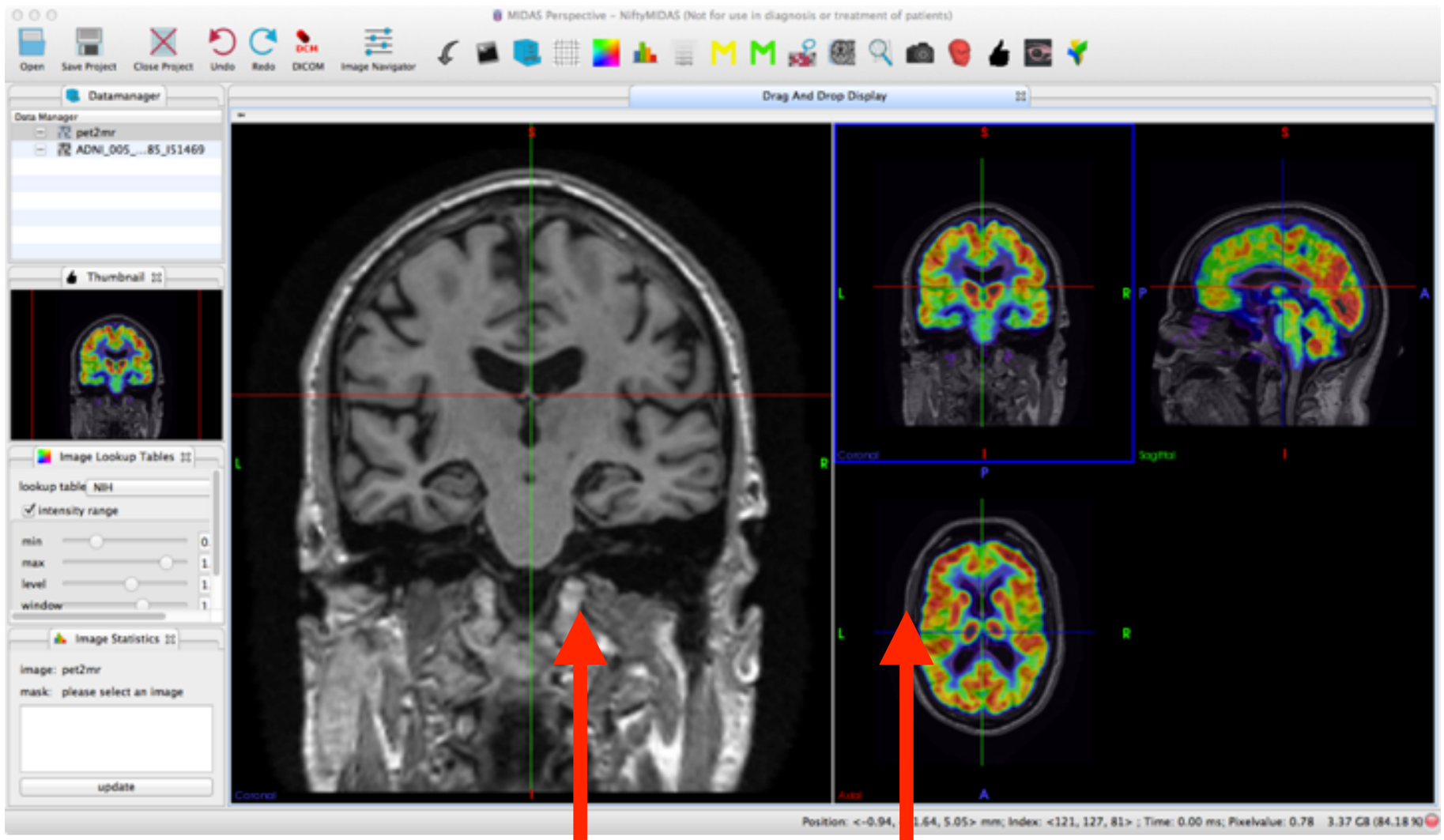
- Key Architecture Principles - Model-View-Controller



- Key Architecture Principles - Plugins - Just add your bit!
- EpiNav, Smart Liver = added specific plugins

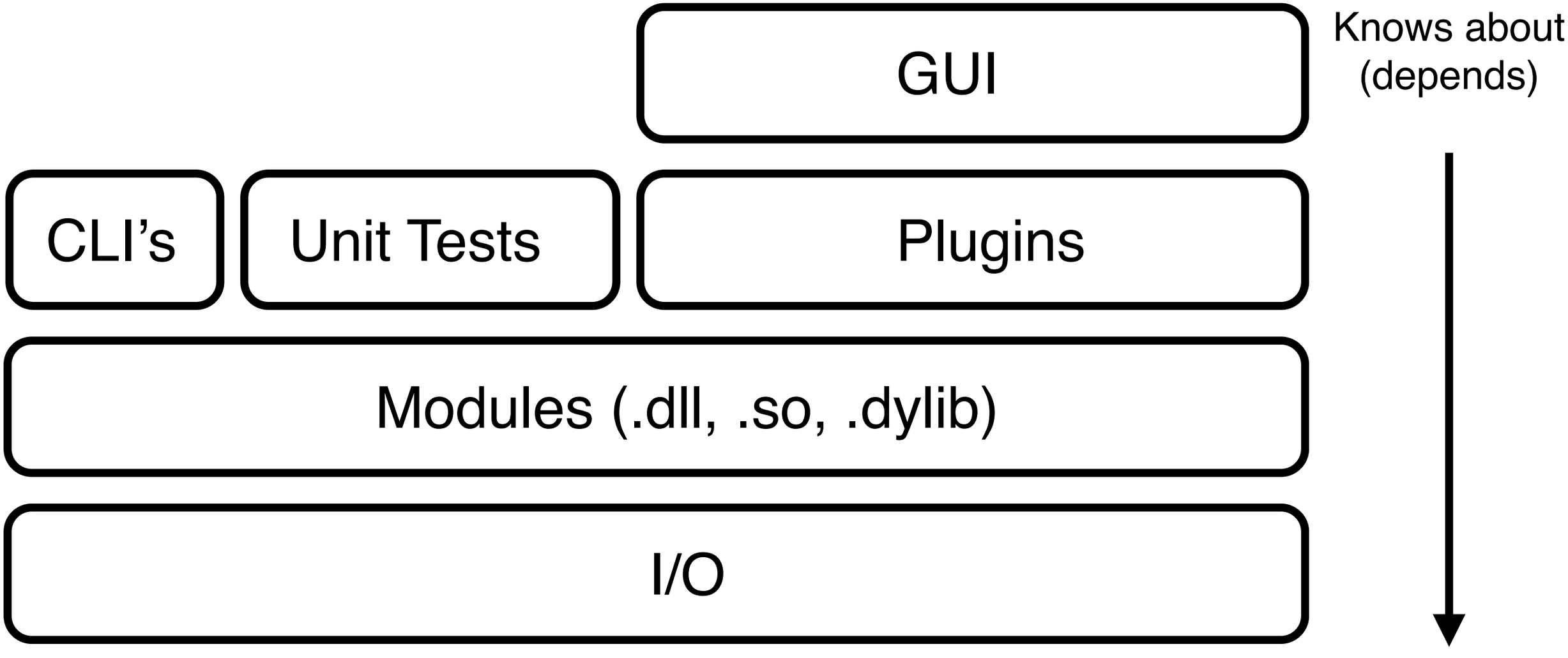
Data Model

Controller

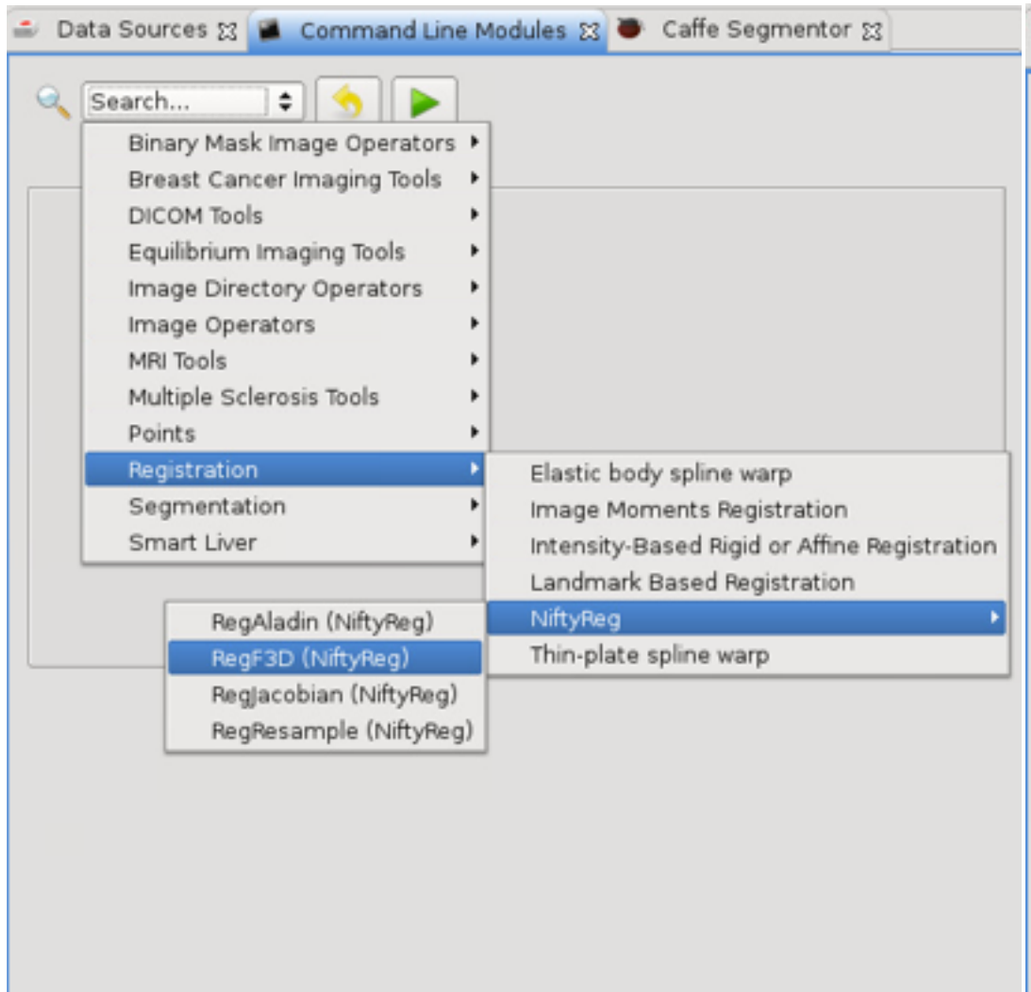


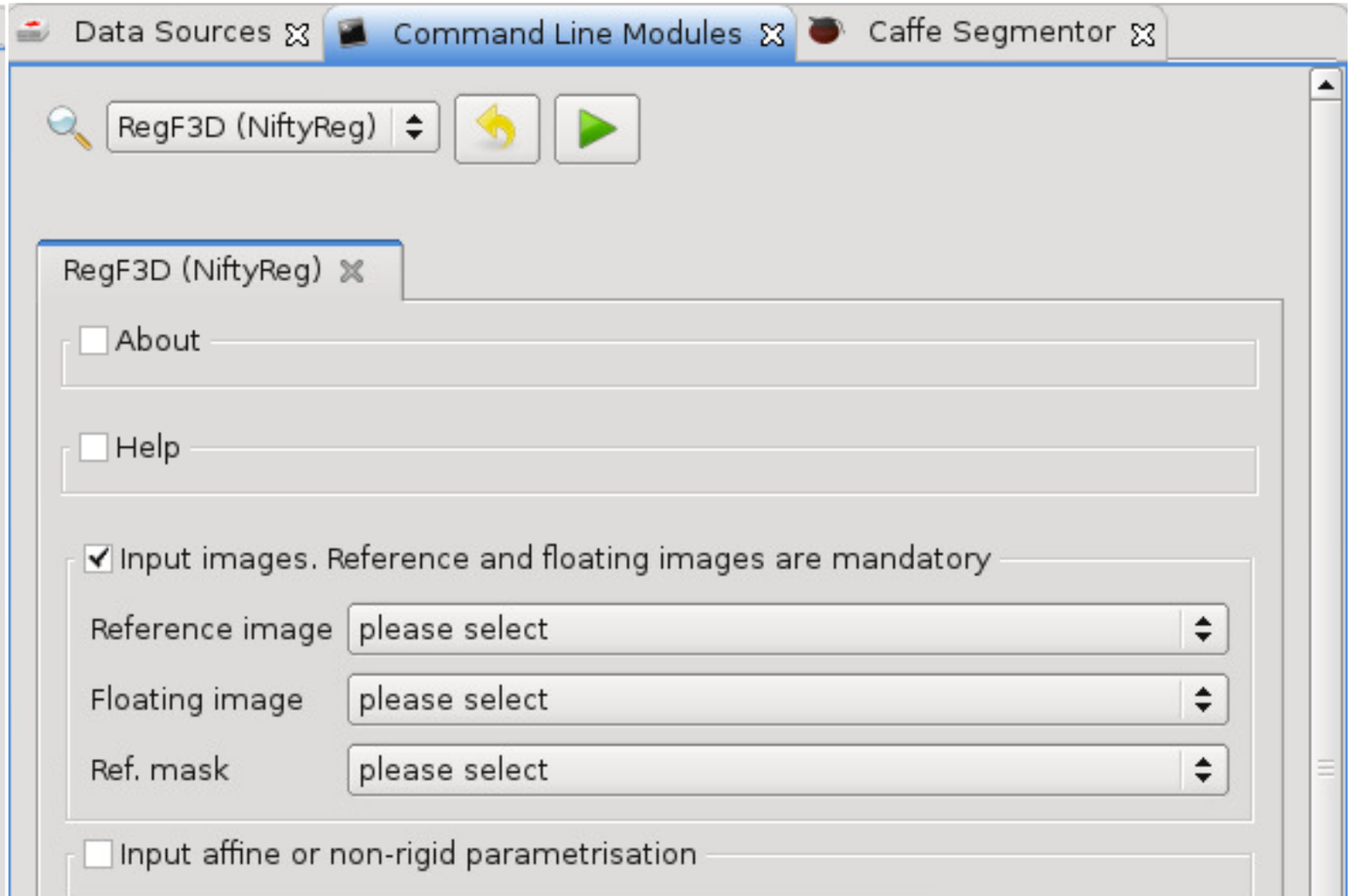
View(s)

