How to build your own 3D scanner

Jakob Wilm, PhD Student

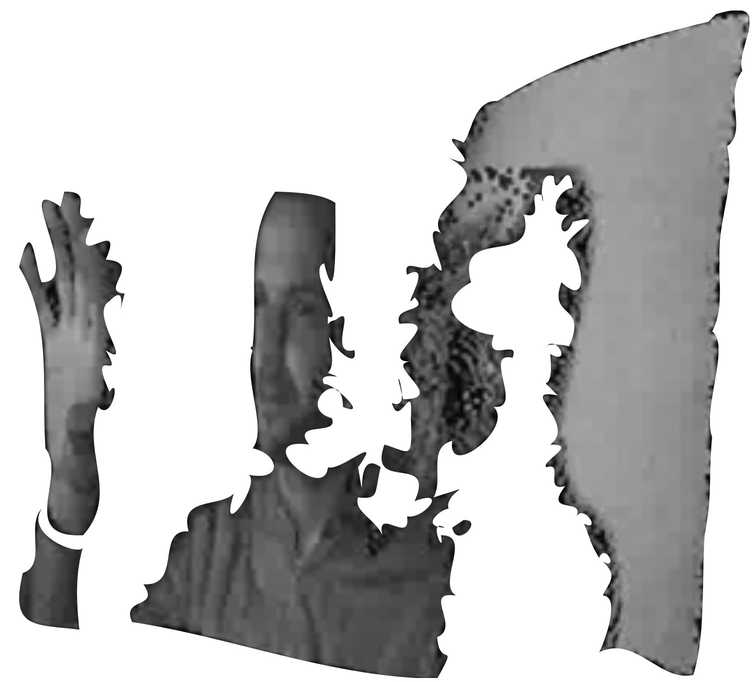
Section for Image Analysis and Computer Graphics, DTU Informatics
Department of Clinical Physiology, Nuclear Medicine and PET, Rigshospitalet

Microsoft Kinect

Microsoft Kinect 2

Asus Xtion

MESA SwissRanger



Occipital Structure

PMDTec Camboard Pico

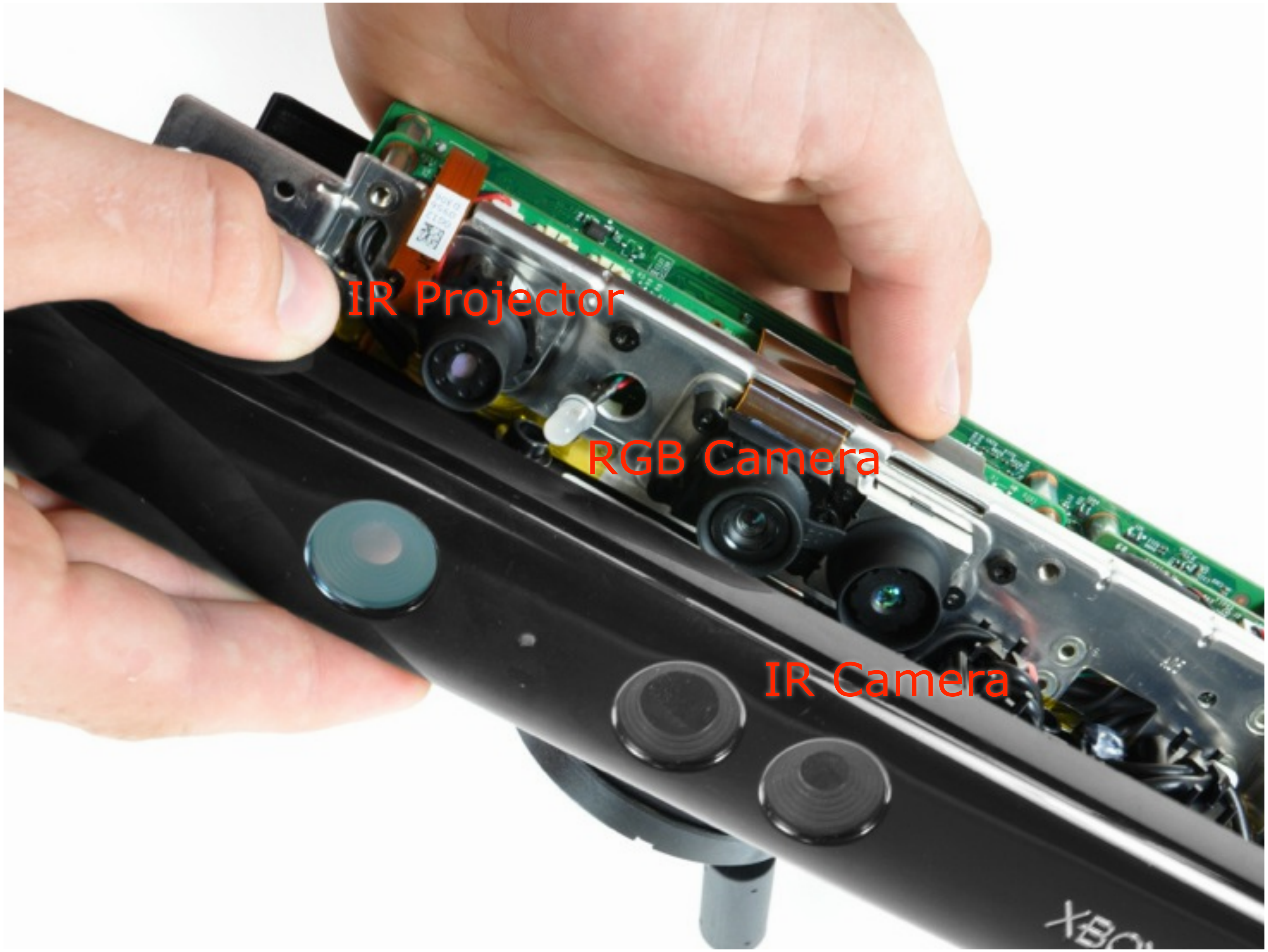
Panasonic d-Imager

Point Grey Bumble Bee

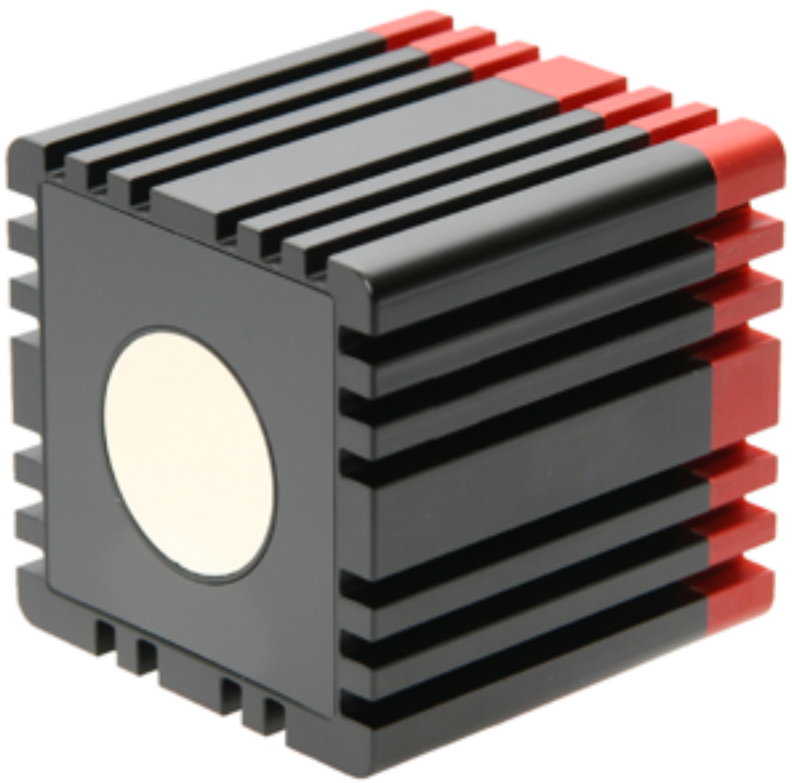
Fujifilm FinePix Real 3D

Microsoft Kinect

Cost	dkr 1.600,-
Resolution	640 x 480
Speed	25 fps
Accuracy	~1 mm
Robustness	high
Hackability	;-|

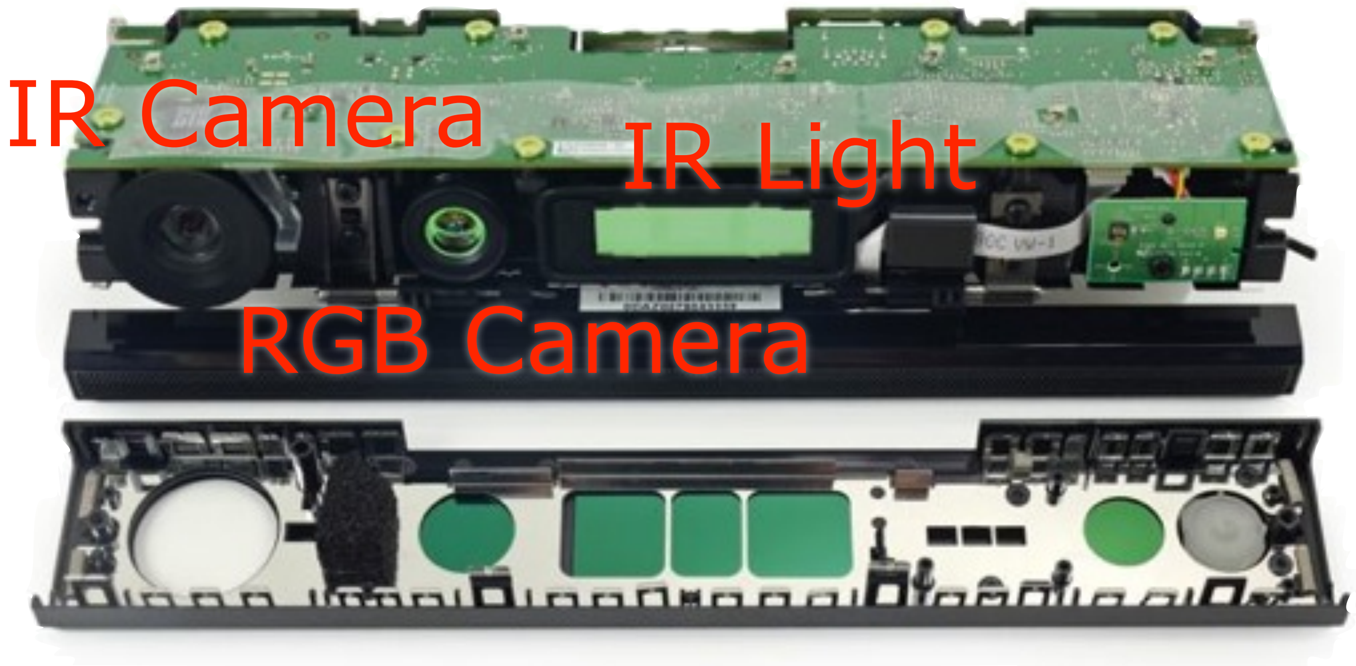

IR Projector
RGB Camera
IR Camera

Mesa SwissRanger SR4000



Cost	dkr 22.000,-
Resolution	176 x 144
Speed	54 fps
Accuracy	~5 mm
Robustness	low
Hackability	;-(

