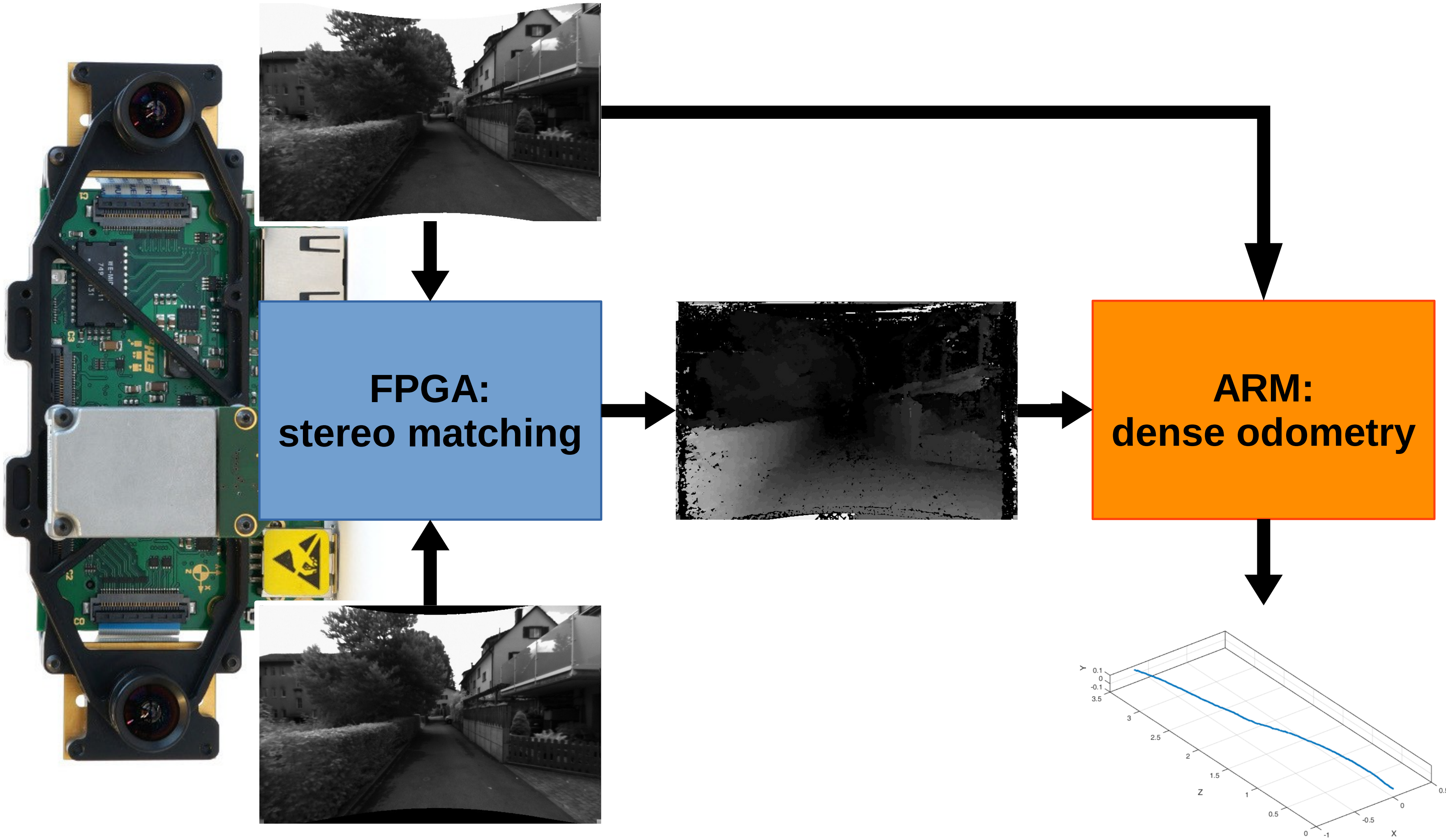
ETH zürich

# EMBEDDED REAL-TIME DENSE STEREOSCOPIC VISUAL ODOMETRY

**Samuel Bryner**

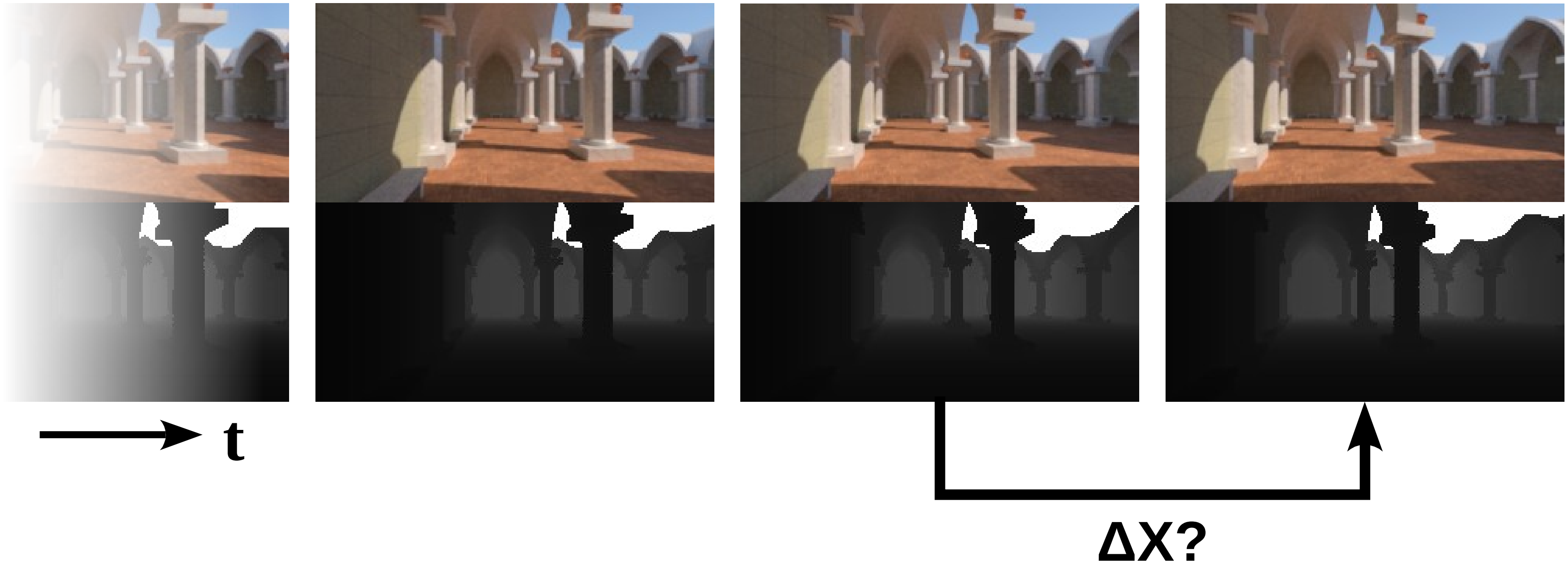
Bachelor Thesis
Supervised by Jörn Rehder and Pascal Gohl.