- Key Architecture Principles - Layered

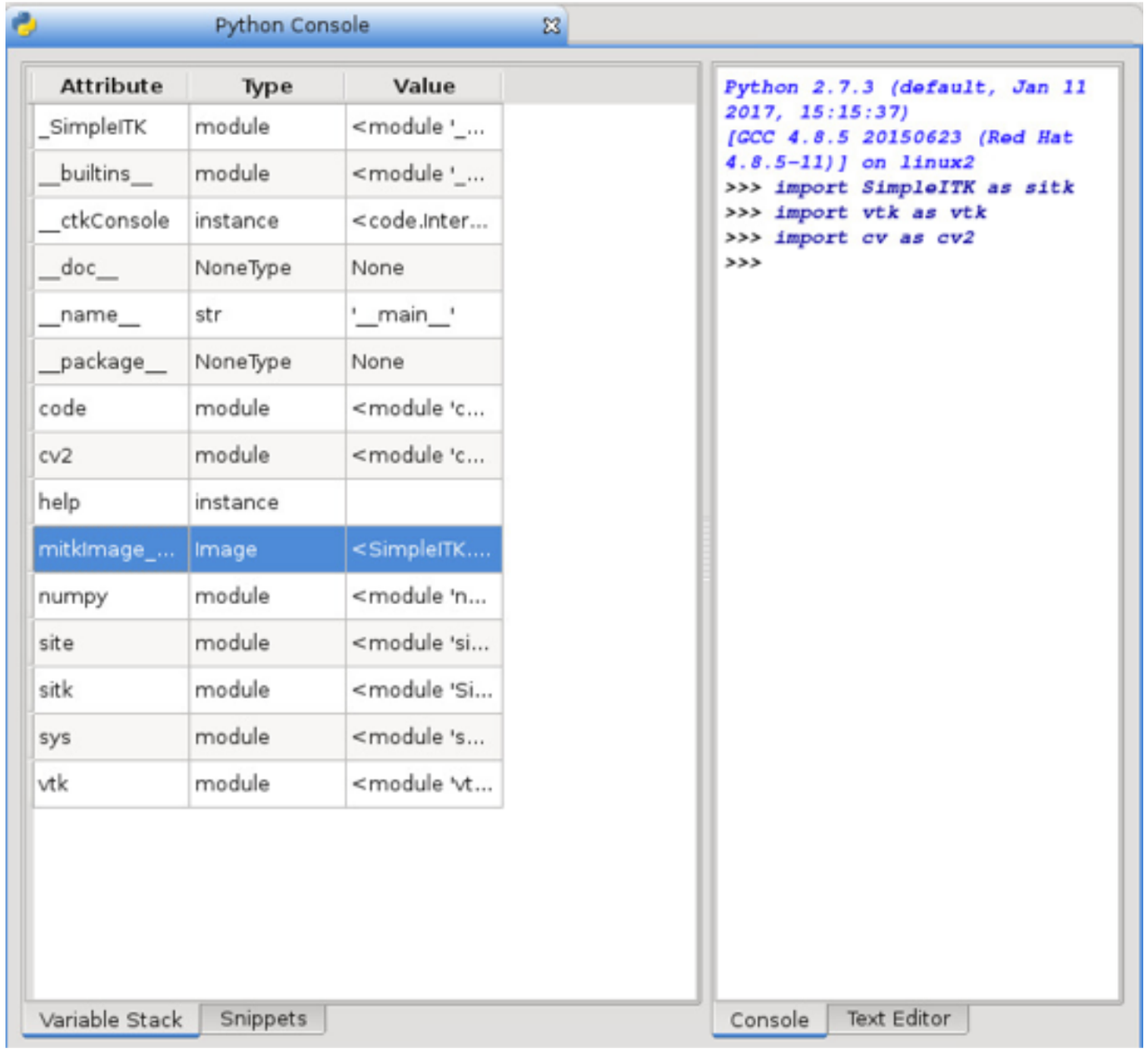


- Integrated
  - NiftyReg, NiftySeg, NiftySim = bundled
  - 150+ command line apps = available, in GUI
    - (C++, python, bash, MATLAB!!!)
    - Useful for getting Windows users to run stuff (e.g. clinicians)

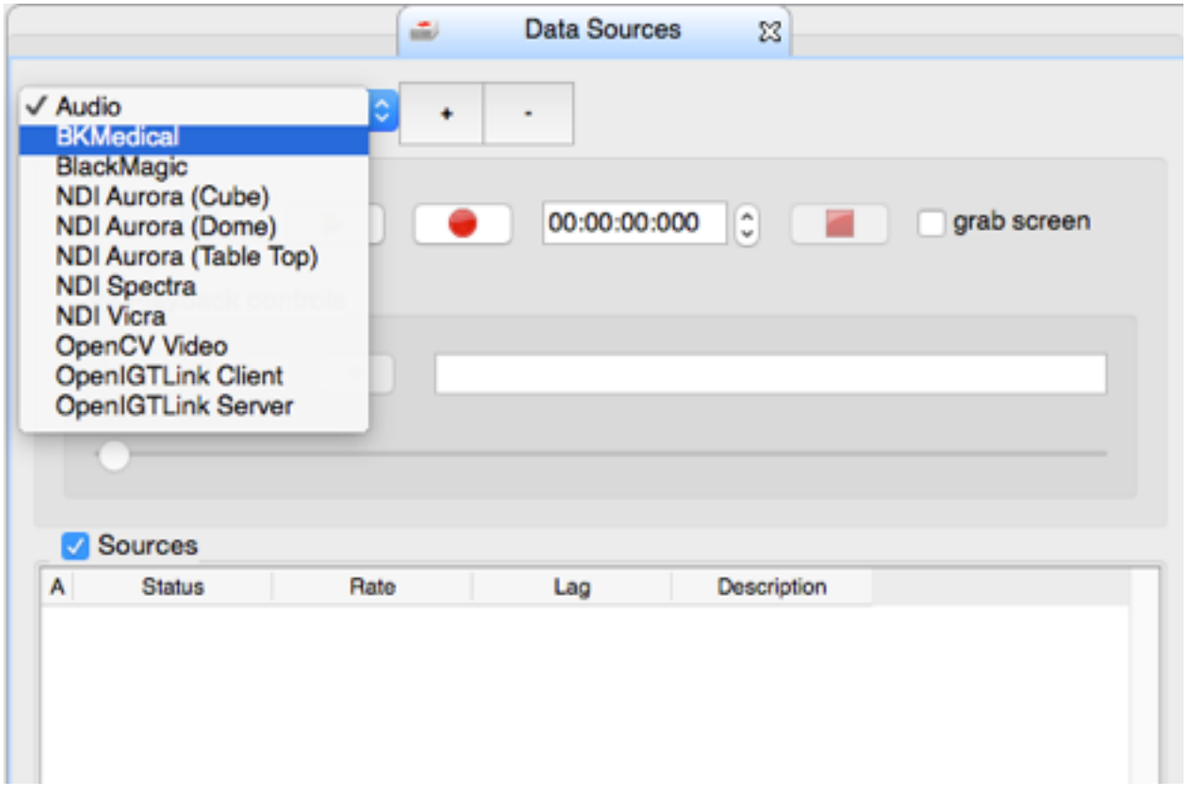


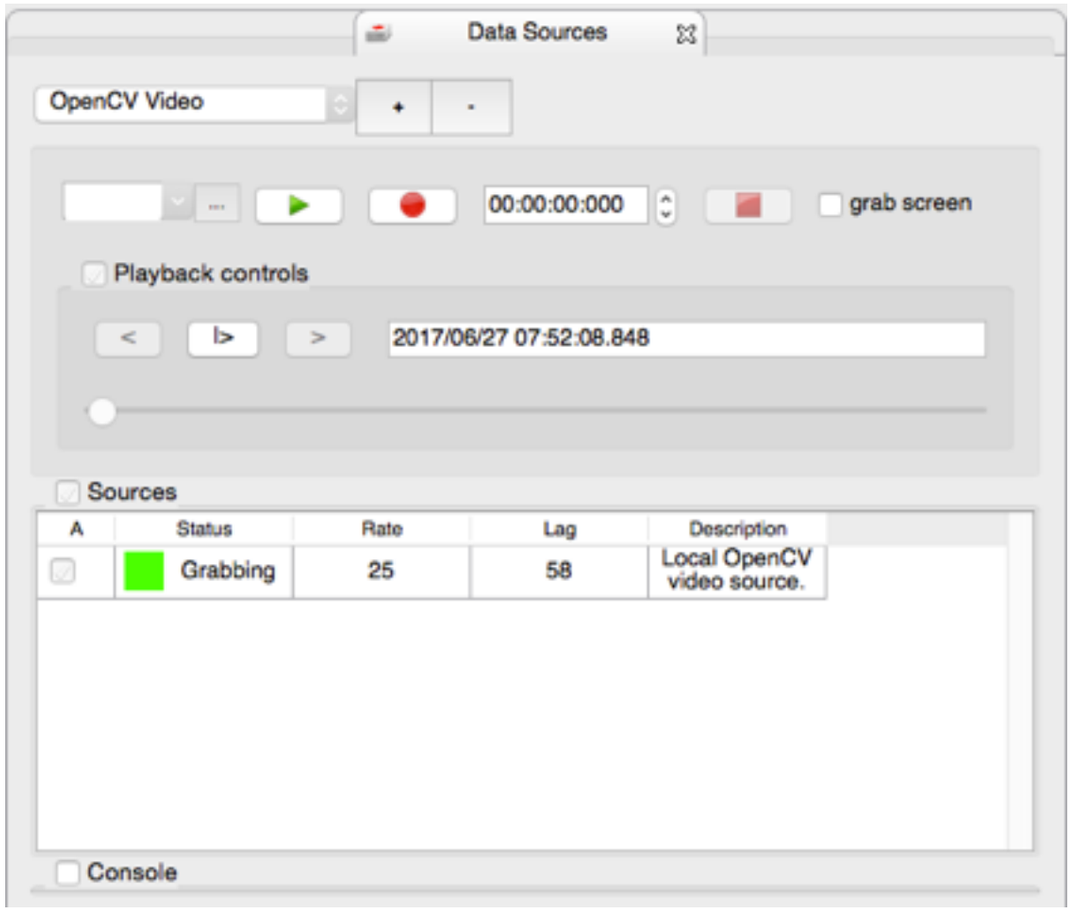


- Python
  - Drop image into plugin
  - Converted to numpy array
  - SimpleITK, OpenCV, VTK, numpy available
  - Any takers?



- Most important IGI plugin - Data Sources plugin
  - Video - OpenCV (Webcam, Epiphan), NVIDIA SDI, BlackMagic (in progress)
  - Trackers - NDI Vicra, Spectra, Atracsys
  - Ultrasound - Ultrasonix, BK Medical via ethernet





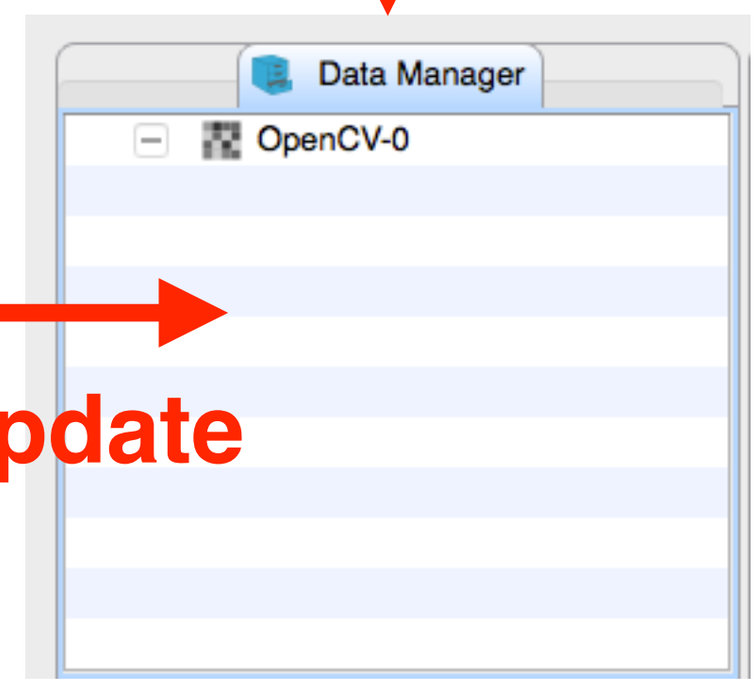
- Independent buffers, update Data Model using timer

Data Sources Plugin

$t_n \ldots t_5, t_4, t_3, t_2, t_1$

$t_n \ldots t_5, t_4, t_3, t_2, t_1$

Data Model



Update

- Timing Calibration
  - Each buffer fills up at different speeds
  - Timestamp of data is always ‘wrong’, independent of FPS

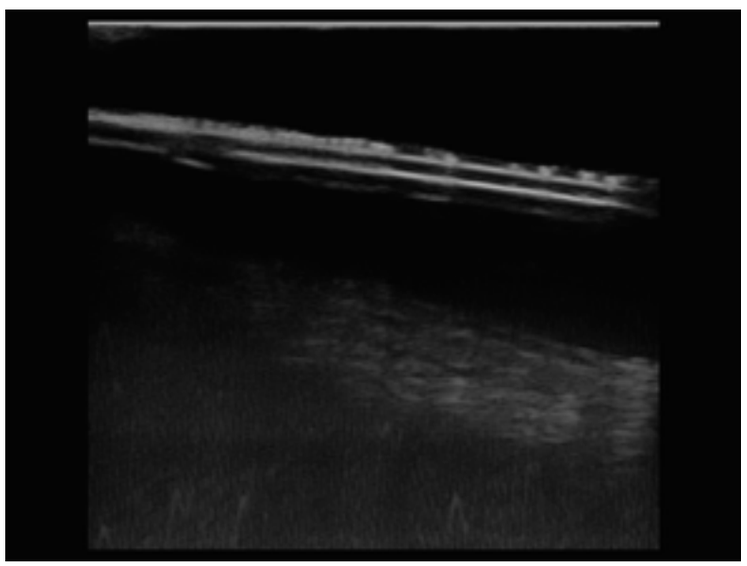
$t_{210}, t_{185}, t_{160}, t_{135}, t_{110}, t_{85}, t_{60}, t_{35}, t_{10}$

t=180? (e.g. tracker < 10ms, ultrasound 50 - 100ms)

$t_{260}, t_{210}, t_{160}, t_{110}, t_{60}, t_{10}$

• Relative Timing Calibration e.g. Video/US to Tracker

(1) Periodic Motion:
10-12 slow cycles
500-1000 samples



(2) NifTK
niftkCalibrationPointClicker

N = Next
P = Previous
Q = Quit

List of 2D points + timestamps

(3) NifTK
niftkTimingCalibration

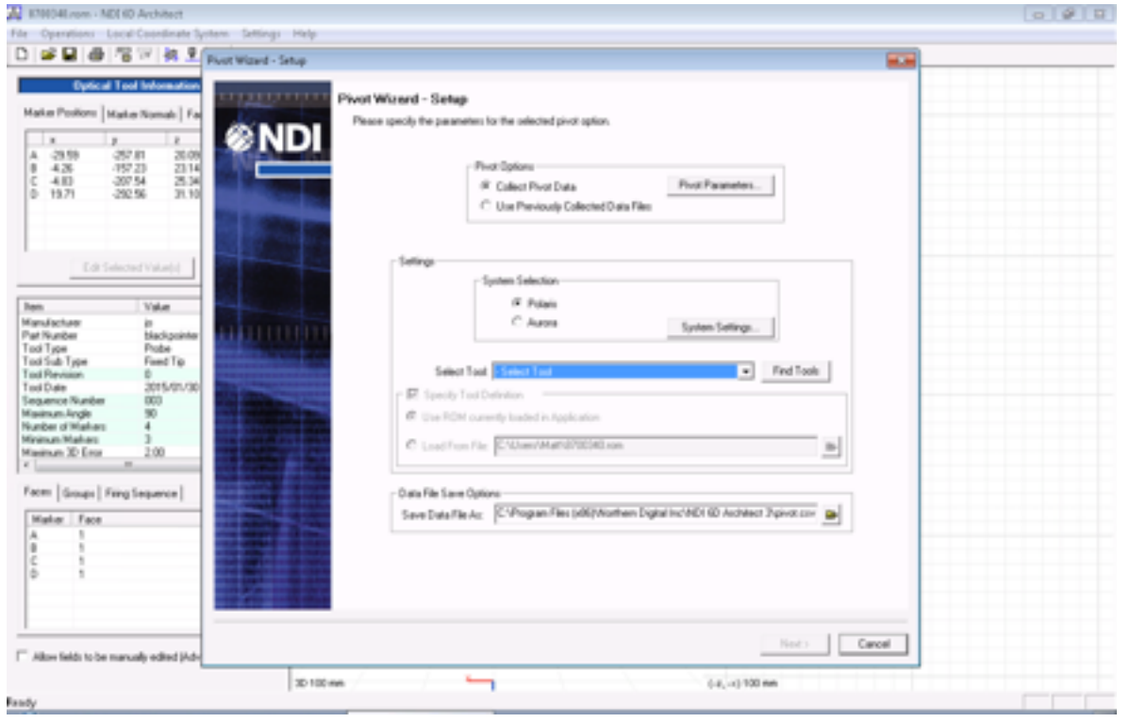
Input:
- 2D points + timestamps
- 3D matrices + timestamps