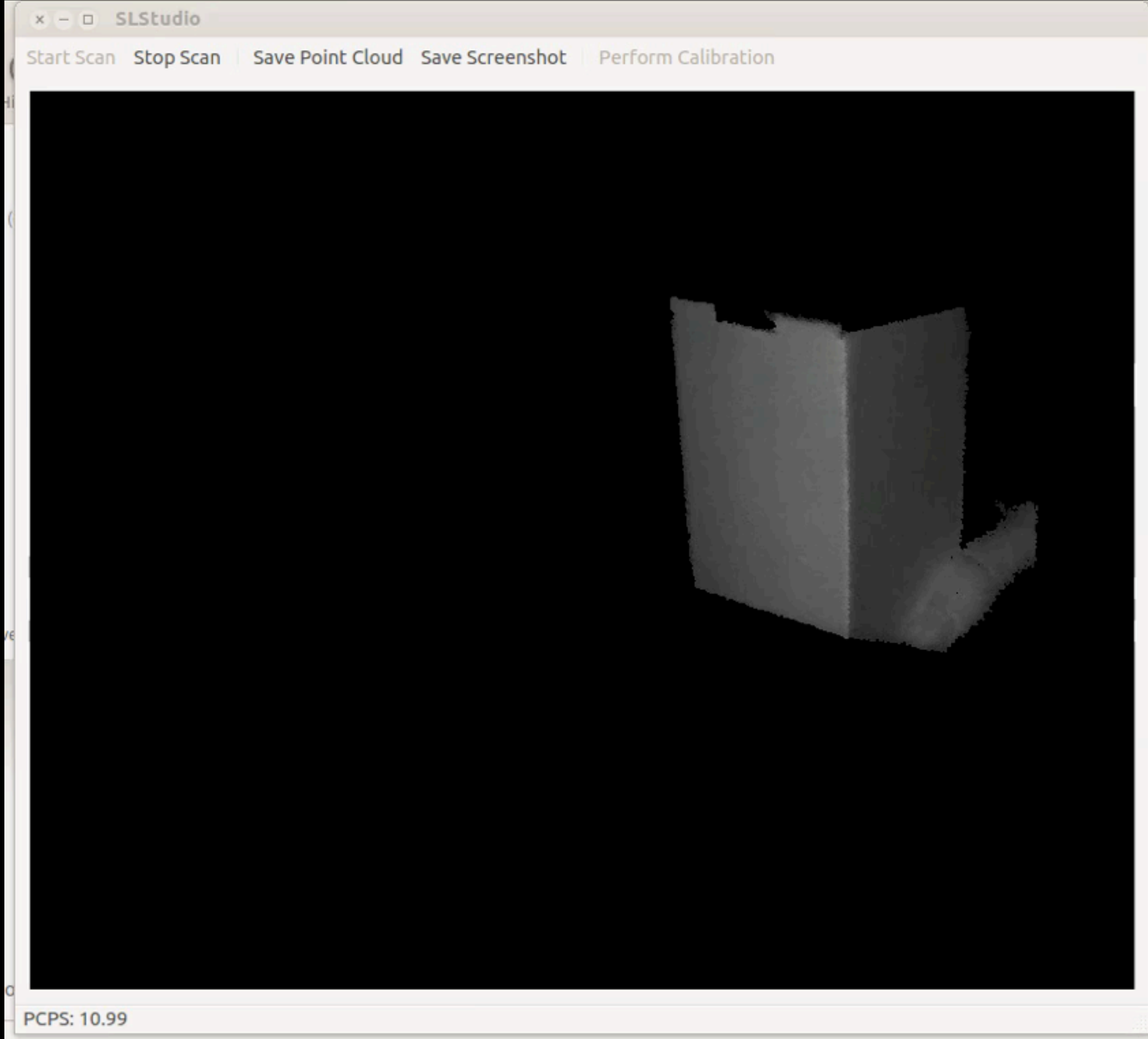
Microsoft Kinect 2



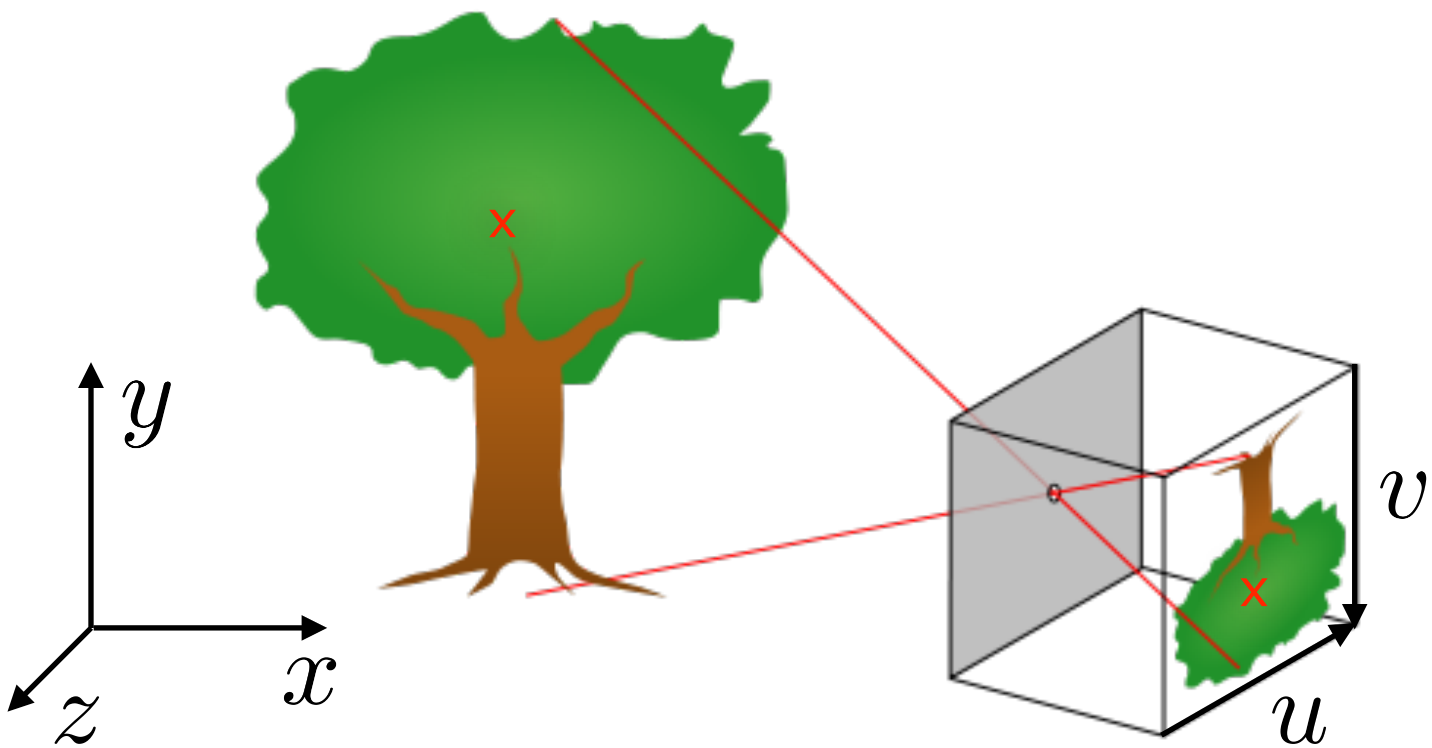
Cost	dkr 2.000,-
Resolution	512 x 424
Speed	30 fps
Accuracy	~5 mm
Robustness	good
Hackability	;-|

DTU Scanner — SLStudio

Cost	dkr 10.000,-
Resolution	variable
Speed	variable
Accuracy	variable
Robustness	variable
Hackability	;->

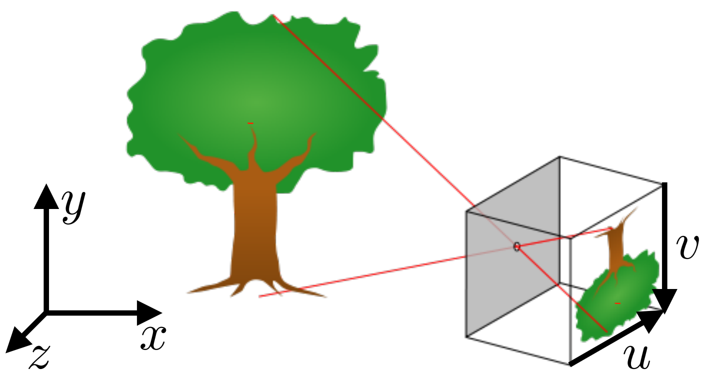

SLStudio
Start Scan
Stop Scan
Save Point Cloud
Save Screenshot
Perform Calibration
PCPS: 10.99

Triangulation



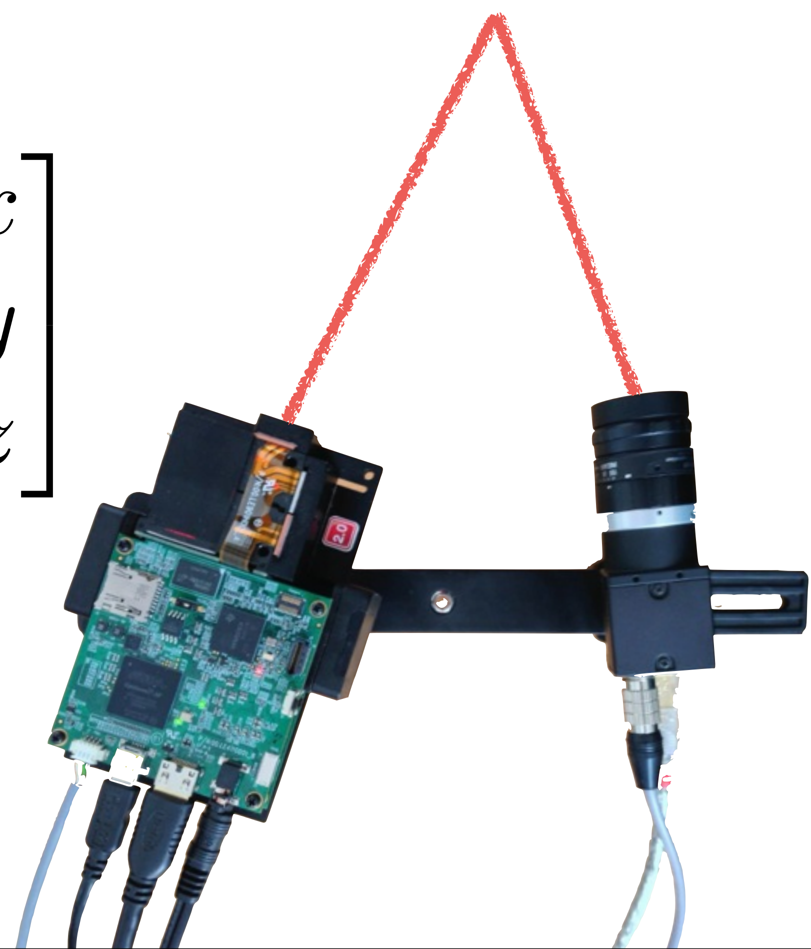
$$\overbrace{}^{\boldsymbol{K}}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \approx \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Triangulation



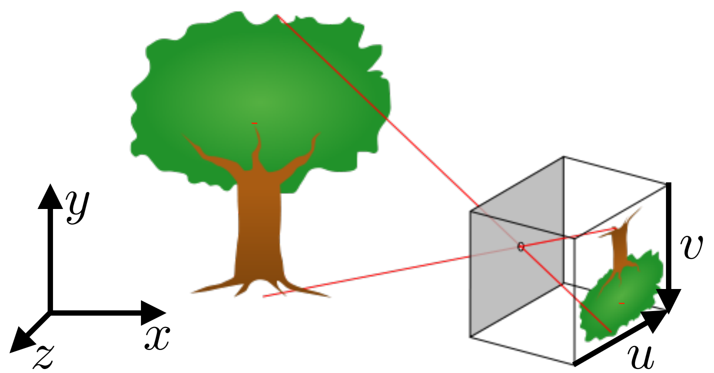
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \approx \overbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}^{\boldsymbol{K}} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} \approx \overbrace{\boldsymbol{K}_p \left[\boldsymbol{R} | \boldsymbol{T}\right]}^{\boldsymbol{P}_p} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