Algorithm:
- PCA - find major axis
- Scale 0 - 1
- Gradient descent on 0ffset
- Cost function = NCC

Output
- lag of 2D points relative to tracker in integer milliseconds

https://cmiclab.cs.ucl.ac.uk/CMIC/NiftyCal/blob/master/Code/niftkTimingCalibration.h

- Pivot Calibration
  - NDI Provides tools
  - Collect tracking matrices while pivoting
  - Know where tip is relative to origin of tracking marker



- Use NifTK/NiftyIGI Data Sources Plugin
- Connect to Tracker
- Record a data set of matrices

- niftkPivotCalibration
  - ––matrixDirectory
  - ––outputMatrixFile

into world coordinates. For each transformation ${}^{W}T_{P(i)}$, the pointer's tip remains static in pointer coordinates (X_P) as well as world coordinates (X_W), as described by following formula:

$$X_W = \left[{}^{W}R_{P(i)}\right] X_P + {}^{W}t_{P(i)}. \quad (1)$$

As depicted by Tuceryan *et al.* [30], following equation needs to be solved (e.g., by QR decomposition) to determine the pointer tip coordinates:

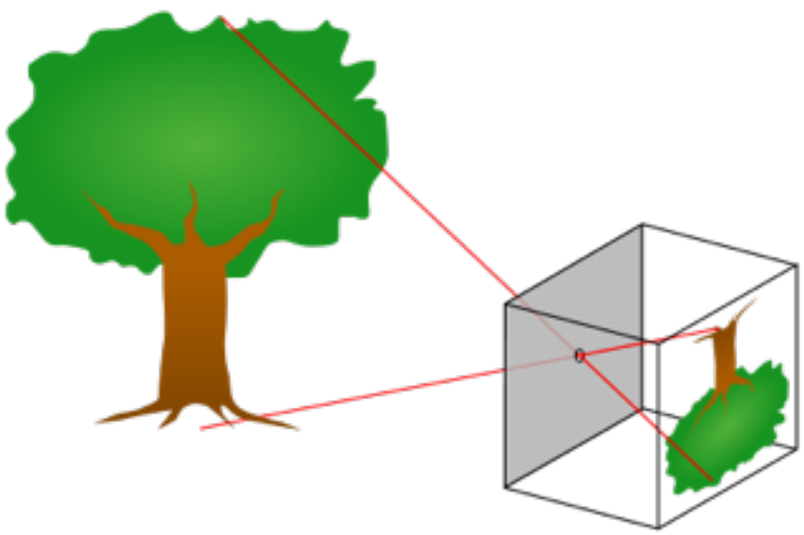
$$\begin{pmatrix} I & -\left[{}^{W}R_{P(1)}\right] \\ I & -\left[{}^{W}R_{P(2)}\right] \\ \vdots & \vdots \\ I & -\left[{}^{W}R_{P(m)}\right] \end{pmatrix} \begin{pmatrix} X_W \\ X_P \end{pmatrix} = \begin{pmatrix} {}^{W}t_{P(1)} \\ {}^{W}t_{P(2)} \\ \vdots \\ {}^{W}t_{P(m)} \end{pmatrix}. \quad (2)$$

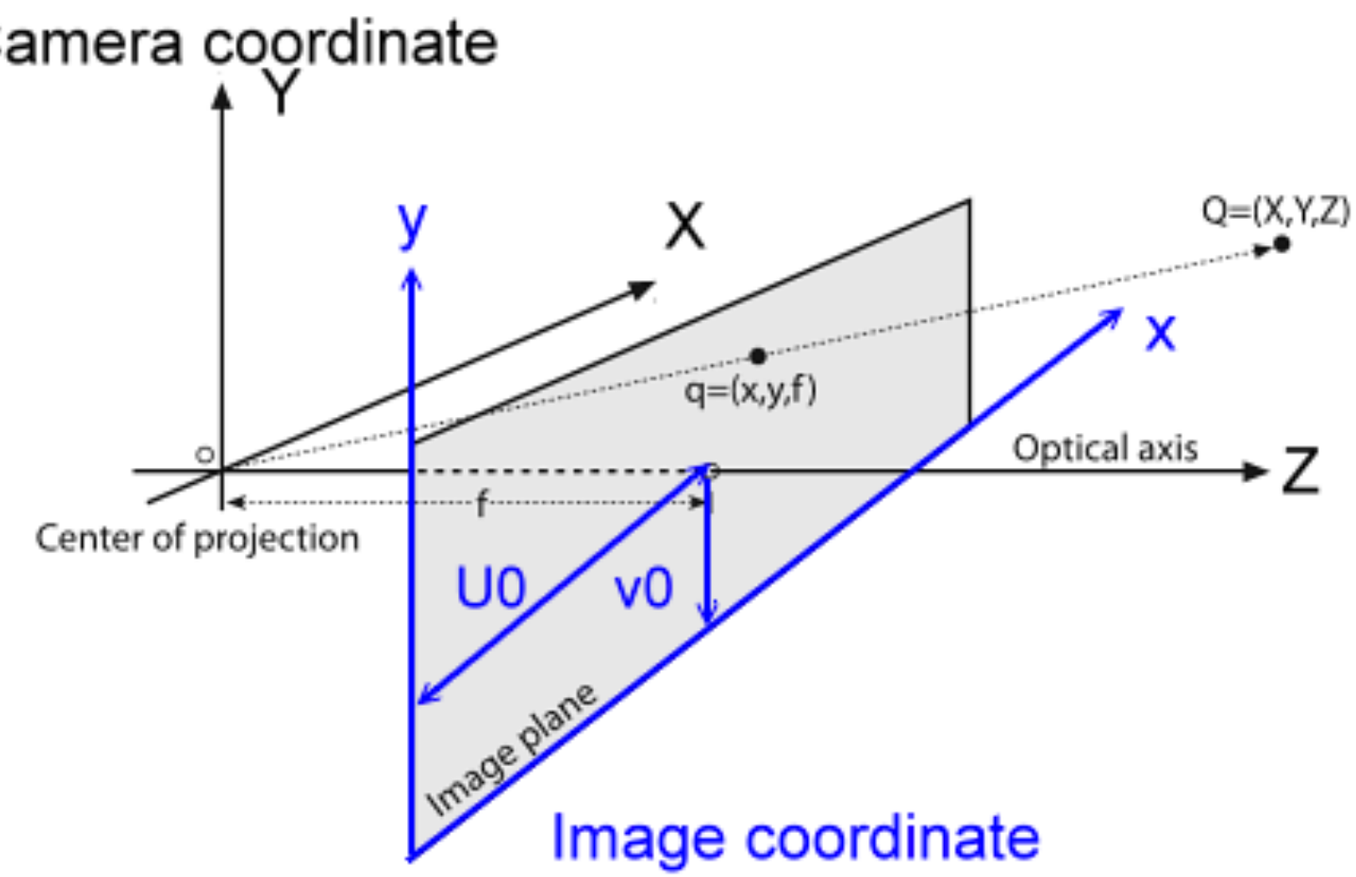
https://github.com/MattClarkson/BARD

http://dx.doi.org/10.1109/TMI.2007.907327

http://dx.doi.org/10.1007/s11548-017-1576-7

- Video Camera Calibration - Pinhole model



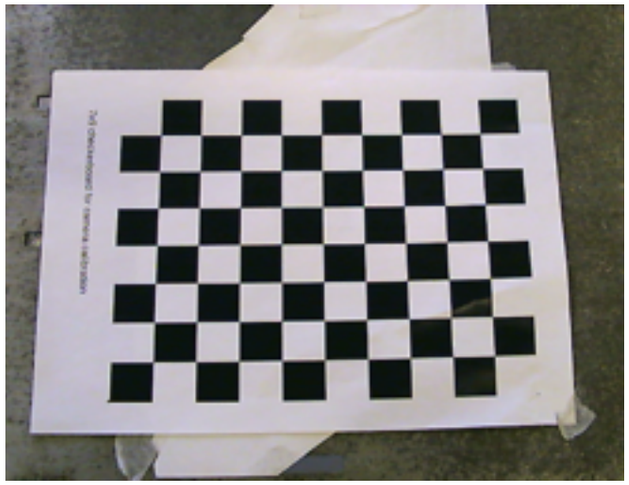


Pinhole camera model

By convention, image plane in front of camera

- The pin-hole model - as in OpenCV/Bouguet

$$s\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}f_x & 0 & c_x\\0 & f_y & c_y\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_1\\r_{21} & r_{22} & r_{23} & t_2\\r_{31} & r_{32} & r_{33} & t_3\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$



$$\begin{bmatrix}x\\y\\z\end{bmatrix}=R\begin{bmatrix}X\\Y\\Z\end{bmatrix}+t$$

$$x'=x/z$$
$$y'=y/z$$
$$u=f_x*x'+c_x$$
$$v=f_y*y'+c_y$$

(without distortion correction)

$$x''=x'\frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6}+2p_1x'y'+p_2(r^2+2x'^2)$$
$$y''=y'\frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6}+p_1(r^2+2y'^2)+2p_2x'y'$$
$$\text{where}\quad r^2=x'^2+y'^2$$
$$u=f_x*x''+c_x$$
$$v=f_y*y''+c_y$$

(with distortion correction)

http://www.opencv.org

- Multiple views of planar object of known dimensions

- Zhang 2000
  - Bouguet Matlab Toolbox
  - OpenCV
- Images
  - 3D points
  - Corresponding 2D points
- Optimise (n*6 + 4 + 4,5,6)
- Cover field of view
- Generally, need whole pattern

Zhang, PAMI 2000, 10.1109/34.888718

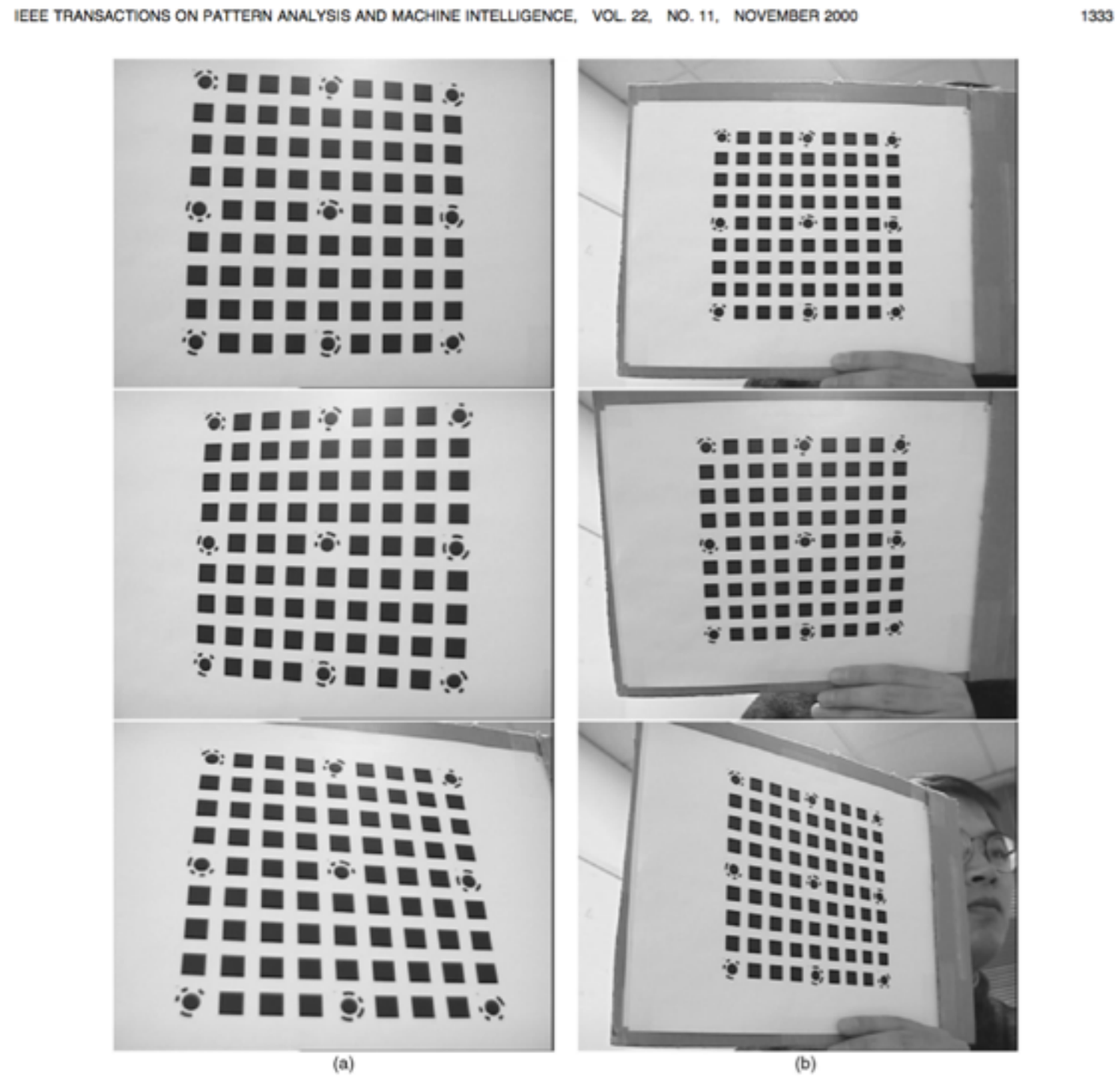


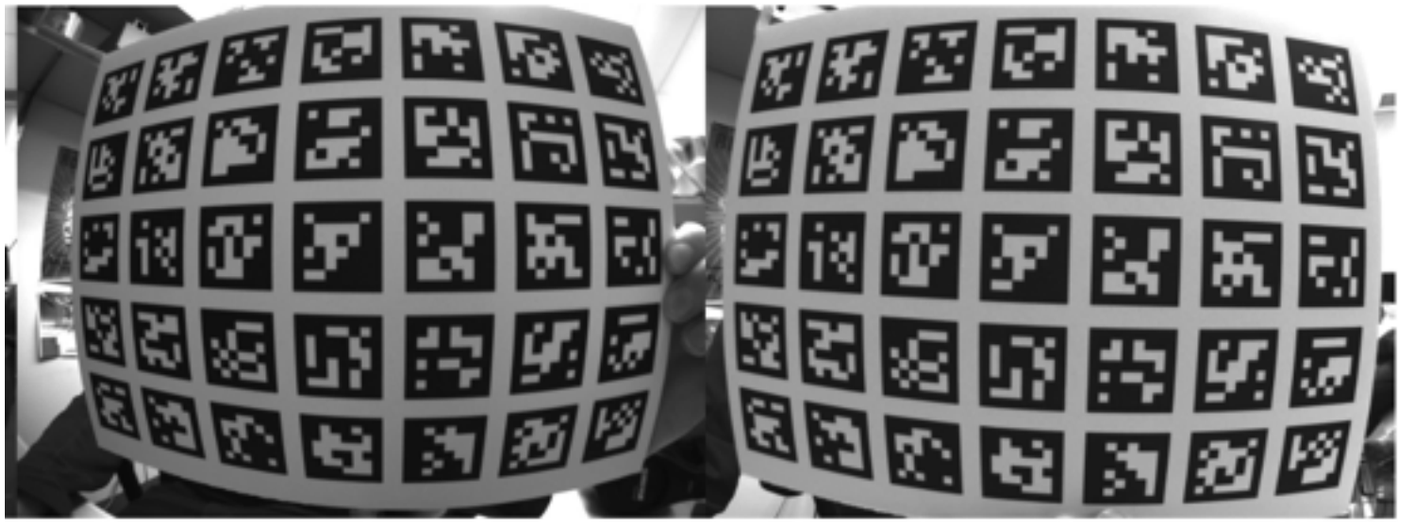
Fig. 1. Two sets of images taken at different distances to the calibration pattern. Each set contains five images. (a) Three images from the set taken at a close distance are shown. (b) Three images from the set taken at a larger distance are shown.