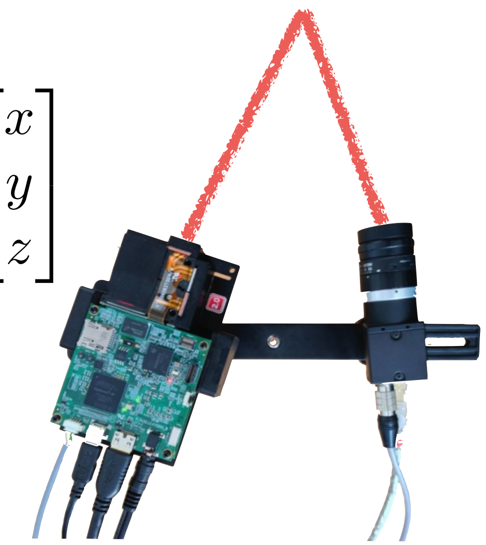
$$\begin{bmatrix} u_c \\ v_c \\ 1 \end{bmatrix} \approx \boldsymbol{K}_c \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Triangulation



$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \approx \overbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}^{\boldsymbol{K}} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

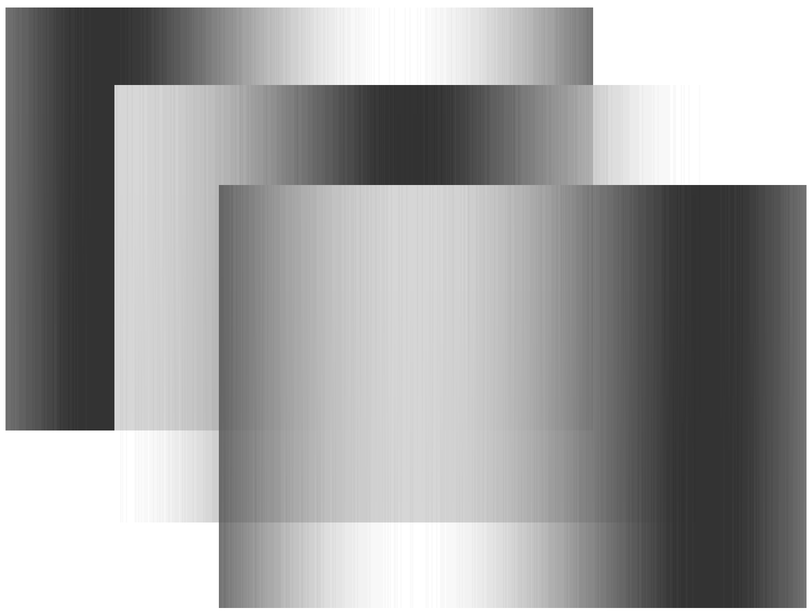
$$\begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} \approx \overbrace{\boldsymbol{K}_p \, [\boldsymbol{R}|\boldsymbol{T}]}^{\boldsymbol{P}_p} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

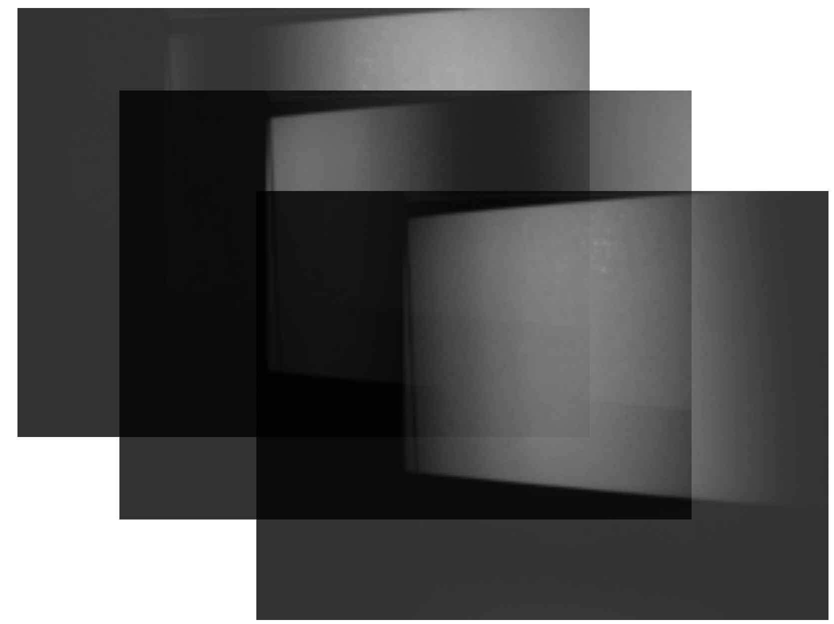


$$\begin{bmatrix} u_c \\ v_c \\ 1 \end{bmatrix} \approx \boldsymbol{K}_c \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} u_c \boldsymbol{P}_c(3) - \boldsymbol{P}_c(1) \\ v_c \boldsymbol{P}_c(3) - \boldsymbol{P}_c(2) \\ u_p \boldsymbol{P}_p(3) - \boldsymbol{P}_p(1) \end{bmatrix} \cdot Q = 0$$

Phase Shifting Interferometry





$$I_n^p(x^p, y^p) = \frac{1}{2} + \frac{1}{2}\cos\left(2\pi\left(\frac{n}{N} - y^p\right)\right)$$

$$I_n^c(x^c, y^c) = A^c + B^c\cos\left(\frac{2\pi n}{N} - \theta\right)$$

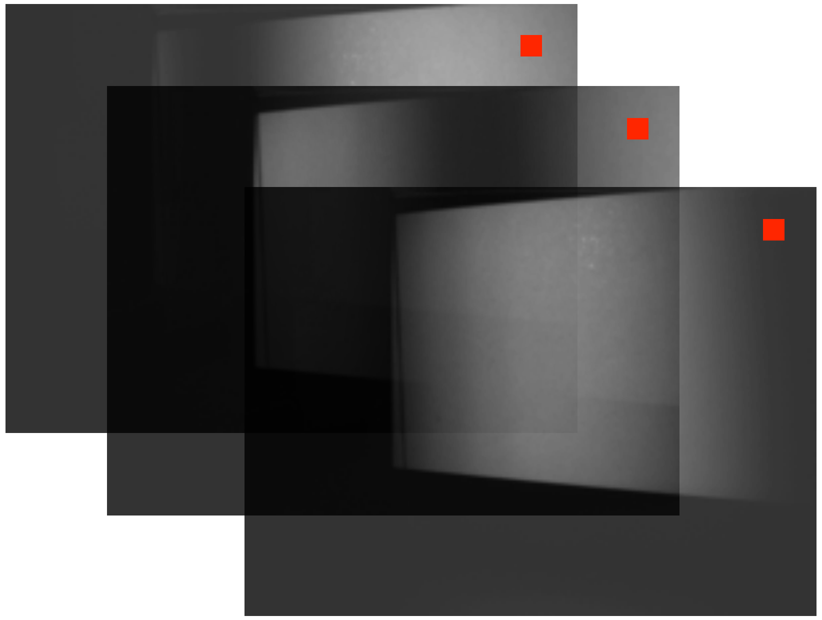
$$B_{\mathcal{R}}^c = \sum_{n=0}^{N-1} I_n^c(x^c, y^c)\cos\left(\frac{2\pi n}{N}\right)$$

$$\theta = \angle(B_{\mathcal{R}}^c + jB_{\mathcal{I}}^c) = \arctan\left\{\frac{B_{\mathcal{I}}^c}{B_{\mathcal{R}}^c}\right\}$$

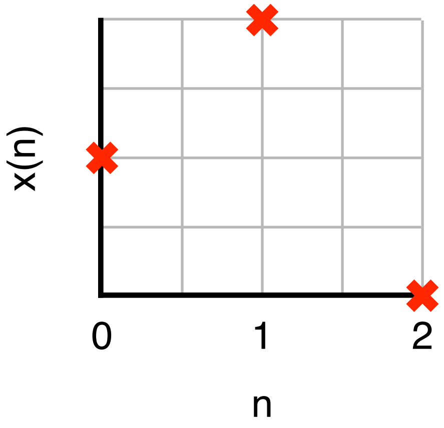
$$B_{\mathcal{I}}^c = \sum_{n=0}^{N-1} I_n^c(x^c, y^c)\sin\left(\frac{2\pi n}{N}\right)$$

$$A^c = \frac{1}{N}\sum_{n=0}^{N-1} I_n^c(x^c, y^c)$$

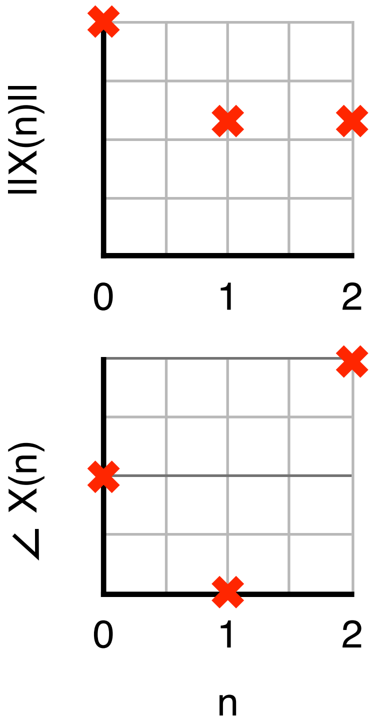
Frequence Domain Interpretation



N = 3



$$\mathcal{F}(x)$$



DC Component $$A^c = \frac{1}{N} X[0]$$

Energy $$B^c = \frac{2}{N} \|X[1]\| = \frac{2}{N} \|X[N-1]\|$$

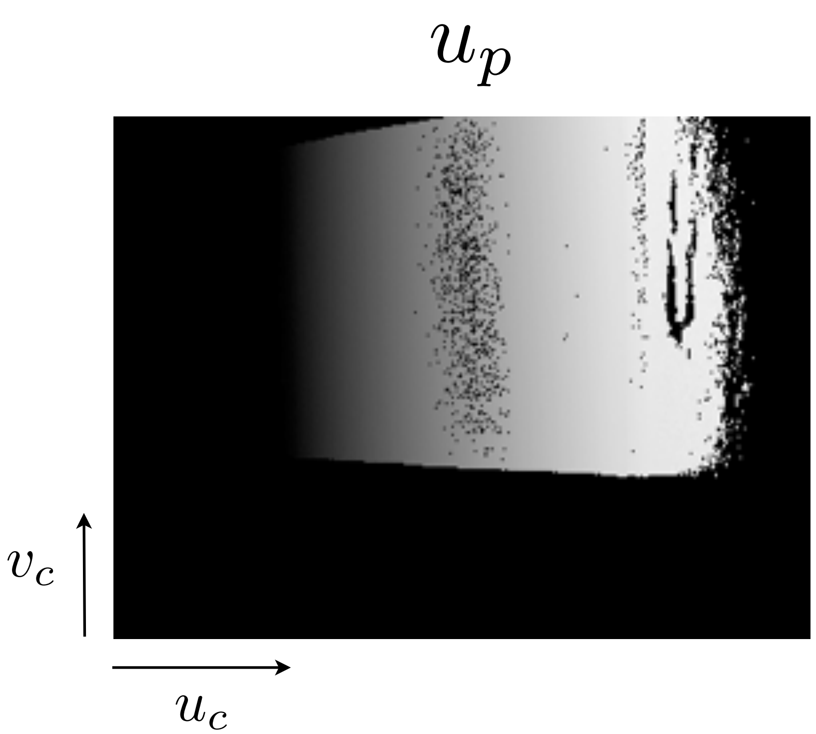
Phase $$\theta = \angle X[1] = -\angle X[N-1]$$

Lau, Liu, Hassebrook (2010). Opt letters, 35(14)

SLStudio

- Modular platform
- Enables 20fps pointclouds (3 frame PSI)
- Key components:
  - Projection interface
  - Industrial camera interface
  - Coding/Decoding & Fast reconstruction
  - Calibration
  - In development: Rigid body tracking
- ~10k LOC

Reconstruction



$$\begin{bmatrix} u_c \\ v_c \\ 1 \end{bmatrix} = \boldsymbol{P}_c Q \qquad \text{and} \qquad \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} = \boldsymbol{P}_p Q$$

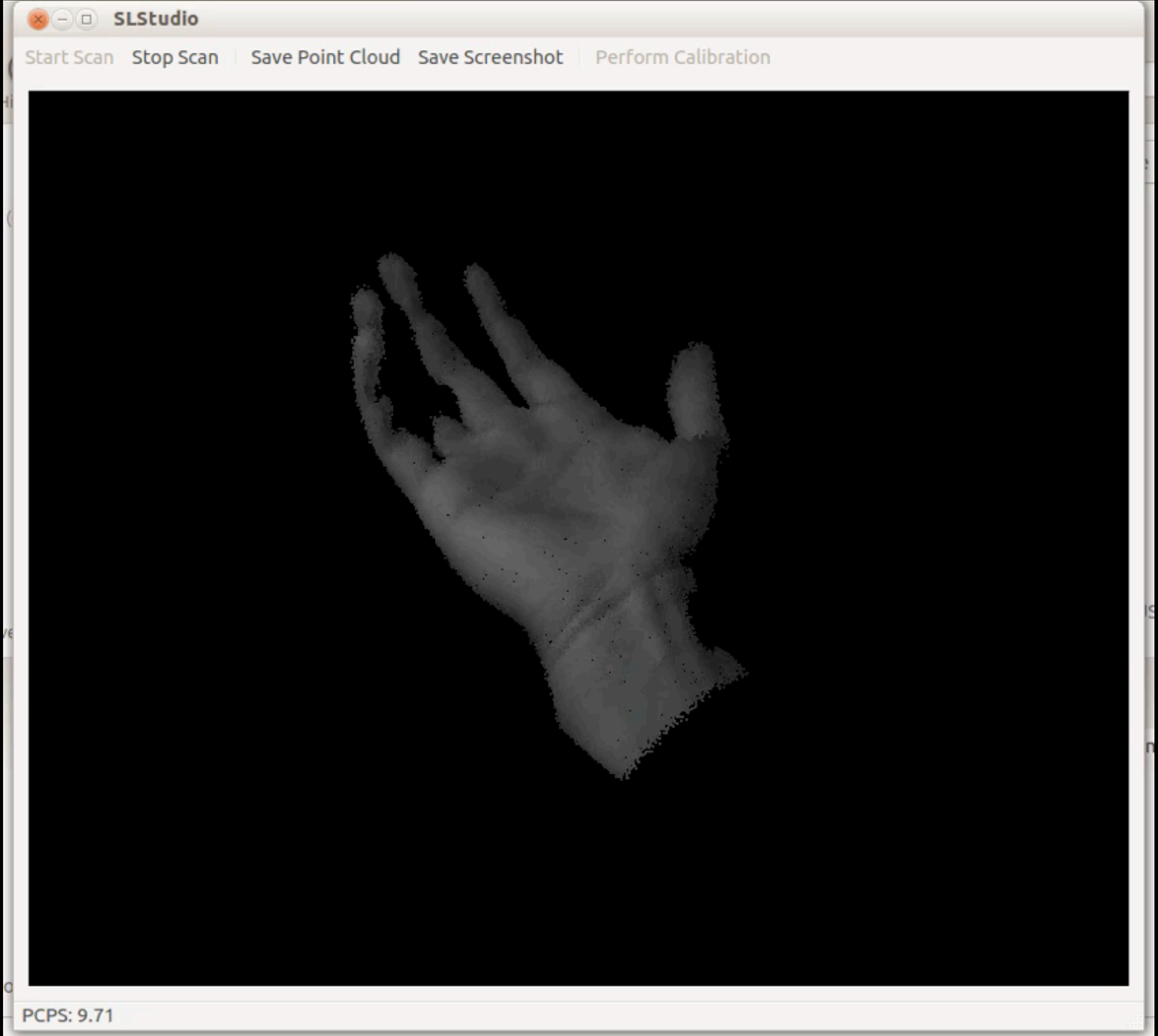
250k points = 20ms!!

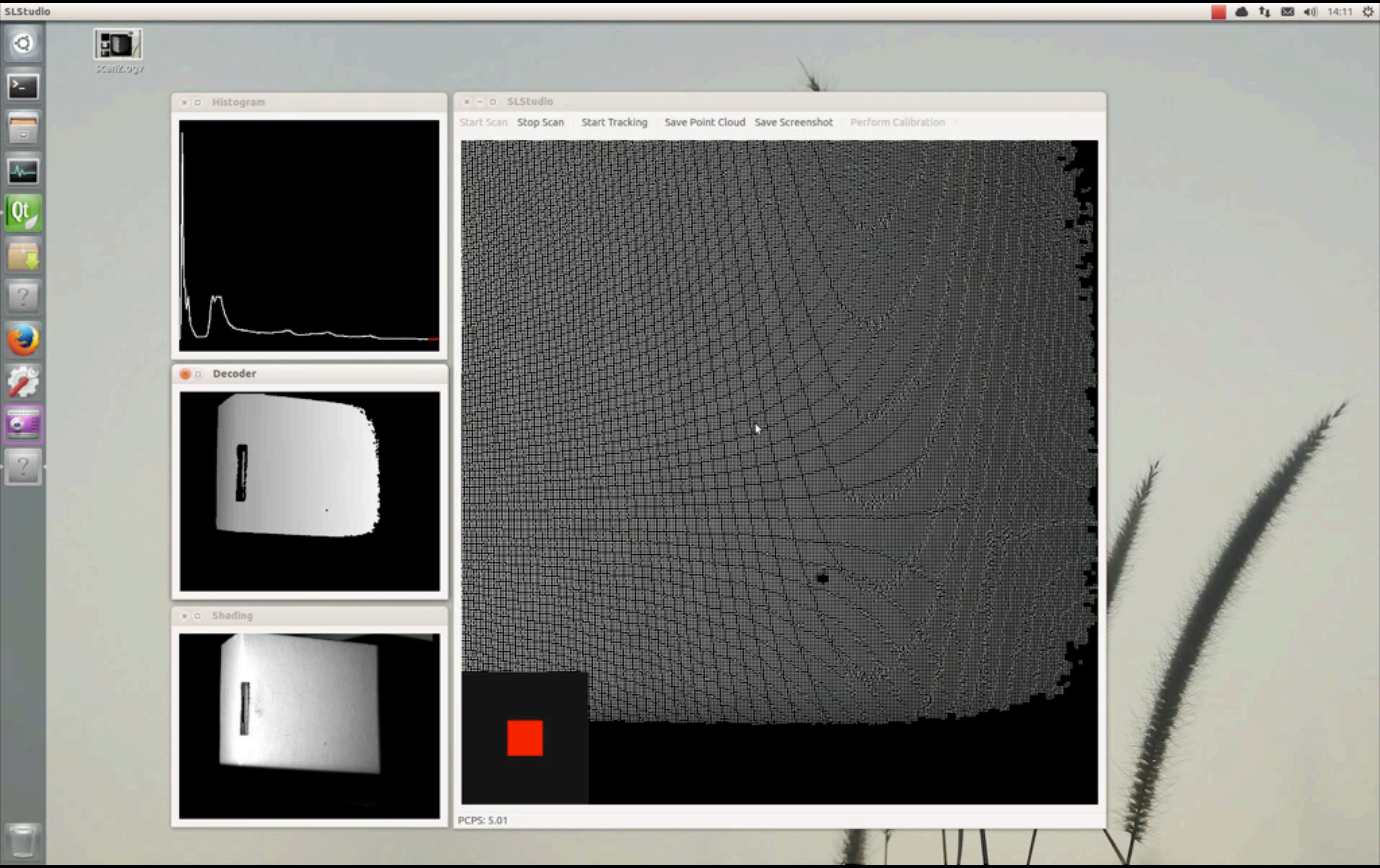
$$\begin{bmatrix} u_c \boldsymbol{P}_c(3) - \boldsymbol{P}_c(1) \\ v_c \boldsymbol{P}_c(3) - \boldsymbol{P}_c(2) \\ u_p \boldsymbol{P}_p(3) - \boldsymbol{P}_p(1) \end{bmatrix} \cdot Q = 0$$

$$\boldsymbol{C}^k_{i,j,l} = \det\left(\boldsymbol{P}_c(i), \boldsymbol{P}_c(j), \boldsymbol{P}_p(l), \boldsymbol{e}_k\right)$$

$$Q_k = \boldsymbol{C}^k_{1,2,1} - u_p \boldsymbol{C}^k_{3,2,1} - v_p \boldsymbol{C}^k_{1,3,1} - u_c \boldsymbol{C}^k_{1,2,2} + u_c u_p \boldsymbol{C}^k_{3,2,2} + u_c v_p \boldsymbol{C}^k_{1,3,2}$$

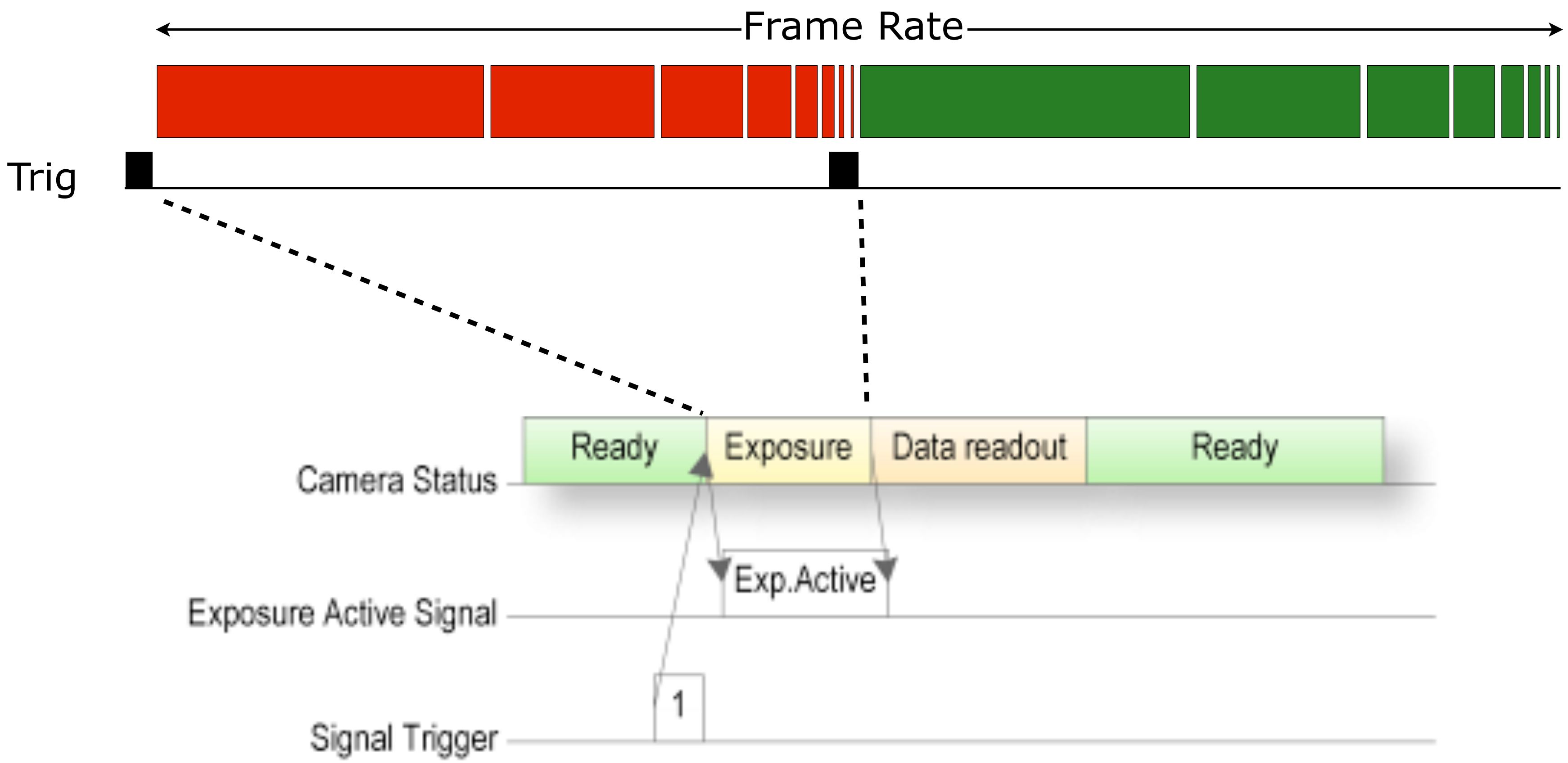
Valkenburg, McIvor (1998). Img Vis Comp. 16(2), 99–110.