- Multiple views of “Tags” (April Tags, ARToolkit etc).

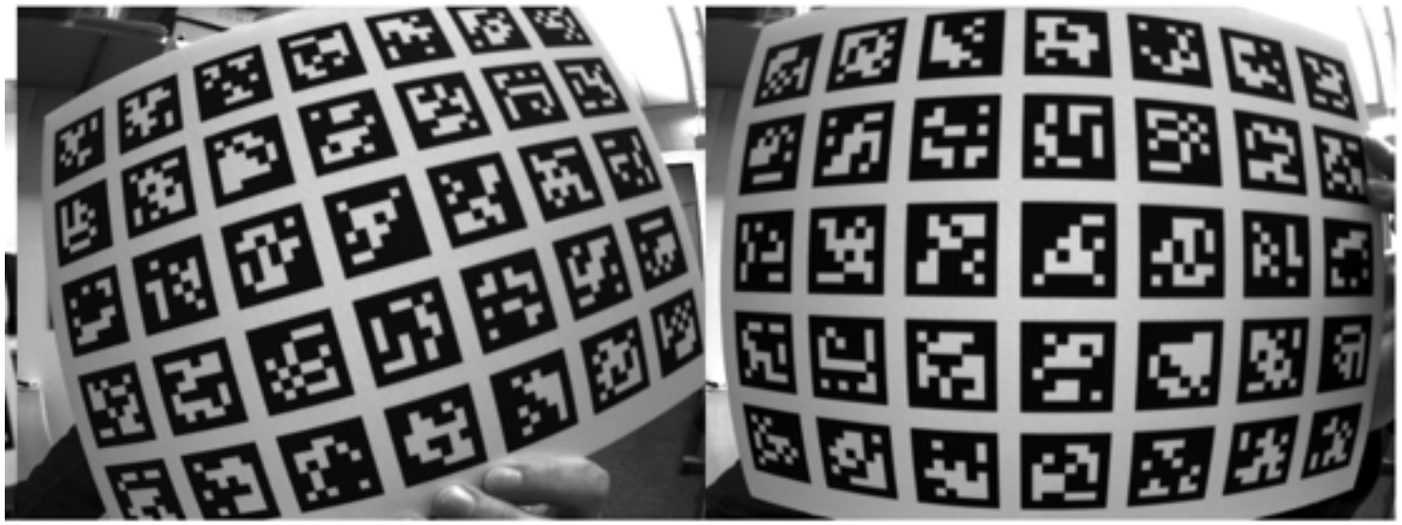
- Richardson 2013
  - AprilTags

- Each tag is “bar code”
- Don’t need all of them

- More difficult image processing
- Laparoscope images poorly (non-uniform) lit
- Only get central tags



(b) Tamron 13FM22IR (648 × 482) (c) Tamron 13FM28IR (648 × 482)



(e) Boowon BW3M30B (648 × 482) (f) Boowon BW3M30B (1296 × 964)

Richardson, IROS 2013 10.1109/IROS.2013.6696595

- Limited by feature detector??

- Datta 2009
  - Iterative, more accurate feature detection

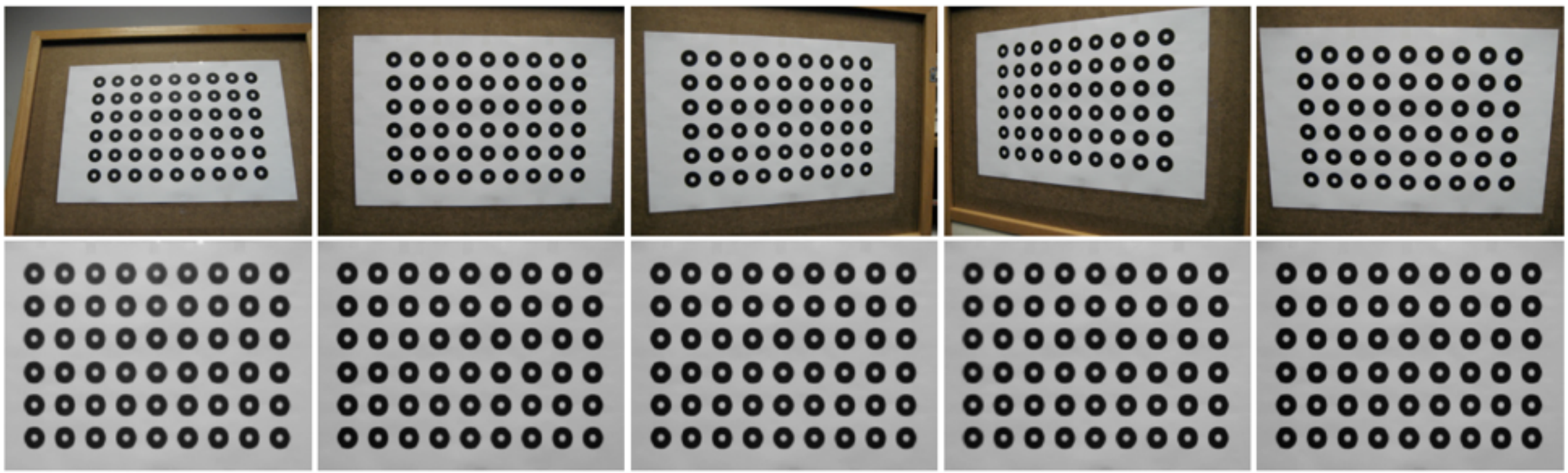


Figure 1: Top row: Input images of the ring calibration pattern. Bottom row: Input images have been undistorted and unprojected to canonical fronto-parallel images. Control points can be precisely localized in the canonical images as compared to the input images.

Datta, ICCV Workshop 10.1109/ICCVW.2009.5457474

- Thompson et al. SPIE 2017
  - Comparison: basically - not much difference
  - Its easy to collect good/bad data using any method

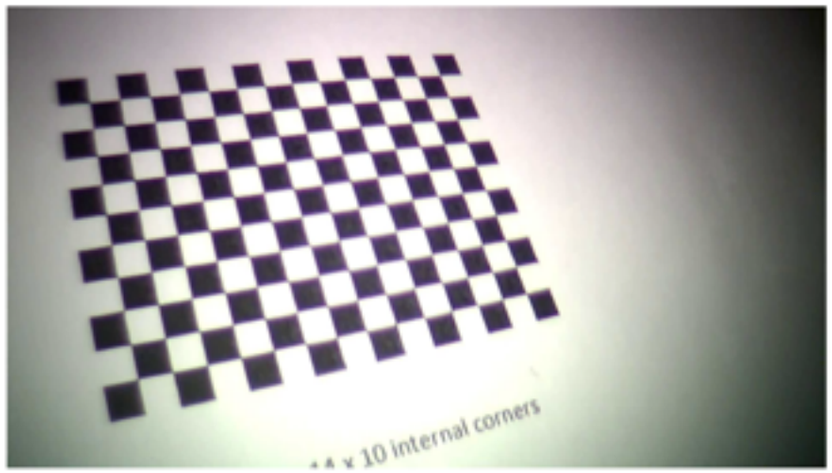


Figure 2. The chessboard calibration pattern.

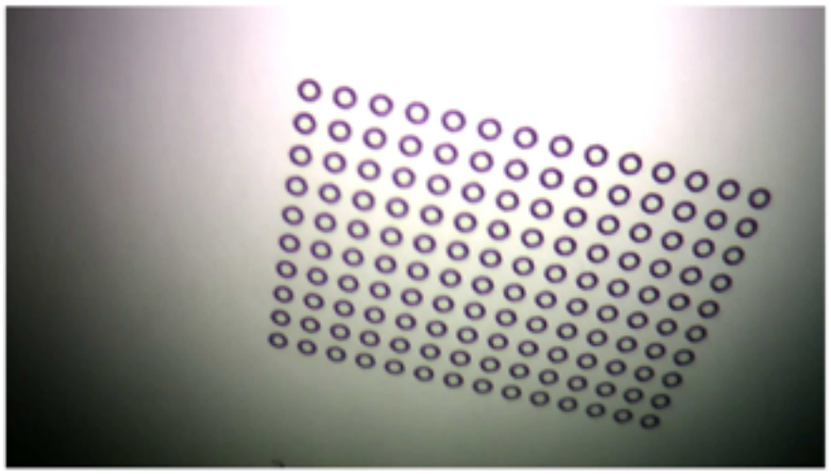


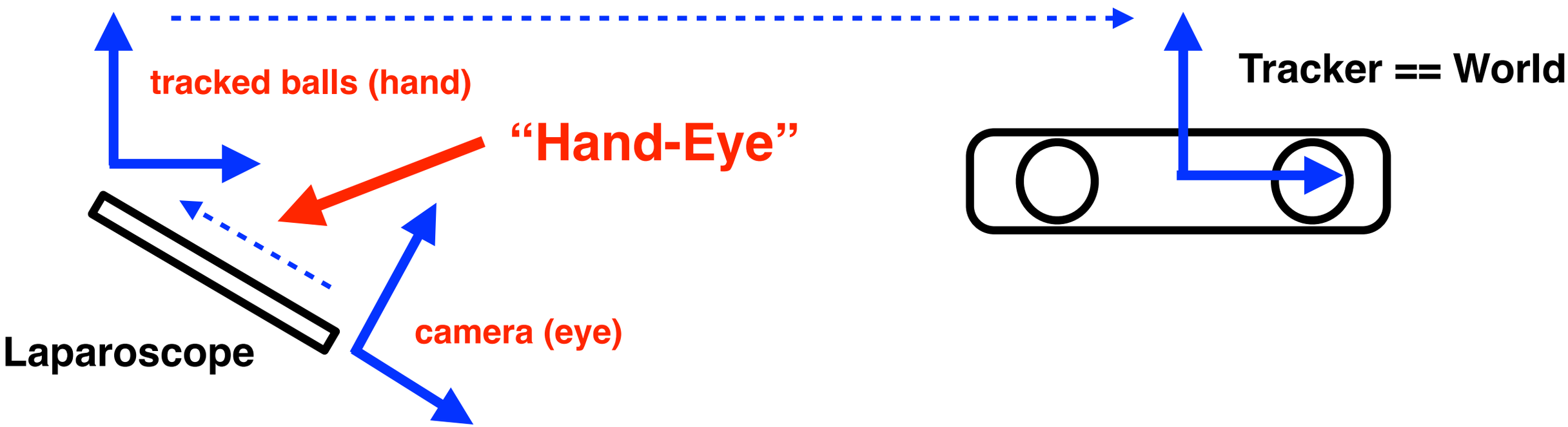
Figure 3. The ring calibration pattern.

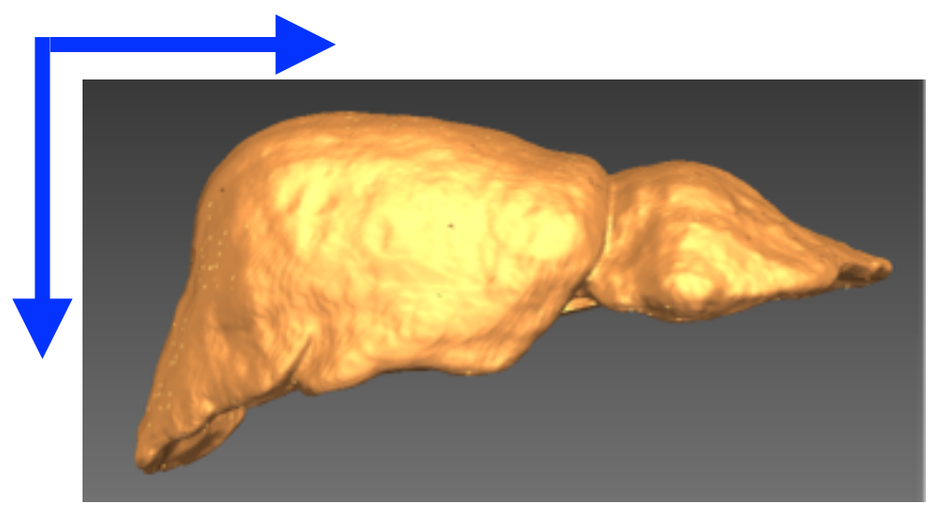
Figure 4. The 24 tag AprilTag calibration pattern.

Figure 5. The 216 tag AprilTag calibration pattern.