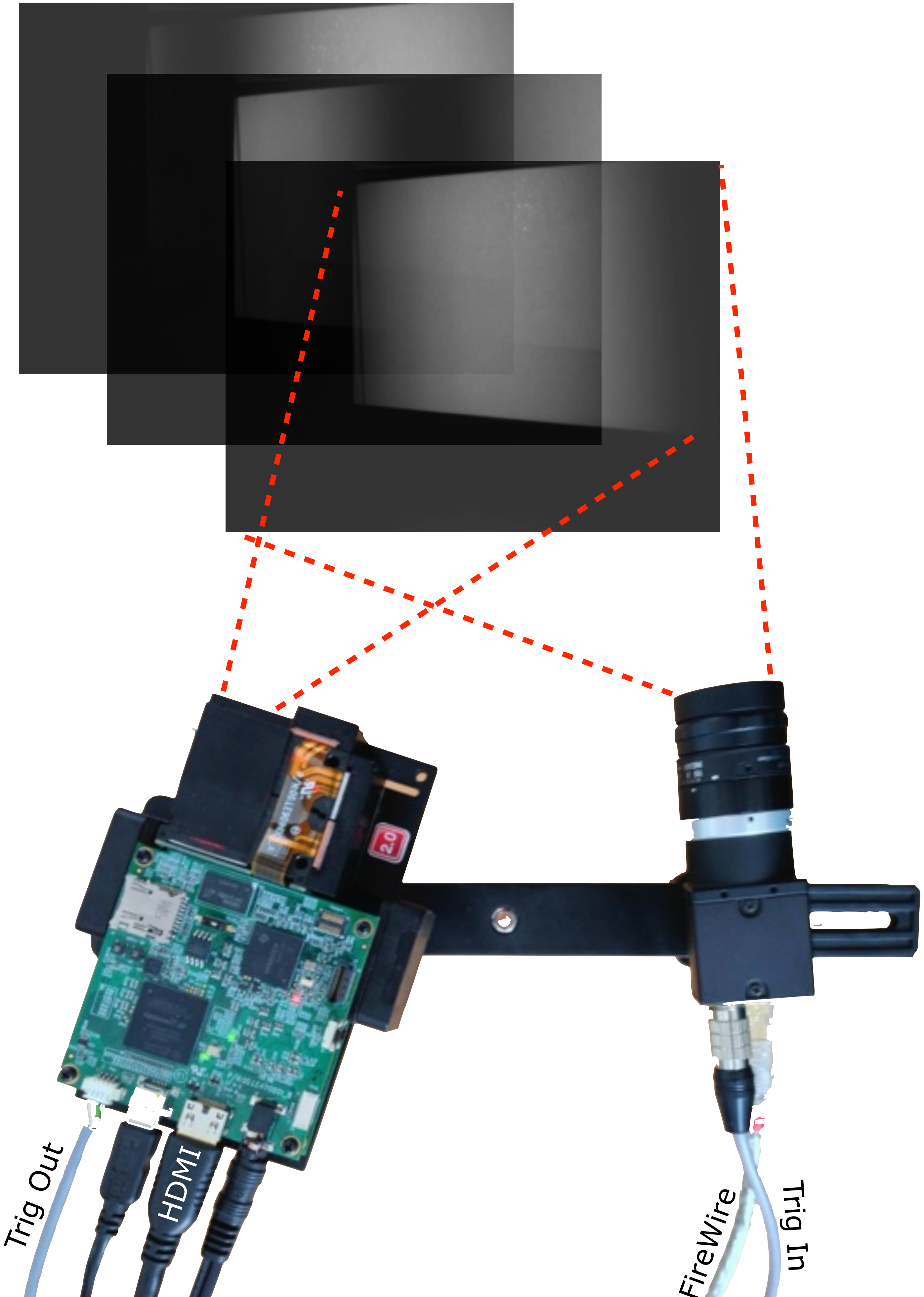

SLStudio
Start Scan
Stop Scan
Save Point Cloud
Save Screenshot
Perform Calibration
PCPS: 9.71

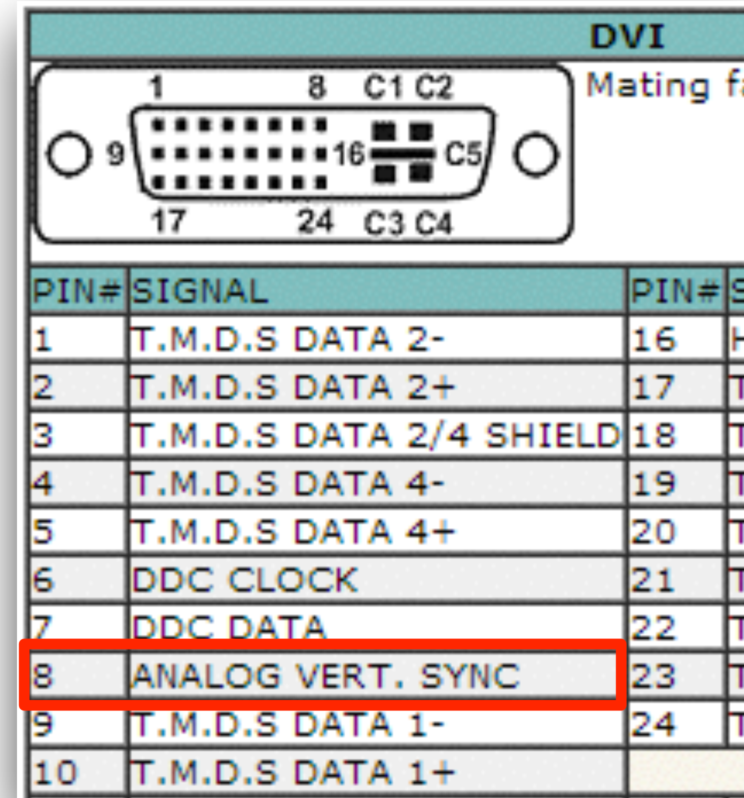

SLStudio
14:11
Histogram
Decoder
Shading
SLStudio
Start Scan
Stop Scan
Start Tracking
Save Point Cloud
Save Screenshot
Perform Calibration
PCPS: 5.01

Radiohead, House of Cards

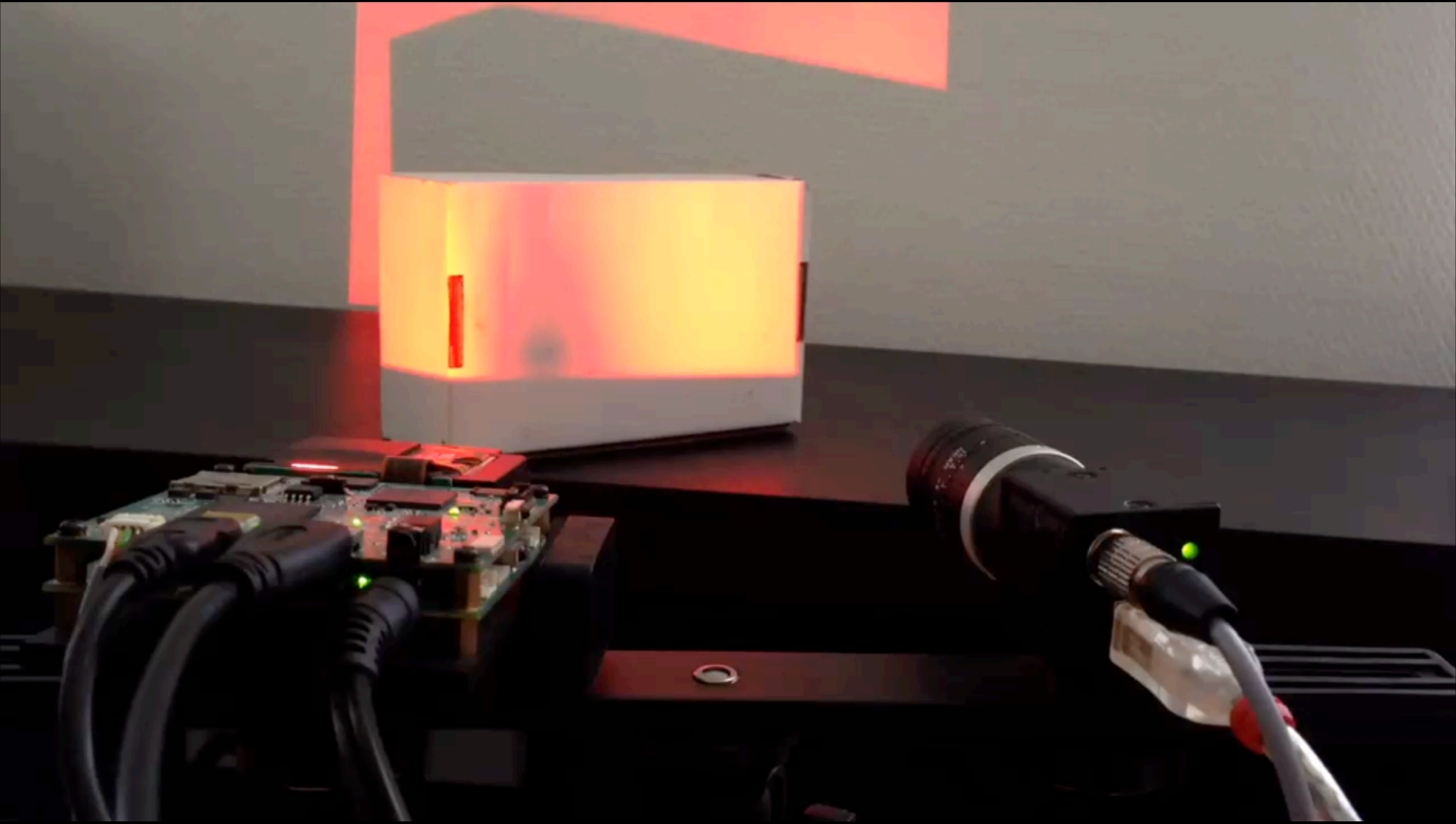
Hardware Trigger



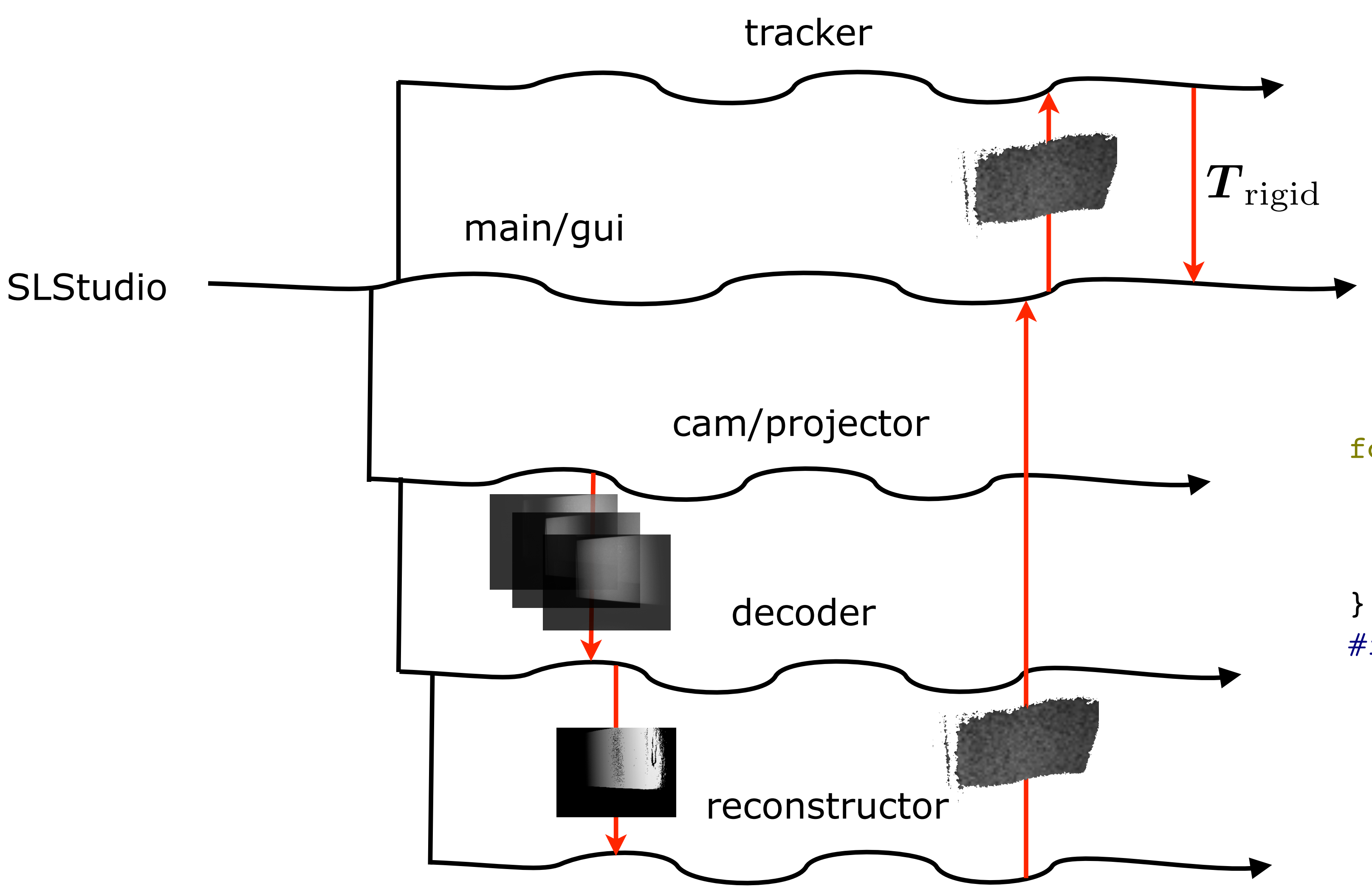




PIN#	SIGNAL	PIN#	SI
1	T.M.D.S DATA 2-	16	H
2	T.M.D.S DATA 2+	17	T.
3	T.M.D.S DATA 2/4 SHIELD	18	T.
4	T.M.D.S DATA 4-	19	T.
5	T.M.D.S DATA 4+	20	T.
6	DDC CLOCK	21	T.
7	DDC DATA	22	T.
8	ANALOG VERT. SYNC	23	T.
9	T.M.D.S DATA 1-	24	T.
10	T.M.D.S DATA 1+		



Parallelization

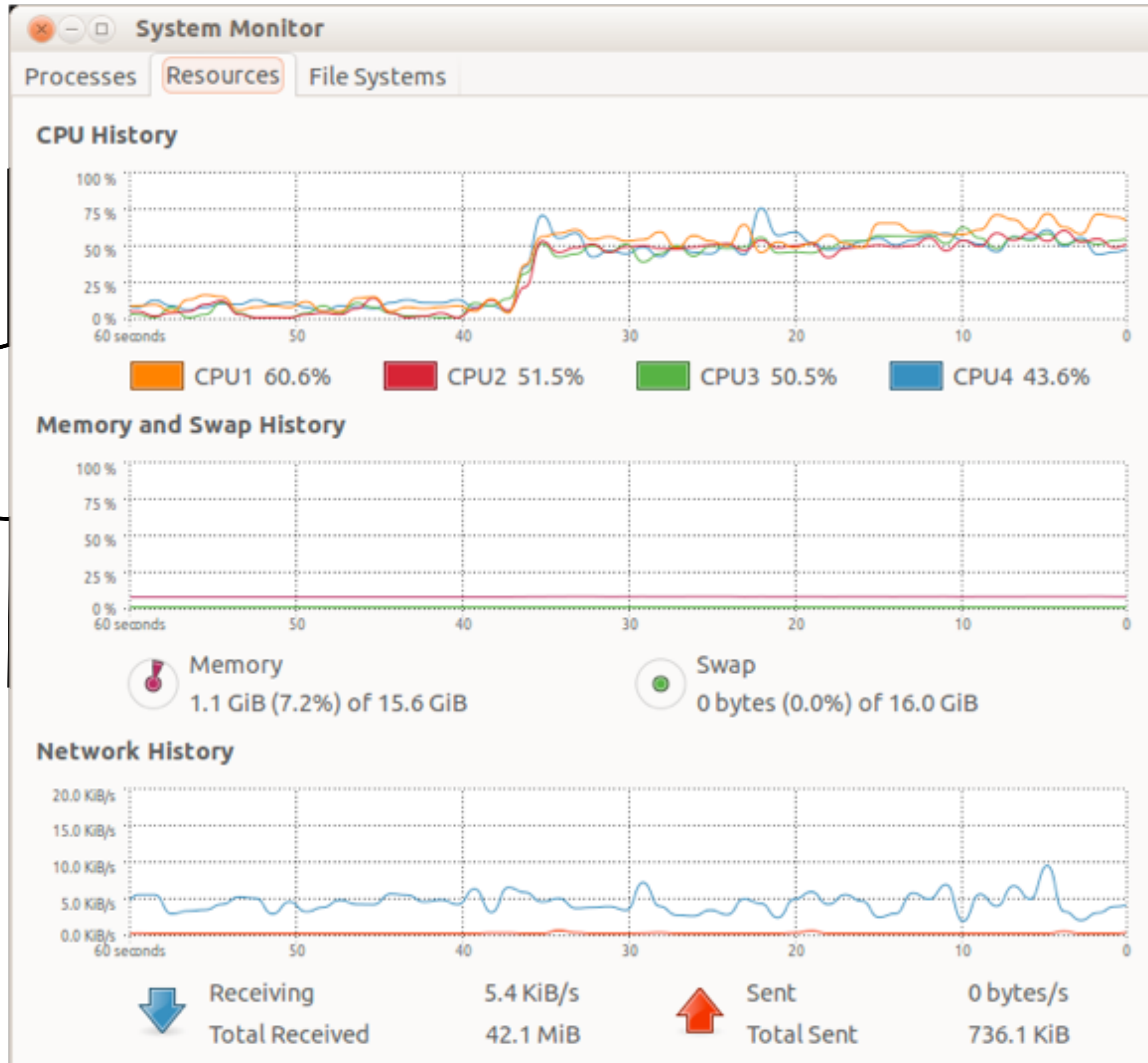


(0, 0)	(0, 1)	(0, 2)	(0, 3)
(1, 0)	(1, 1)	(1, 2)	(1, 3)
(2, 0)	(2, 1)	(2, 2)	(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

```
for(int i = 0; i < mat.rows; i++){
    for(int j = 0; j < mat.cols; j++){
        ...
    }
}
#if CV_SSE2
    if( USE_SSE2 )
    {
       ...
       __m128i r0 = _mm_loadu_si128(...);
       __m128i r1 = _mm_loadu_si128(...);
```