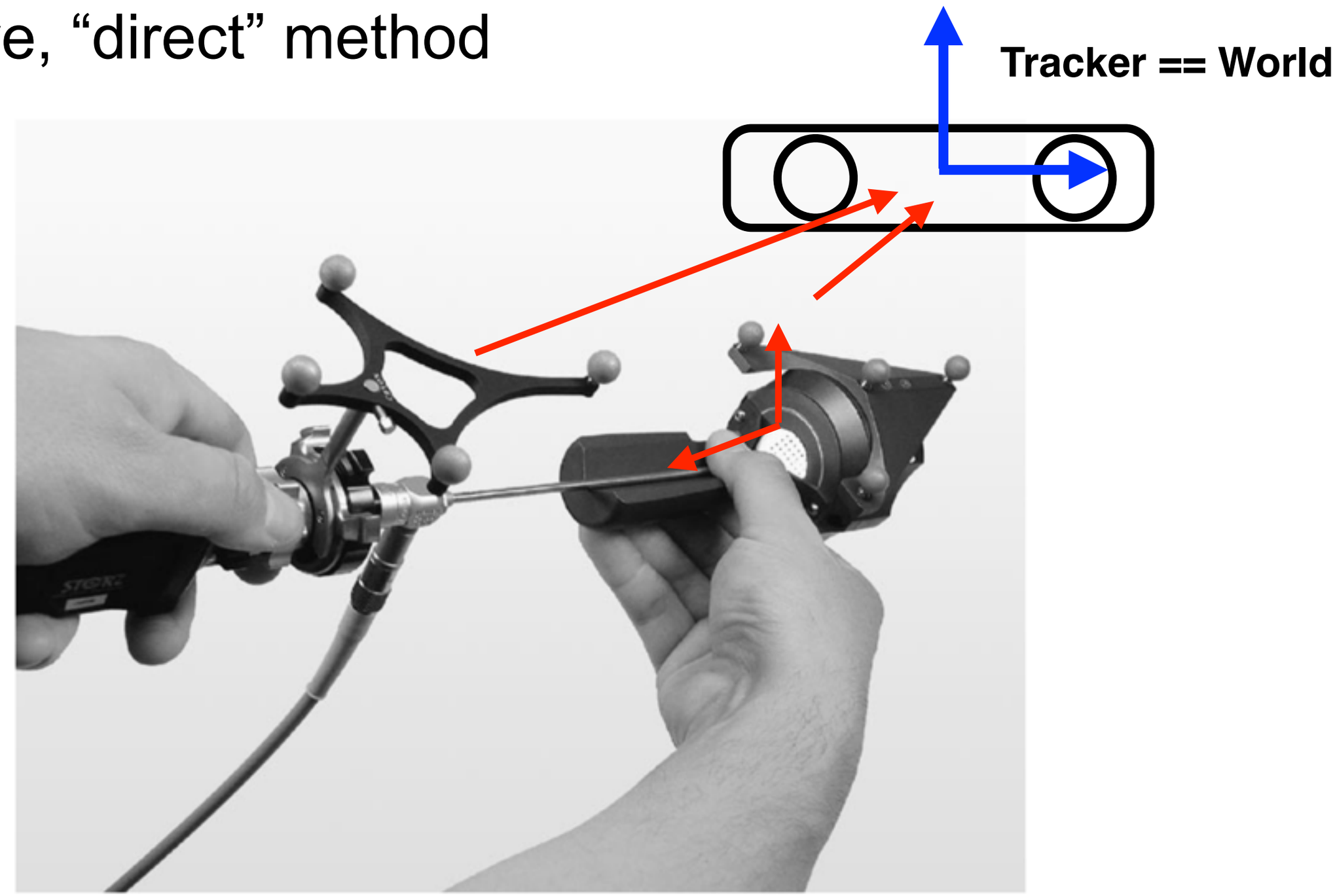
Thompson et al. SPIE 2017: http://dx.doi.org/10.1007/s11548-016-1364-9

- Hand-eye for laparoscopes





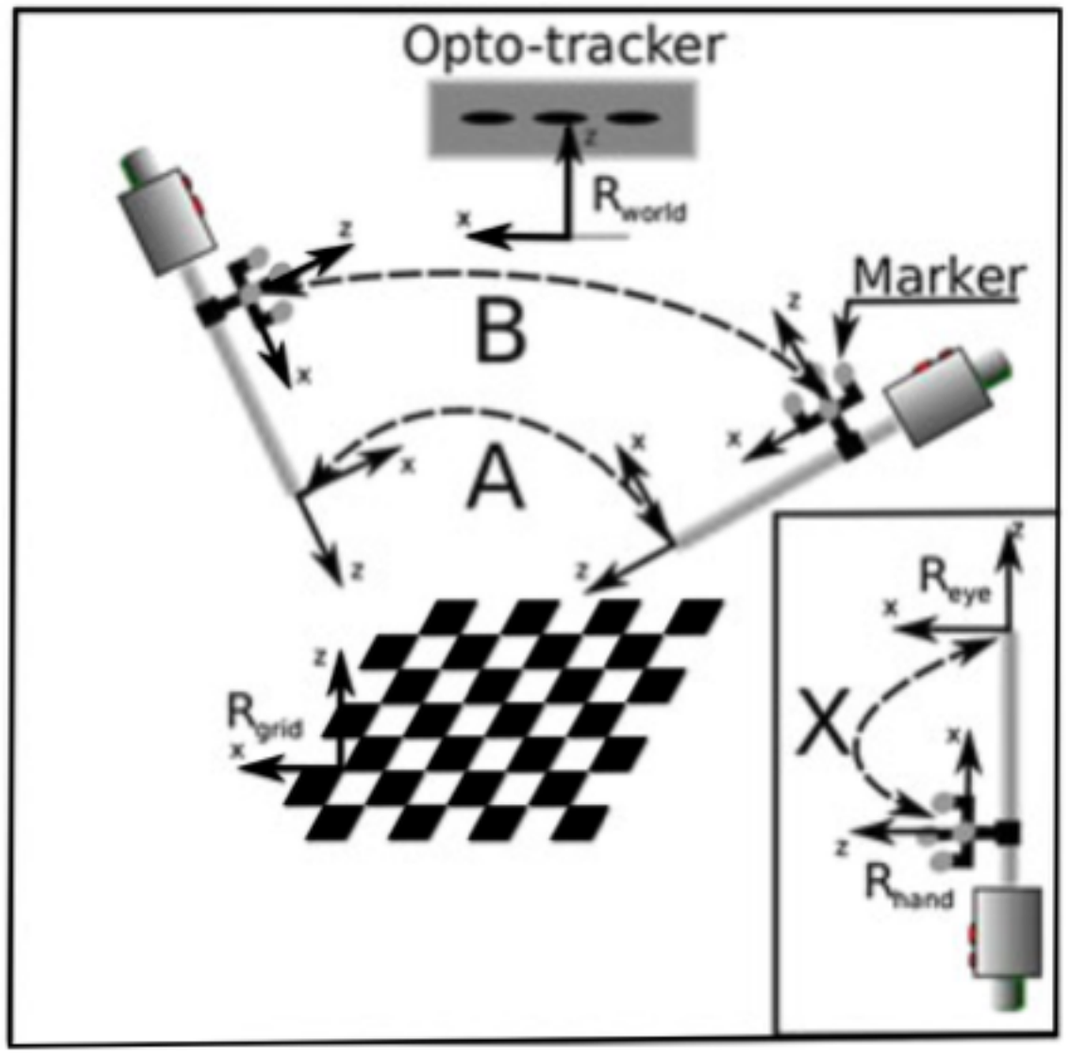
- Hand-eye, “direct” method



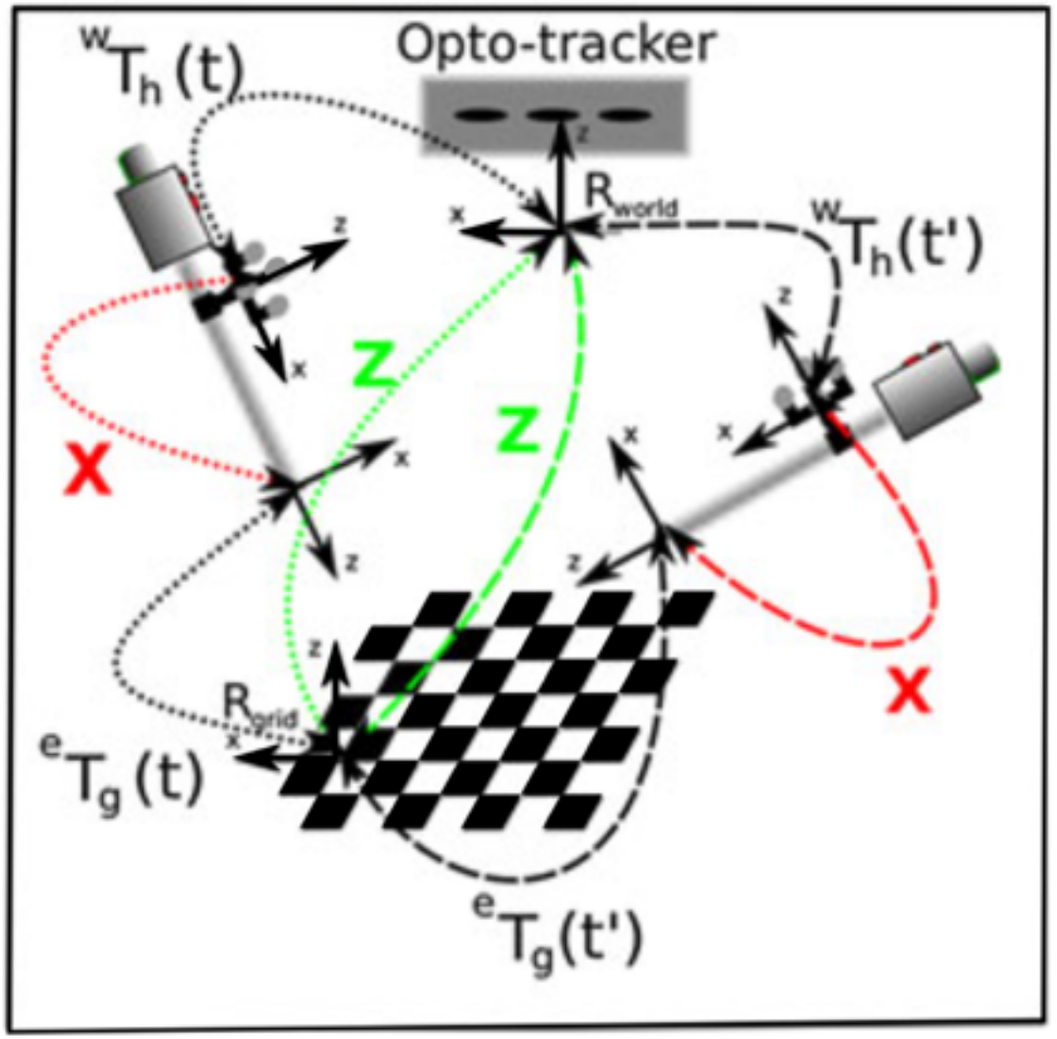
Shahidi TMI 2002 10.1109/TMI.2002.806597

- Multiple views, solve linear+non-linear optimisation

AX = XB

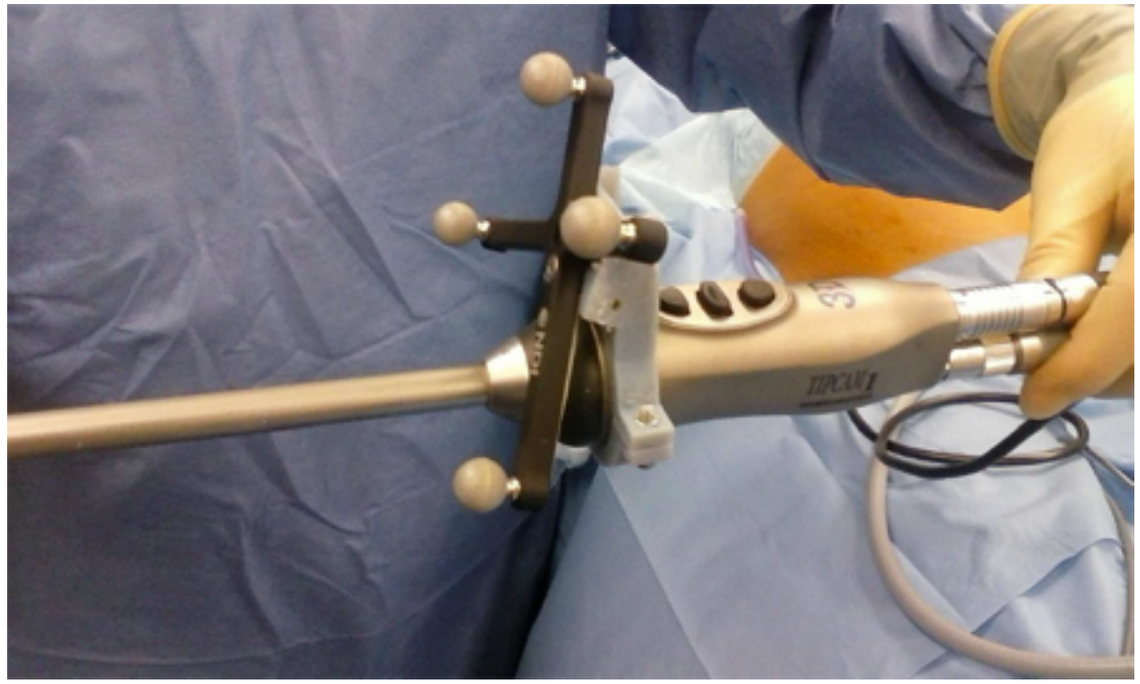


Tsai and Lenz, IEEE Trans Robotics and Automation Vol 5, Issue 3, 1989



Malti Int J. MRCAS 2013, 10.1002/rcs.1478

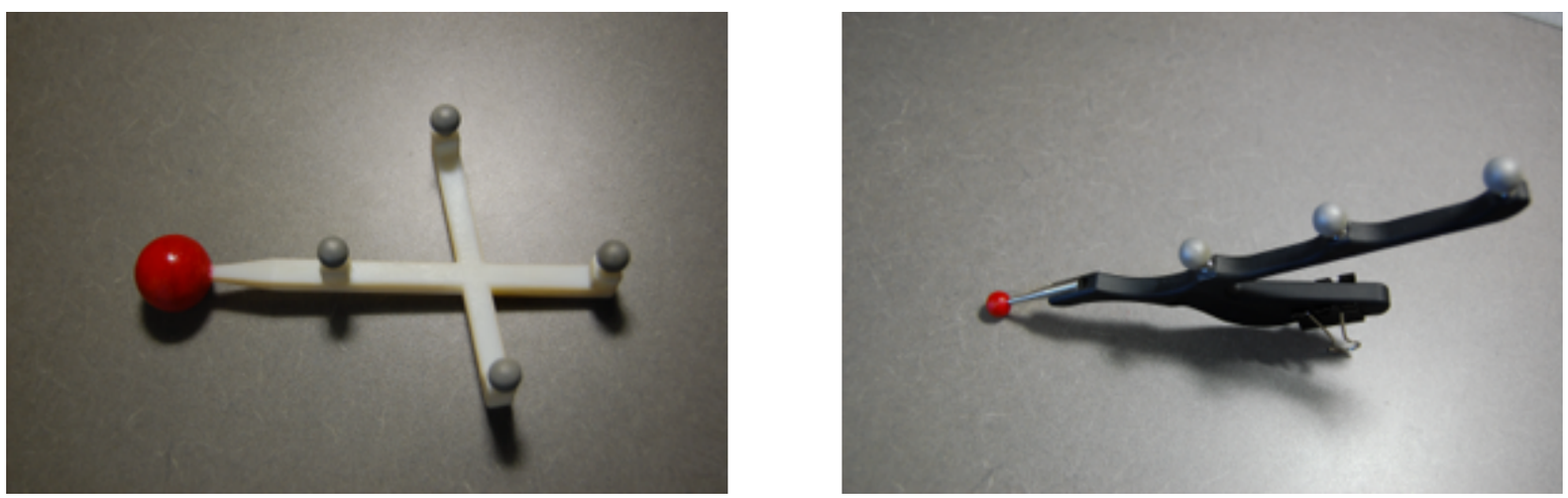
- Surely all solved by now?
- Laparoscopes - can it be done by theatre staff? sterile?



Thompson et al. IJCARS 2016: http://dx.doi.org/10.1007/s11548-016-1364-9

- Surely all solved by now?
- Laparoscopes - can it be done by theatre staff? sterile?