Parallelization

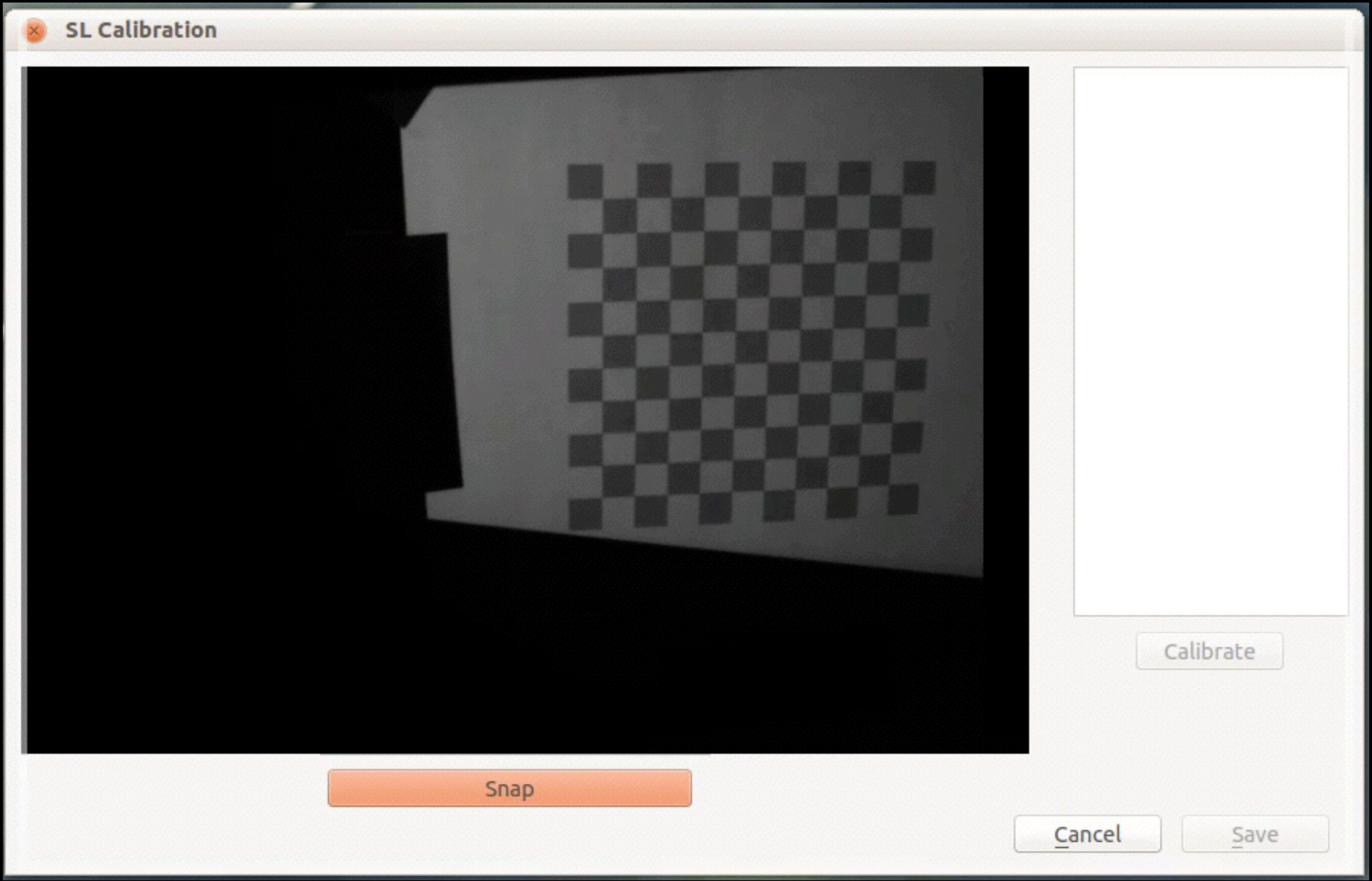
SLStudio



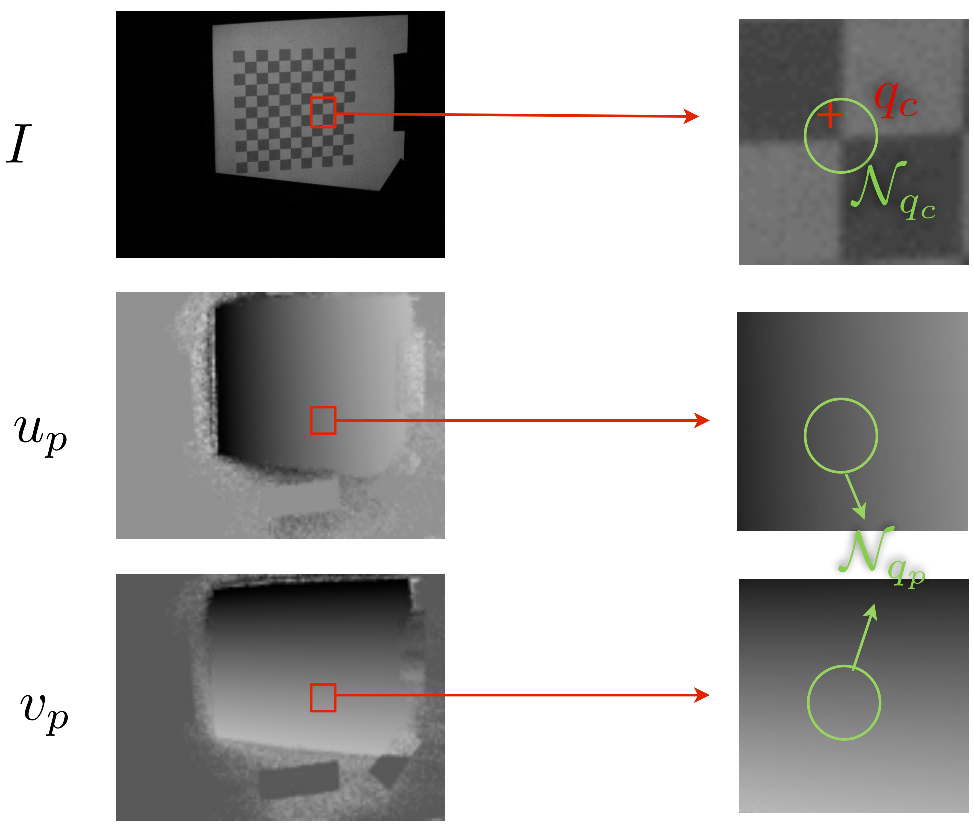
0, 2)	(0, 3)
1, 2)	(1, 3)
2, 2)	(2, 3)
3, 2)	(3, 3)

```
< mat.rows; i++){
; j < mat.cols; j++){
```

```
mm_loadu_si128(...);
mm_loadu_si128(...);
```


SL Calibration
Calibrate
Snap
Cancel
Save

Local Homographies

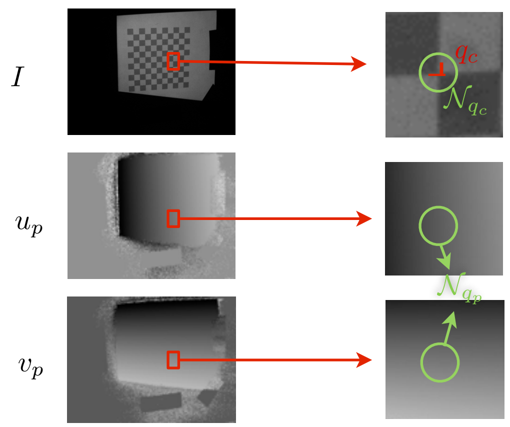


$$\hat{\boldsymbol{H}} = \arg\min_{\boldsymbol{H}} \|\boldsymbol{H} q_{ci} - q_{pi}\| \qquad \forall \qquad q_{ci} \in \mathcal{N}_{q_c}, q_{pi} \in \mathcal{N}_{q_p}$$

$$\hat{q_p} = \hat{\boldsymbol{H}} q_c$$

Moreno, Taubin (2012). 3DIMPVT. 2012(77)

Local Homographies



$$\hat{\boldsymbol{H}} = \arg\min_{\boldsymbol{H}} \|\boldsymbol{H} q_{ci} - q_{pi}\| \qquad \forall \qquad q_{ci} \in \mathcal{N}_{q_c}, q_{pi} \in \mathcal{N}_{q_p}$$

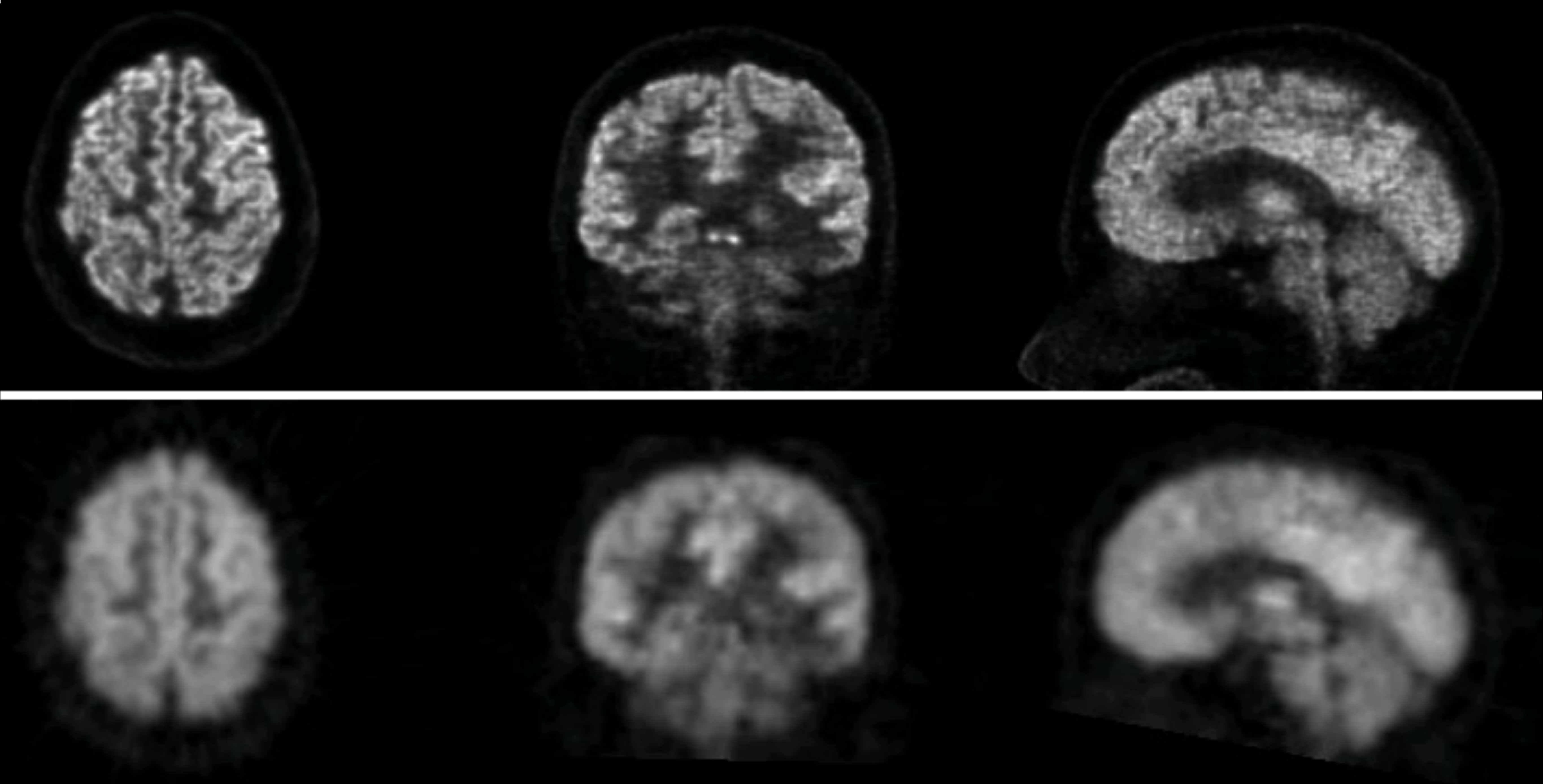
$$\hat{q}_p = \hat{\boldsymbol{H}} q_c$$

```
cv::Mat H = cv::findHomography(N_qc, N_qp, CV_LMEDS);
cv::Point3d Q = cv::Point3d(cv::Mat(H*cv::Mat(cv::Point3d(qc.x, qc.y, 1.0))));
cv::Point2f qp = cv::Point2f(Q.x/Q.z, Q.y/Q.z);

...

//calibrate the camera
double cam_error = cv::calibrateCamera(Q, qc, frameSize, Kc, kc, cam_rvecs, cam_tvecs, flags,
                                        cv::TermCriteria(cv::TermCriteria::COUNT + cv::TermCriteria::EPS, 50, DBL_EPSILON));

//calibrate the projector
double proj_error = cv::calibrateCamera(Q, qp, screenSize, Kp, kp, proj_rvecs, proj_tvecs, flags,
                                        cv::TermCriteria(cv::TermCriteria::COUNT + cv::TermCriteria::EPS, 50, DBL_EPSILON));

//stereo calibration
cv::Mat Rp, Tp, E, F;
double stereo_error = cv::stereoCalibrate(Q, qc, qp, Kc, kc, Kp, kp, frameSize, Rp, Tp, E, F,
                                          cv::TermCriteria(cv::TermCriteria::COUNT + cv::TermCriteria::EPS, 50, DBL_EPSILON),
                                          cv::CALIB_FIX_INTRINSIC);
```

Moreno, Taubin (2012). 3DIMPVT. 2012(77)