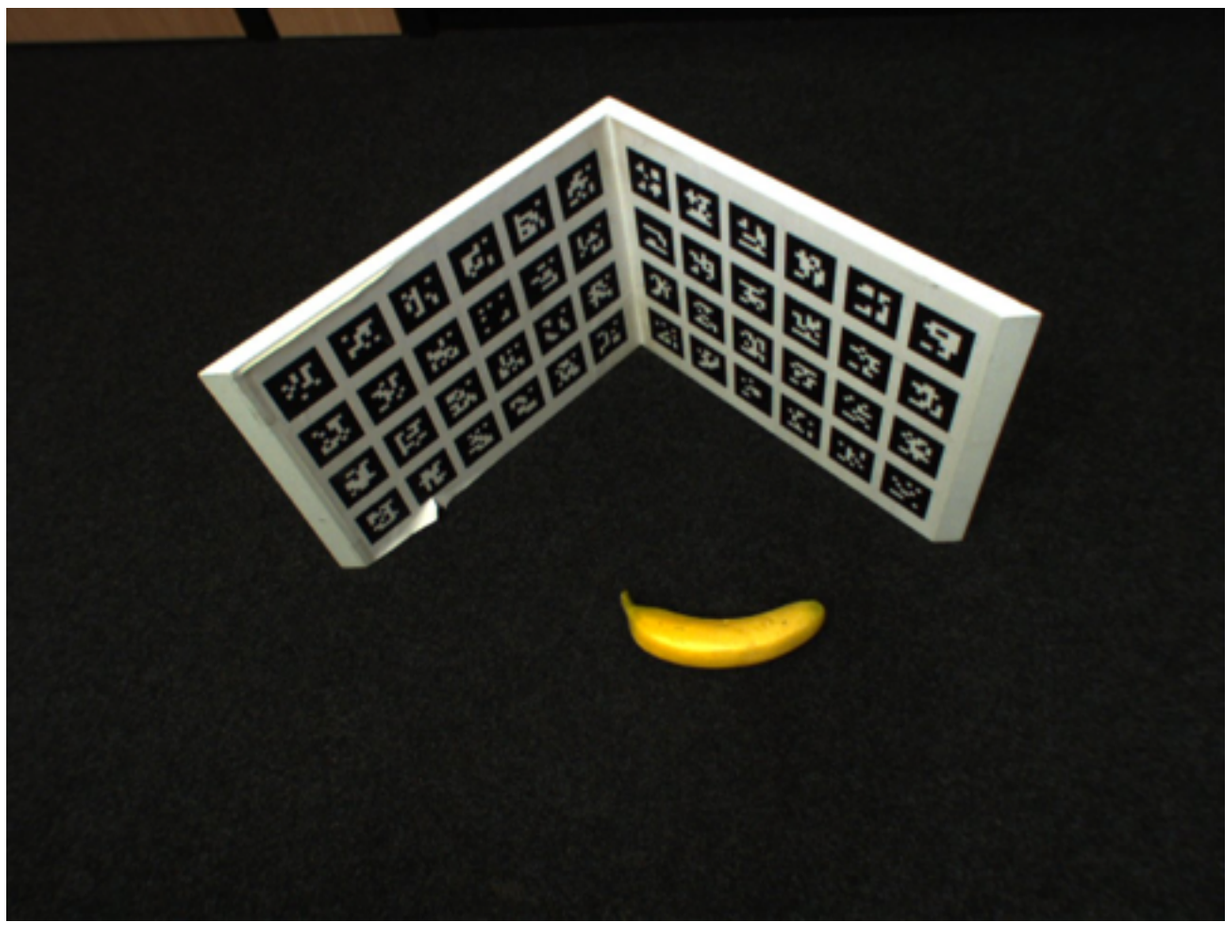
(a) (b) (c)

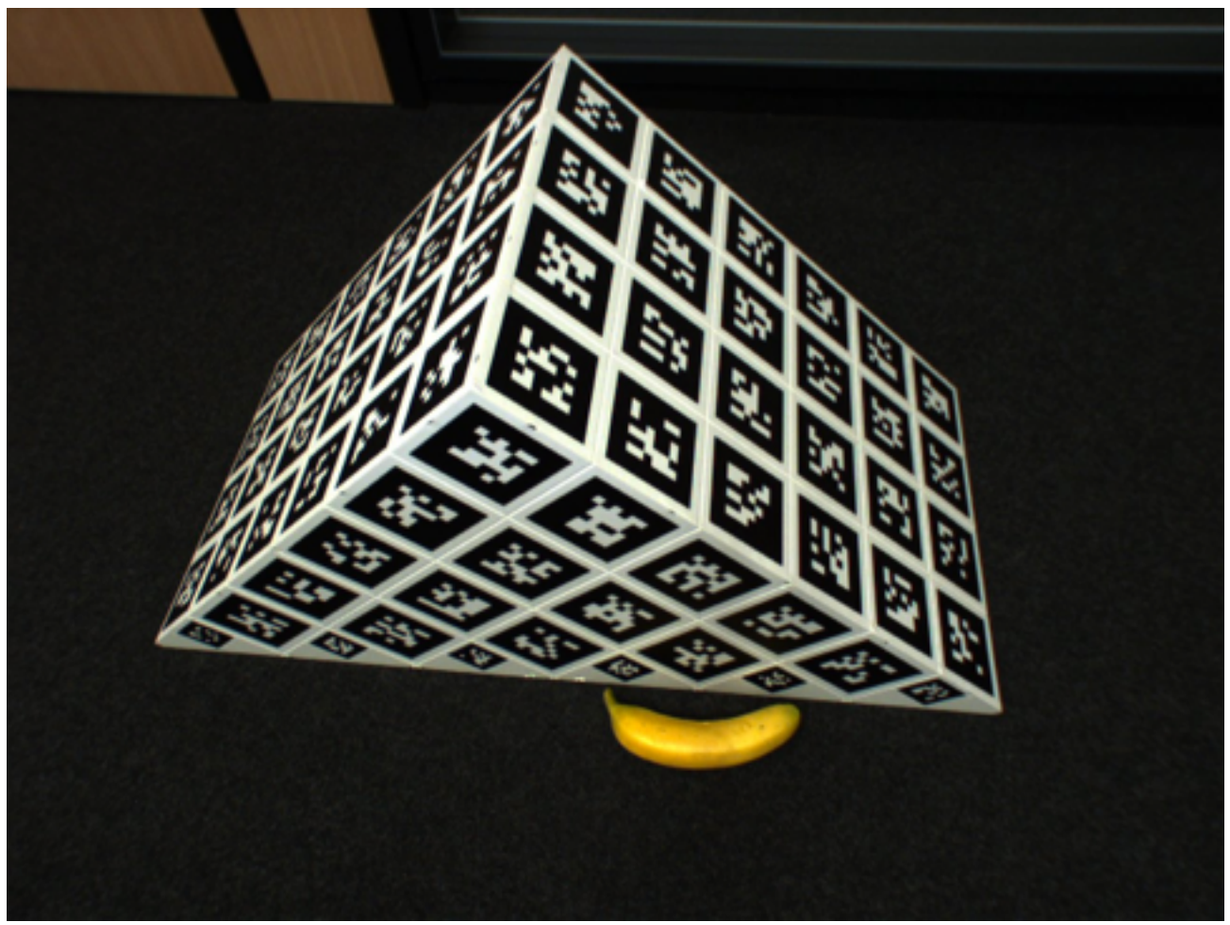


Morgan et al. IJCARS/IPCAI 2017:

Liu et al. IJCARS 2017 http://dx.doi.org/10.1007/s11548-017-1623-4

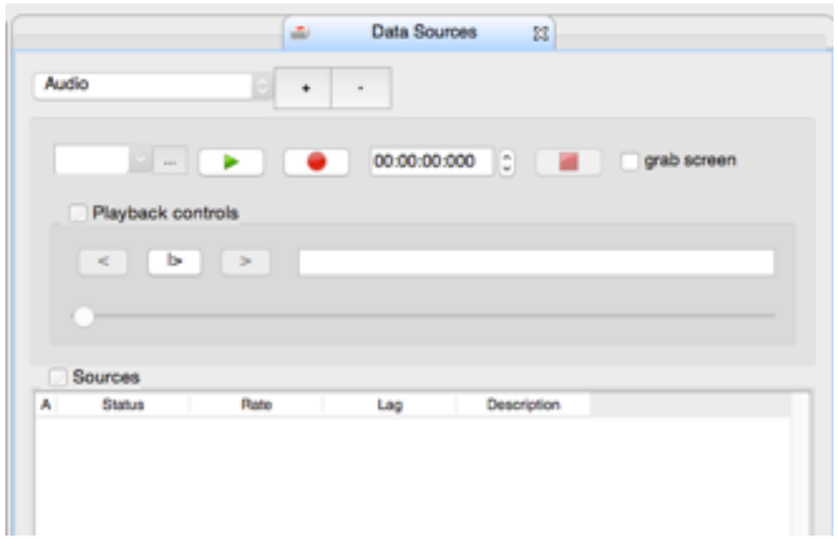
- In operating room? 1-view calibration?
- Tsai's method? 1987?
- Perceive3D.com



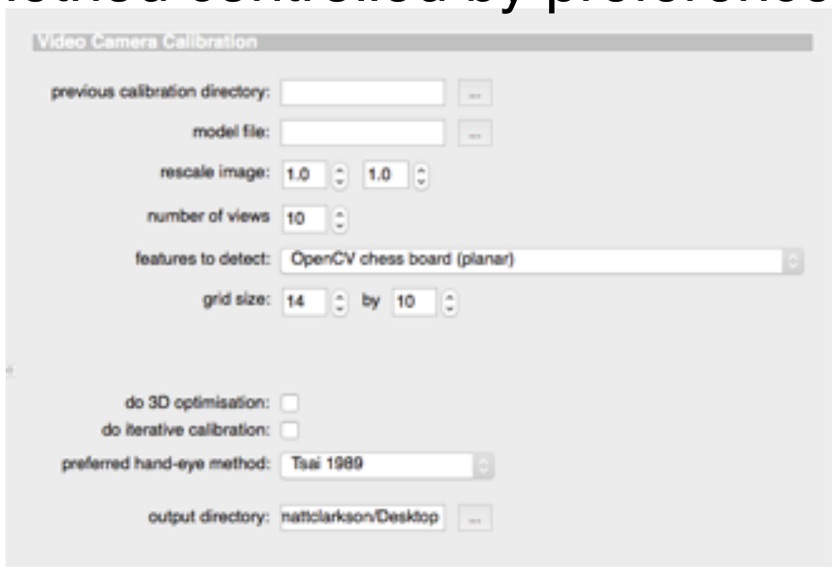


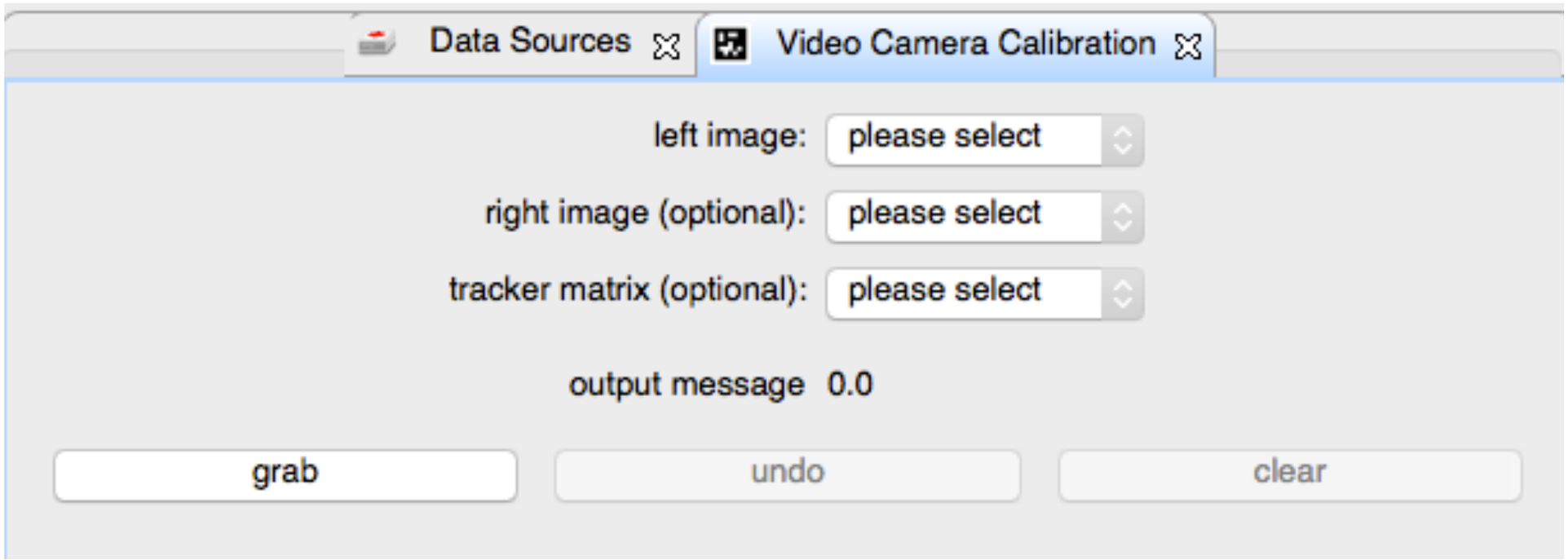
• Video Calibration Plugin

Any Video / Tracker



Method controlled by preferences

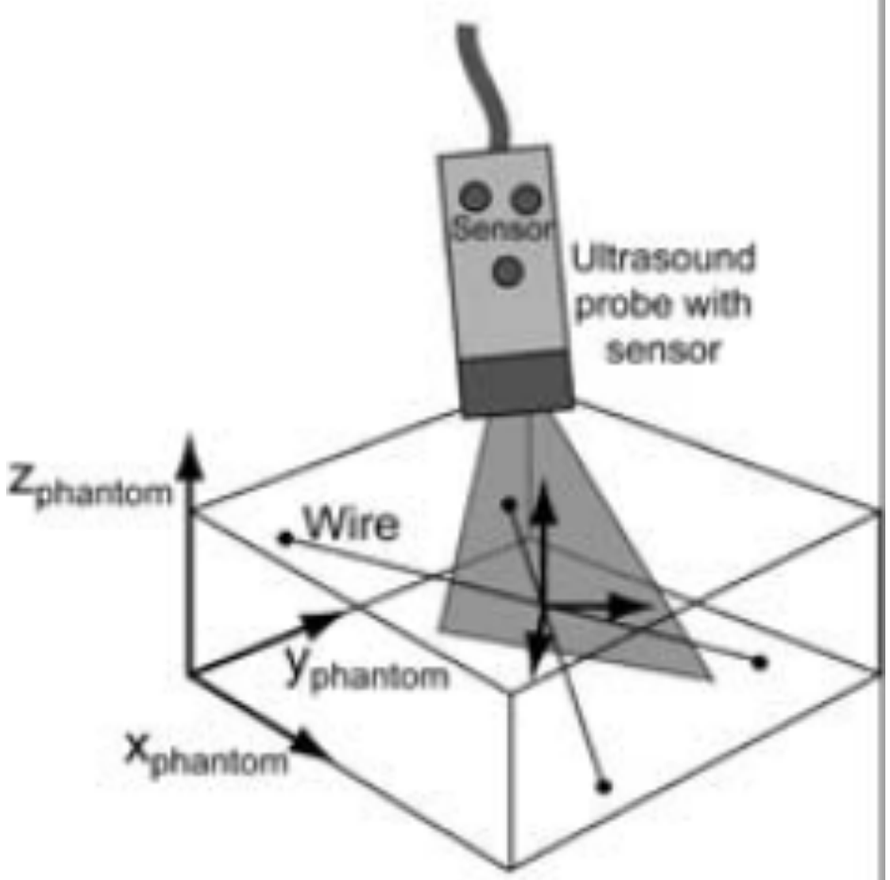




Mono
Stereo
Hand-Eye

- Short Summary
  - NiftyCal: smaller “research package”, integrated within NifTK
  - Implements:
    - Intrinsics: OpenCV, mono, stereo, varying number of points per view, chessboards, circle grids, April Tags, Zhang 2000, Tsai 1987, Datta 2009
    - Hand-eye: Tsai 1989, “Direct” method (Shahidi 2009), Malti et al 2013, Morgan 2017 in progress.