Computer Vision assisted Motion Correction in Medical Imaging

Photo courtesy of Oline V. Olesen, DTU

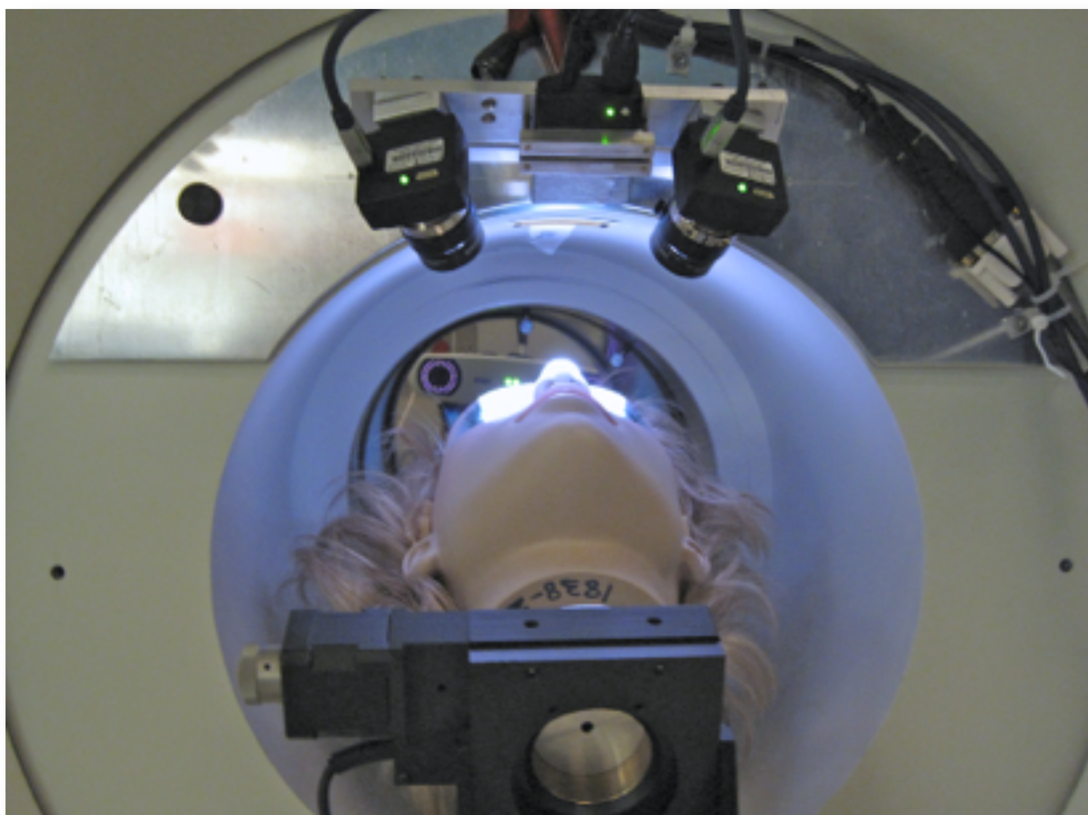


Courtesy of Oline V. Olesen, DTU

Motion Tracking

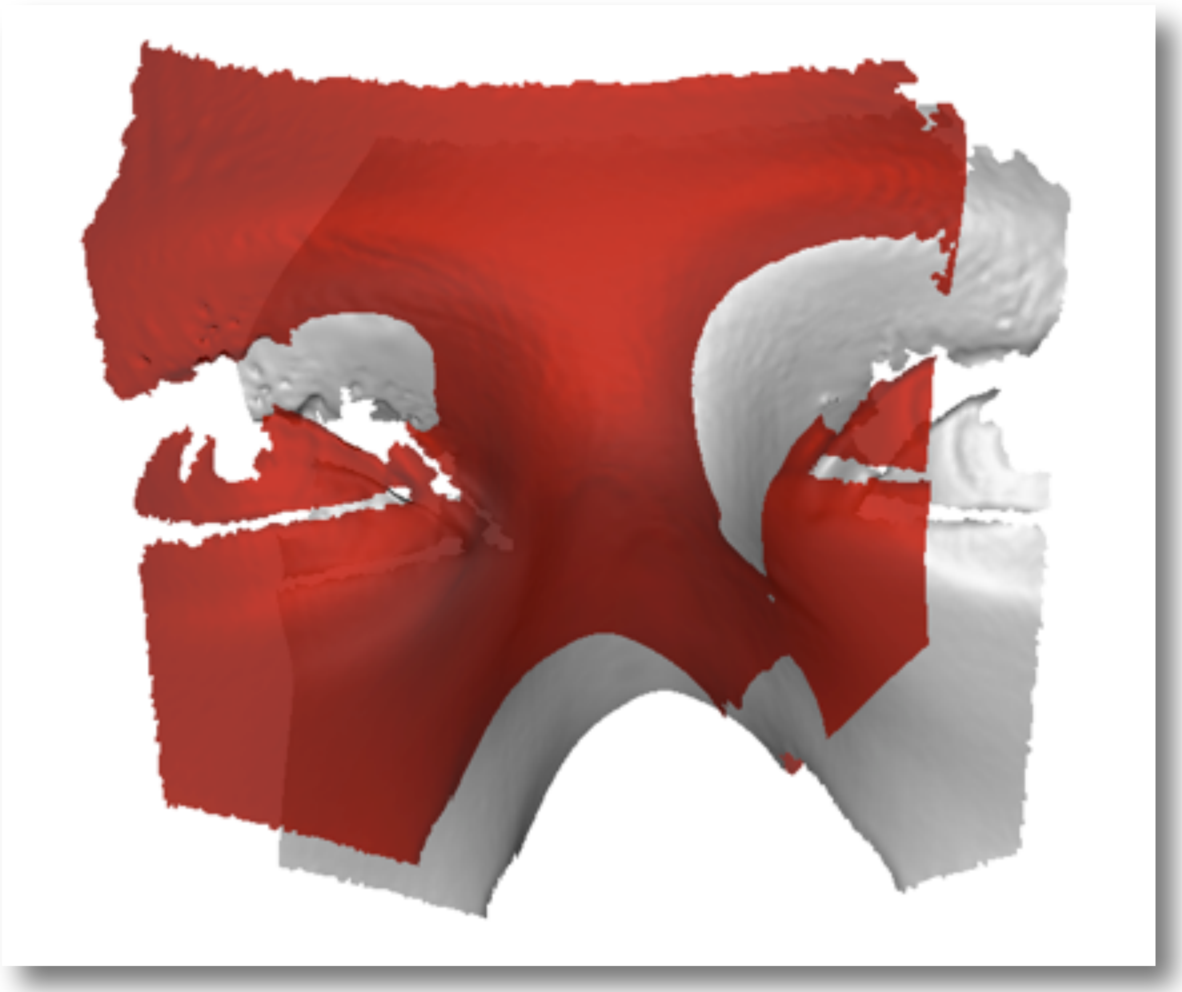
Courtesy of Oline V. Olesen, DTU

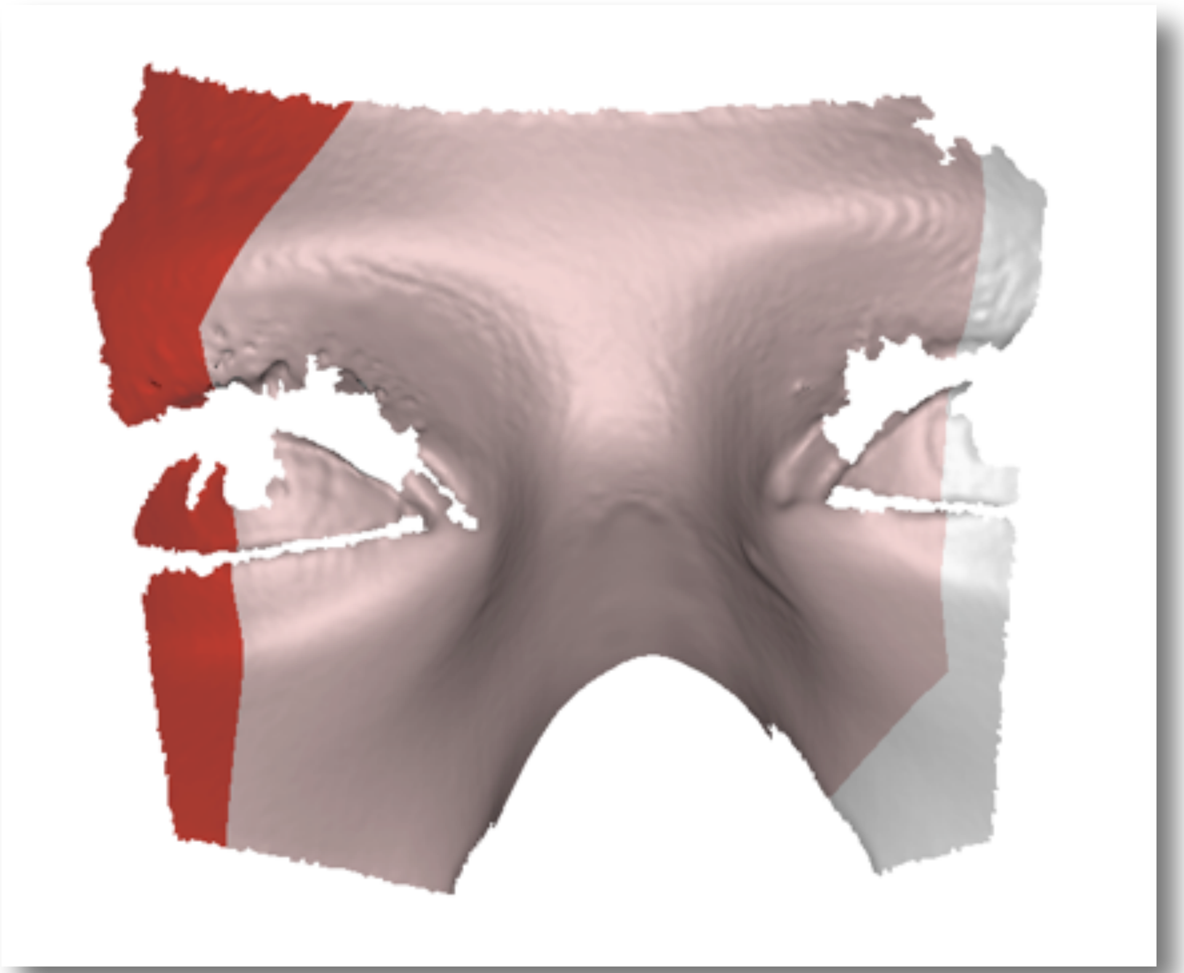
Pose Estimation



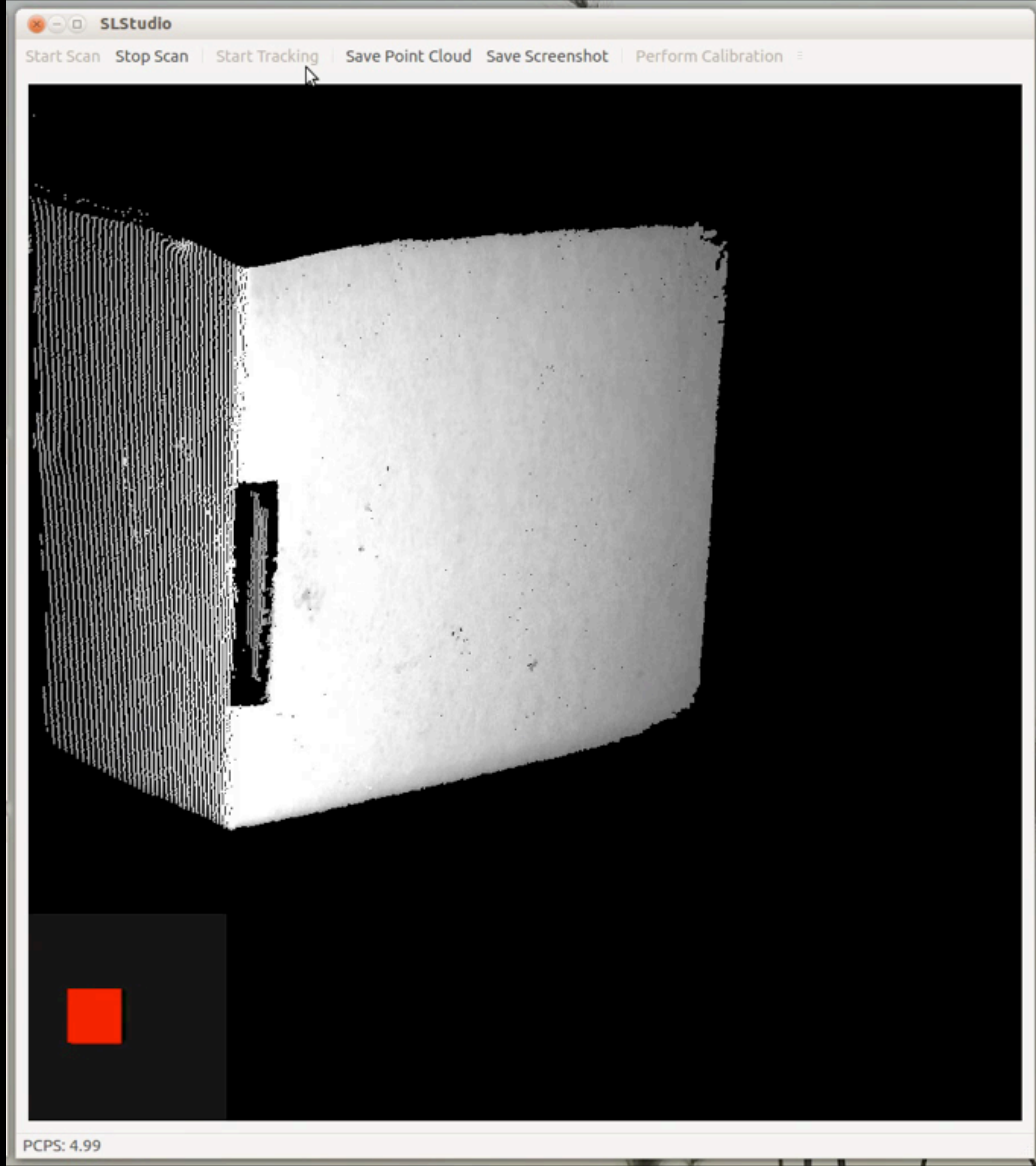
Courtesy of Oline V. Olesen, DTU

Pose Estimation





Wilm et al. (2011), Proc SCIA, LNCS 6688


SLStudio
Start Scan
Stop Scan
Start Tracking
Save Point Cloud
Save Screenshot
Perform Calibration
PCPS: 4.99

Fin