- Ultrasound Calibration - Pin, Cross-wire
  - 2005 Mercier et al: https://doi.org/10.1016/j.ultrasmedbio.2004.11.001



$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = T_{pw} \cdot T_{ws} \cdot T_{si} \cdot \begin{pmatrix} s_x \cdot u_k \\ s_y \cdot v_k \\ 0 \\ 1 \end{pmatrix}. \qquad (2)$$

- Record Data - Data Sources Plugin
- **niftkCalibrationPointClicker**
  - click on each 2D point
- **niftkUltrasoundPinCalibration**
  - compute calibration from points and tracking

- Ultrasound Calibration - Using Tracked Pointer

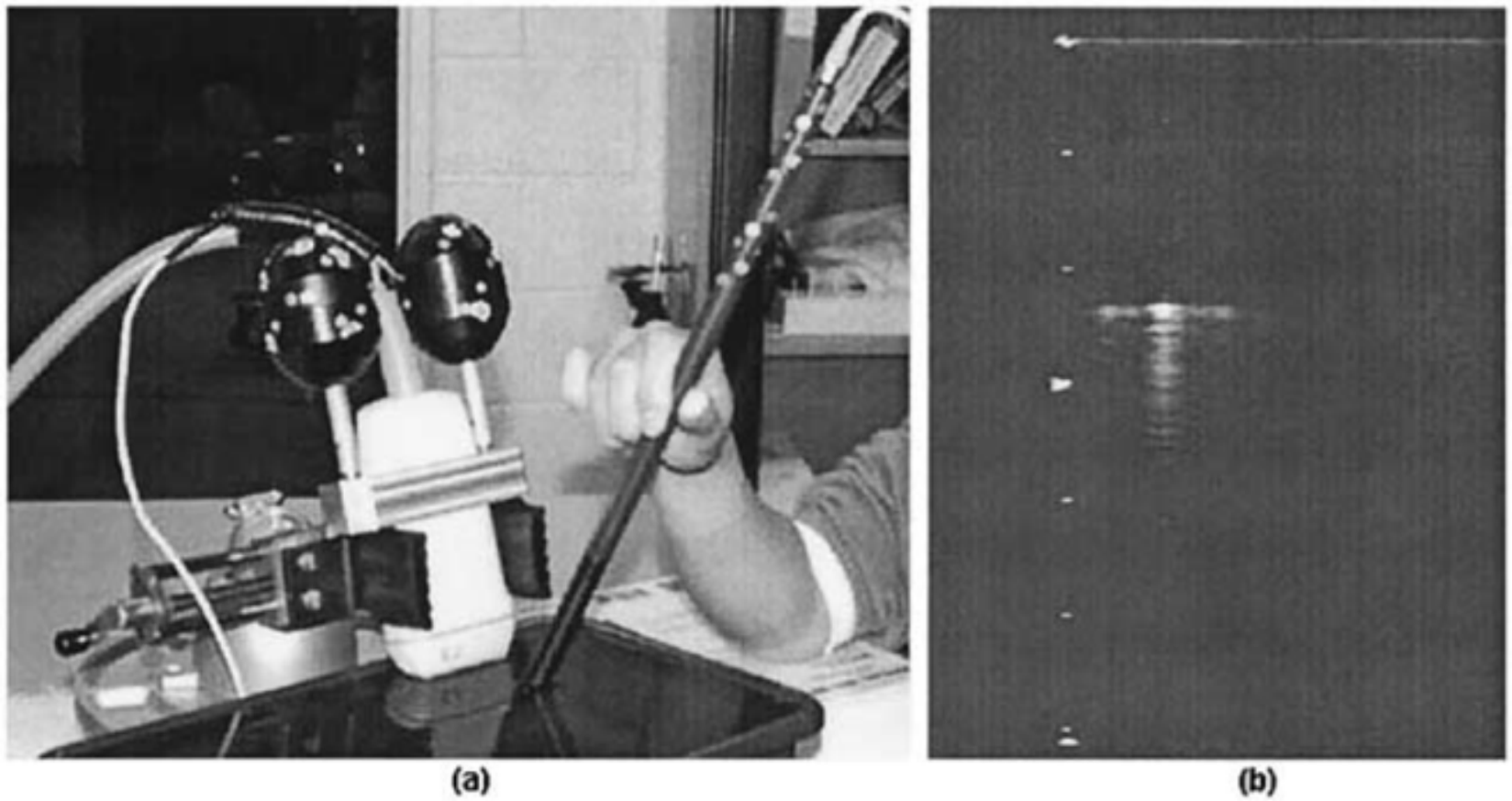
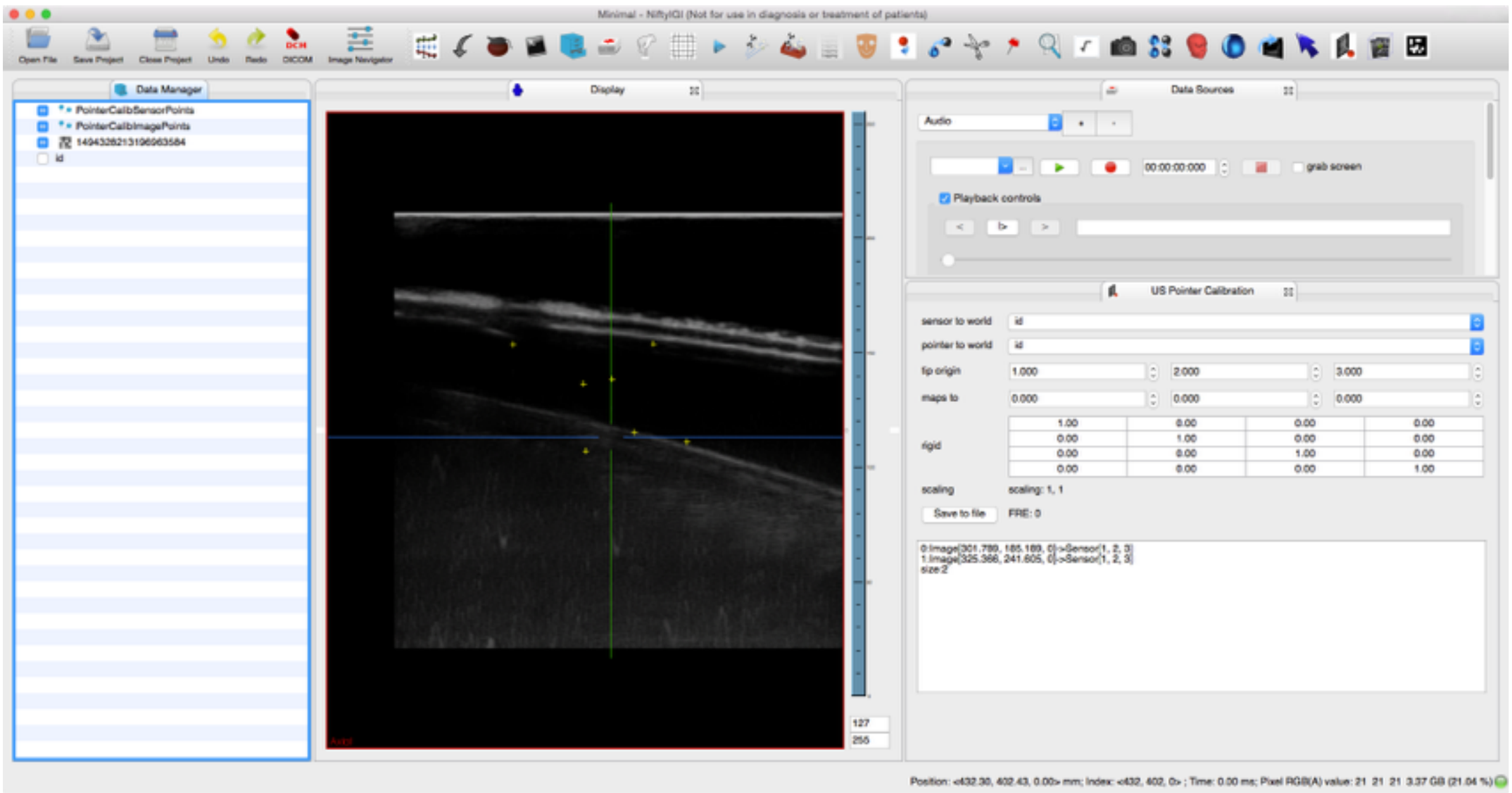


Fig. 3. The experimental setup. (a) The US signal from a tracked transducer is projected into a water bath while physical points are collected in its beam. (b) An US image of the pointer tip.

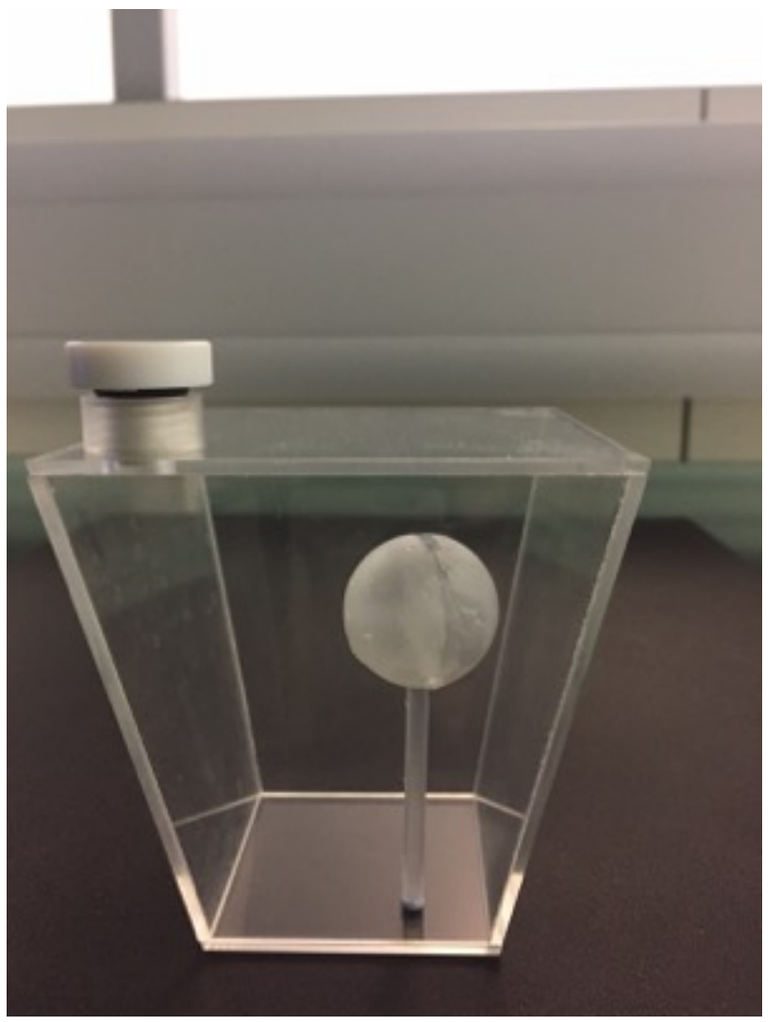
Muratore et al. http://dx.doi.org/https://doi.org/10.1016/S0301-5629(01)00469-0

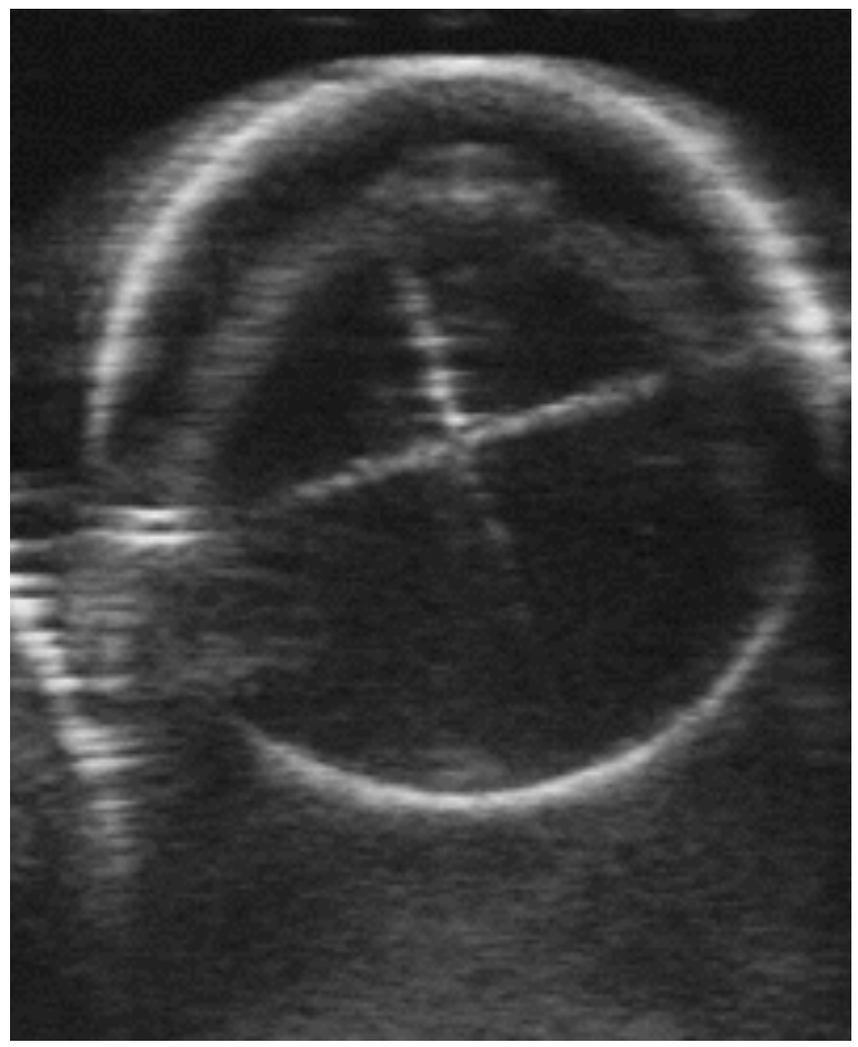
- Ultrasound Calibration - Using Tracked Pointer
  - (best with 2 people)

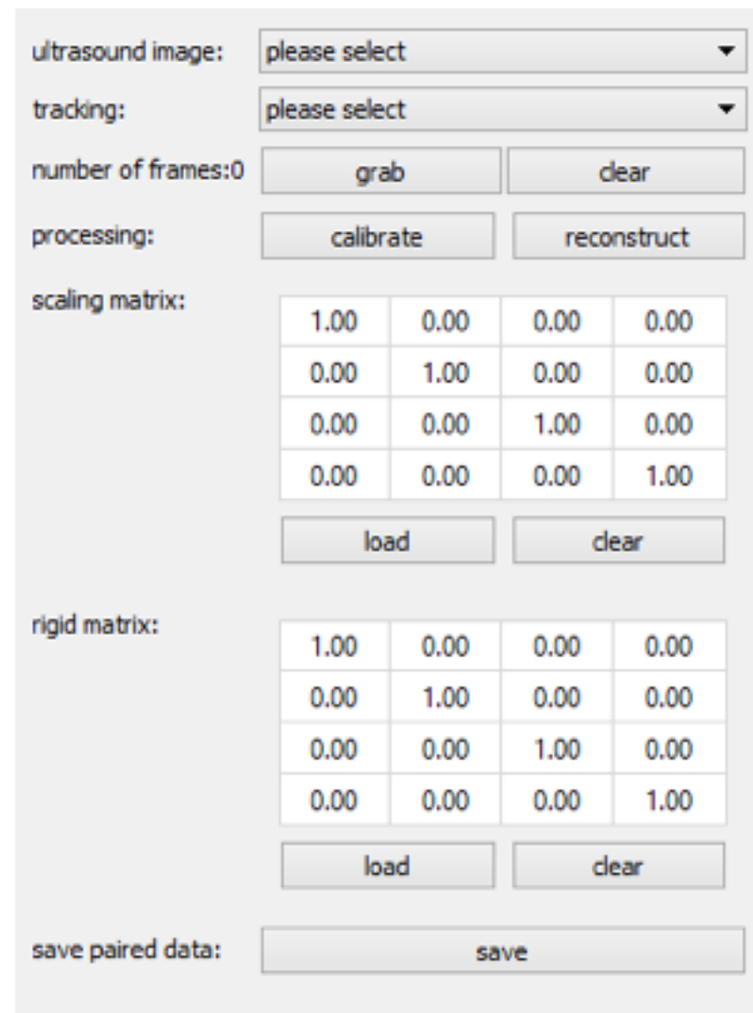


Muratore et al. http://dx.doi.org/https://doi.org/10.1016/S0301-5629(01)00469-0

- Automated Ultrasound Calibration (no clicking)
- Ping-pong ball in water bath
- Detect great circle - gives you image point

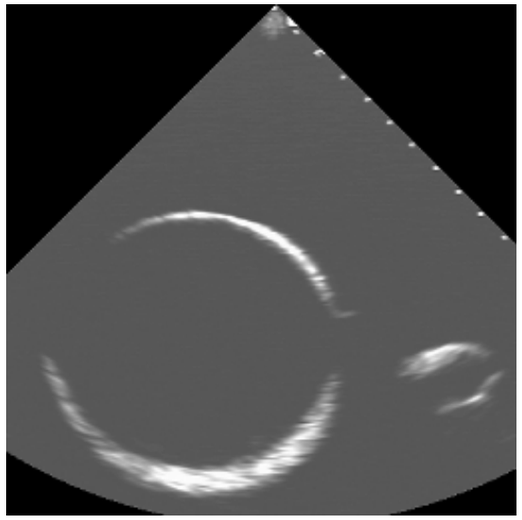


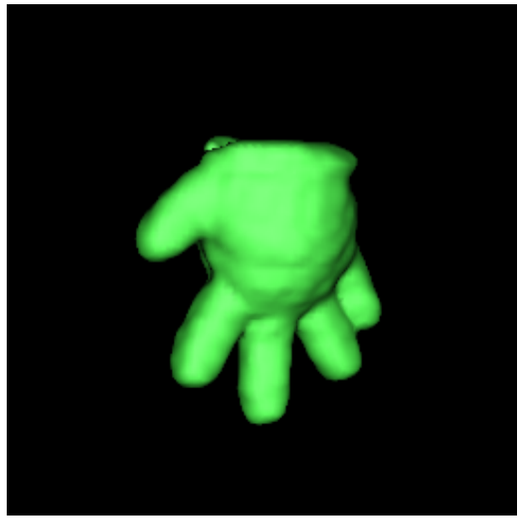


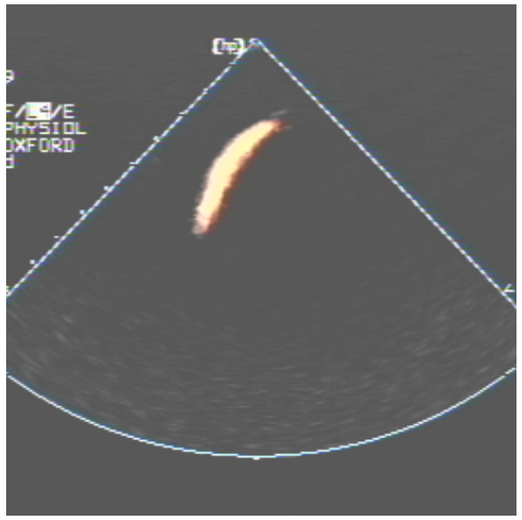


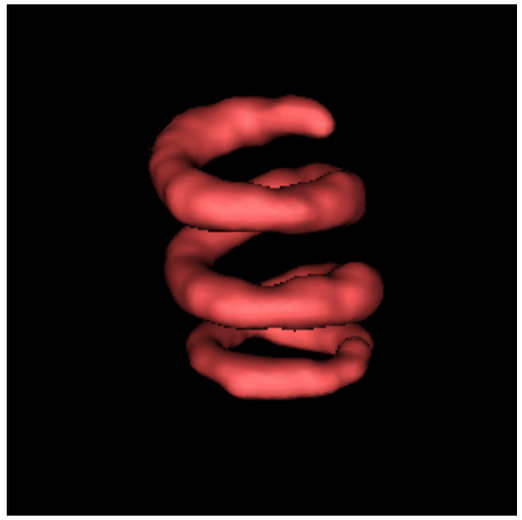
Guofang, Daniil, unpublished.

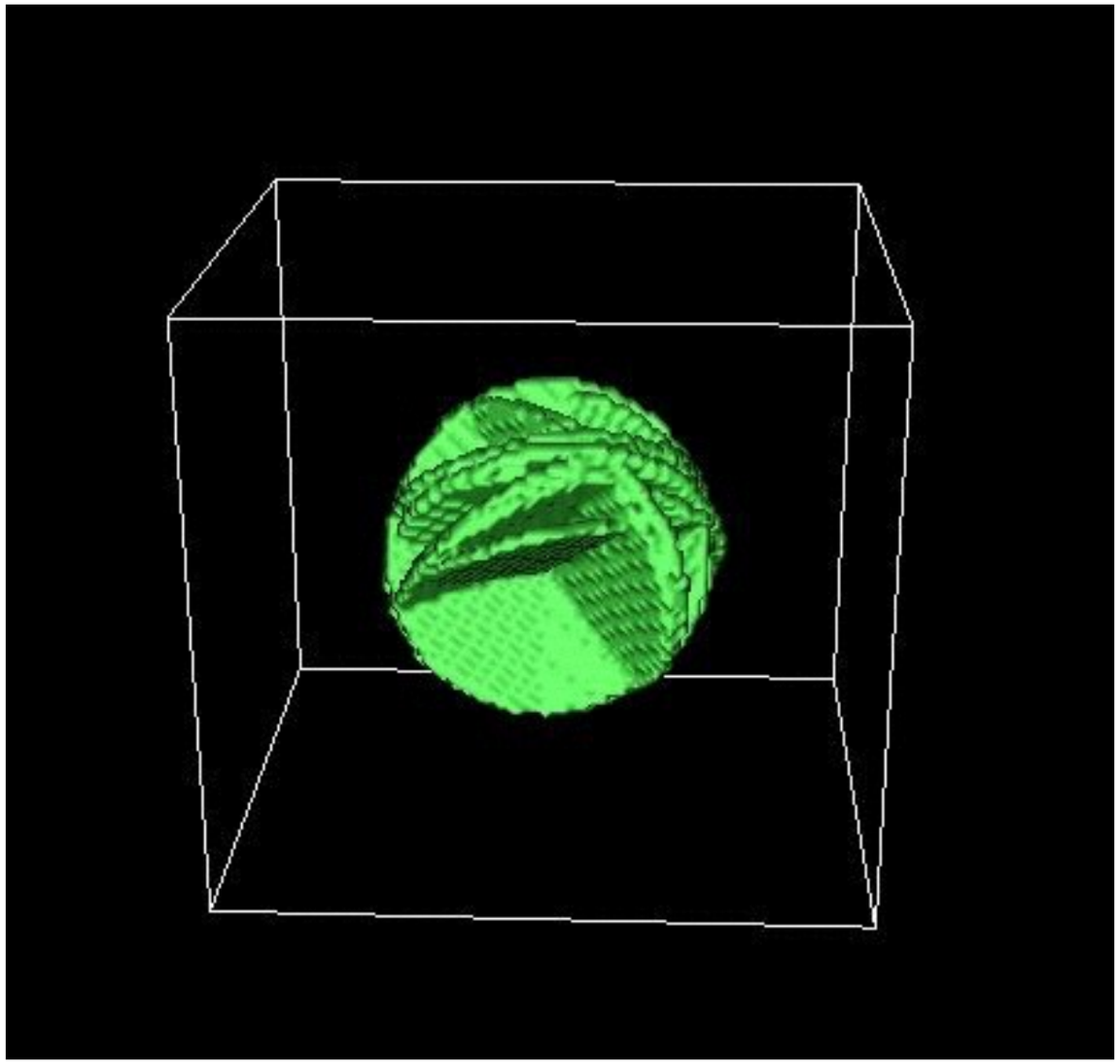
- Automated Ultrasound Calibration (no clicking)
- NiftyIGI plugin for calibration/reconstruction
  - Single Grab, continuous record.







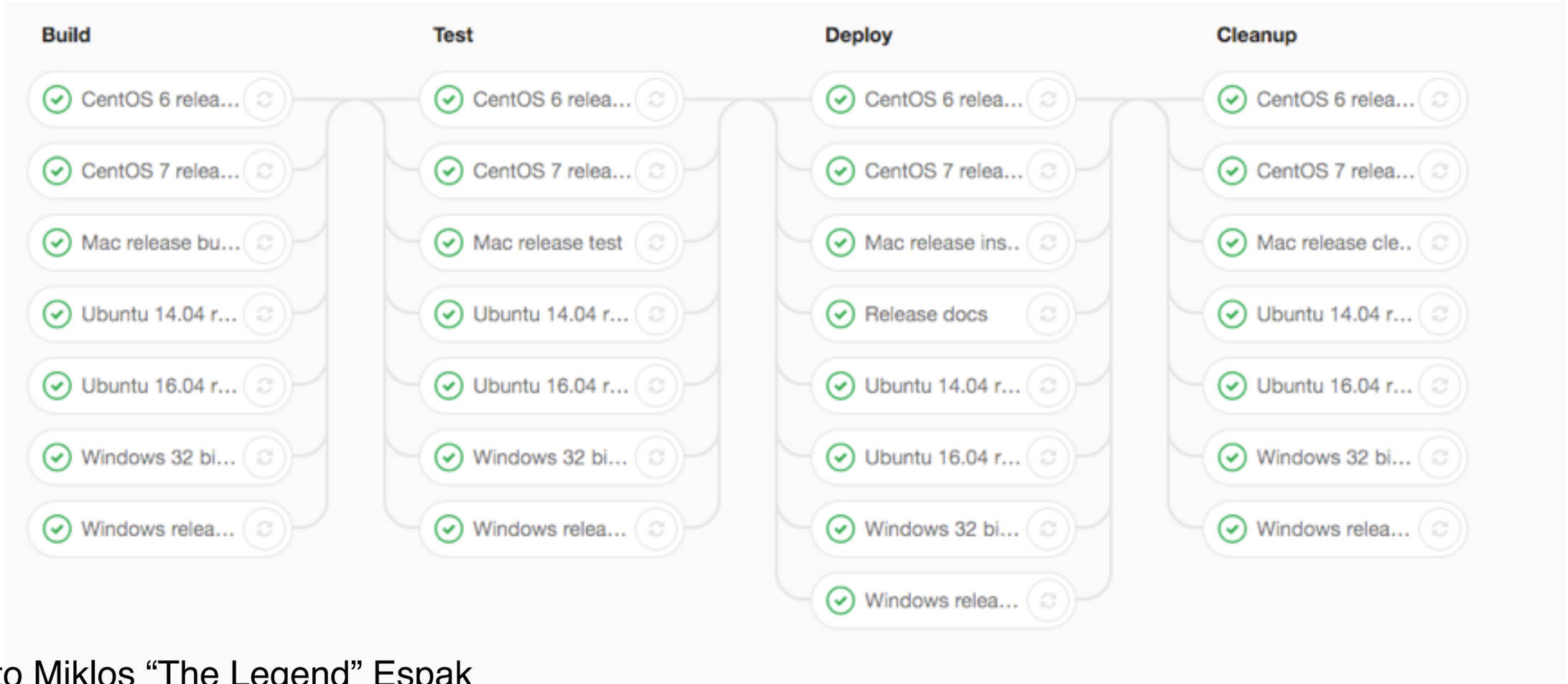




- Summary
  - Lots of calibration, recording methods
  - Don’t repeat all this!

- NifTK releases
  - Windows, Linux, Mac - generally, depends on hardware
  - CS managed machines
  - Cluster machines

- The $1,000,000 question - Can I build it?
- C.I servers - reliable build
  - BUT: Platform, compiler, CMake, Qt, 64/32, Debug/Release
  - Need a sanitised machine



Thanks to Miklos “The Legend” Espak

- The $1,000,000 question - Can I build it?
  - Build instructions: - Fairly thorough

NifTK 17.5.1 - b3a6660

CMIC's Translational Medical Imaging Platform

Main Page | Related Pages | Modules | Namespaces | Classes | Files

The NifTK Documentation

Build Instructions

Are you really a NifTK developer?

If you just want to use NifTK, and don't plan on extending it, and don't need up to the minute code changes, then you can just use the binary packages that can be download from the followin

- https://cmiclab.cs.ucl.ac.uk/CMIC/NifTK/install

If you are keen to build your own then read on.

The super-build process

NifTK and most of its dependencies are built using CMake, through a single process called 'super-build'. However, the super-build can't be expected to do absolutely everything. The number build assumes that Qt, Doxygen, Git and certain system libraries are in place. In effect the super-build is really taking care of the application specific libraries, not the necessary system librari

The application specific dependencies (Boost, VTK, ITK etc.) are automatically downloaded from:

```
http://cmic.cs.ucl.ac.uk/platform/dependencies
```

which can be accessed on the CS Unix filesystem here:

```
/cs/sys/www0/marine/html/cmic.cs.ucl.ac.uk/platform/dependencies
```

The directory is mounted on the CS managed Linux machines and e.g. jet.cs.ucl.ac.uk.

In principal, the super-build concept is identical on Linux, Windows and Mac, and it considerably simplifies the build process. The steps are summarised below, with screenshots from a Mac.

1. Before You Get Started - Known Caveats

- The $1,000,000 question - Can I build it?
  - Don’t wait 2 weeks, then complain
  - Ask for all the versions first.

- Virtual Box
  - Standard guaranteed development environment
  - Ubuntu 16.04 - NifTK
    - AprilTags, ArUco, Boost, Caffe, CTK, DCMTK, Eigen, GDCM, gflags, glog, PCL, HDF5, ITK, MITK, NiftyCal, NiftyLink, NiftyReg, NiftySeg, NiftySim, OpenCV, ProtoBuf, SimpleITK, SlicerExecutionModel, VL, VTK
    - No GPU

- WEISS centre
  - Core infrastructure support
  - More IGI / platform projects

- Getting involved?
  - Until now - project specific teams
  - Developers - Tuesday 11:00 - NifTK meeting

- Developing Using NifTK
  - We need Tutorials
    - Command Line Apps: ITK, VTK, PCL, OpenCV
    - Modules: Unit Testing
    - Plugins:
    - GUI:
    - Specific User Tasks
    - Please contribute suggestions

- Standardised build?
  - e.g. Ubuntu 16.04 - with RoS

- Getting Involved
  - Mailing list: https://www.mailinglists.ucl.ac.uk/mailman/listinfo/niftk-developers

- To Do:
  - Use IGI stuff: more testers the better, contribute to research
  - Improve docs: easy way to learn
  - Contribute to tickets / discussion

- New Features
  - Atracsys, ultrasound reconstruction, GPU Direct
  - Data record/playback
  - Deformable modelling? Real-time GPU video?

Thank you