FPGA:
stereo matching
ARM:
dense odometry

# MOTIVATION

- integrate FPGA and ARM
- stereo on FPGA ⇒ embedded odometry

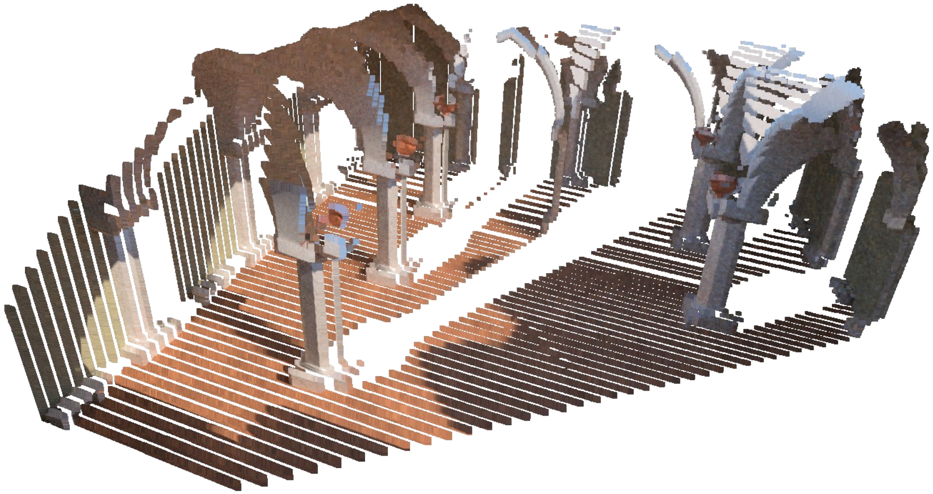
# METHOD: the problem



# project into space

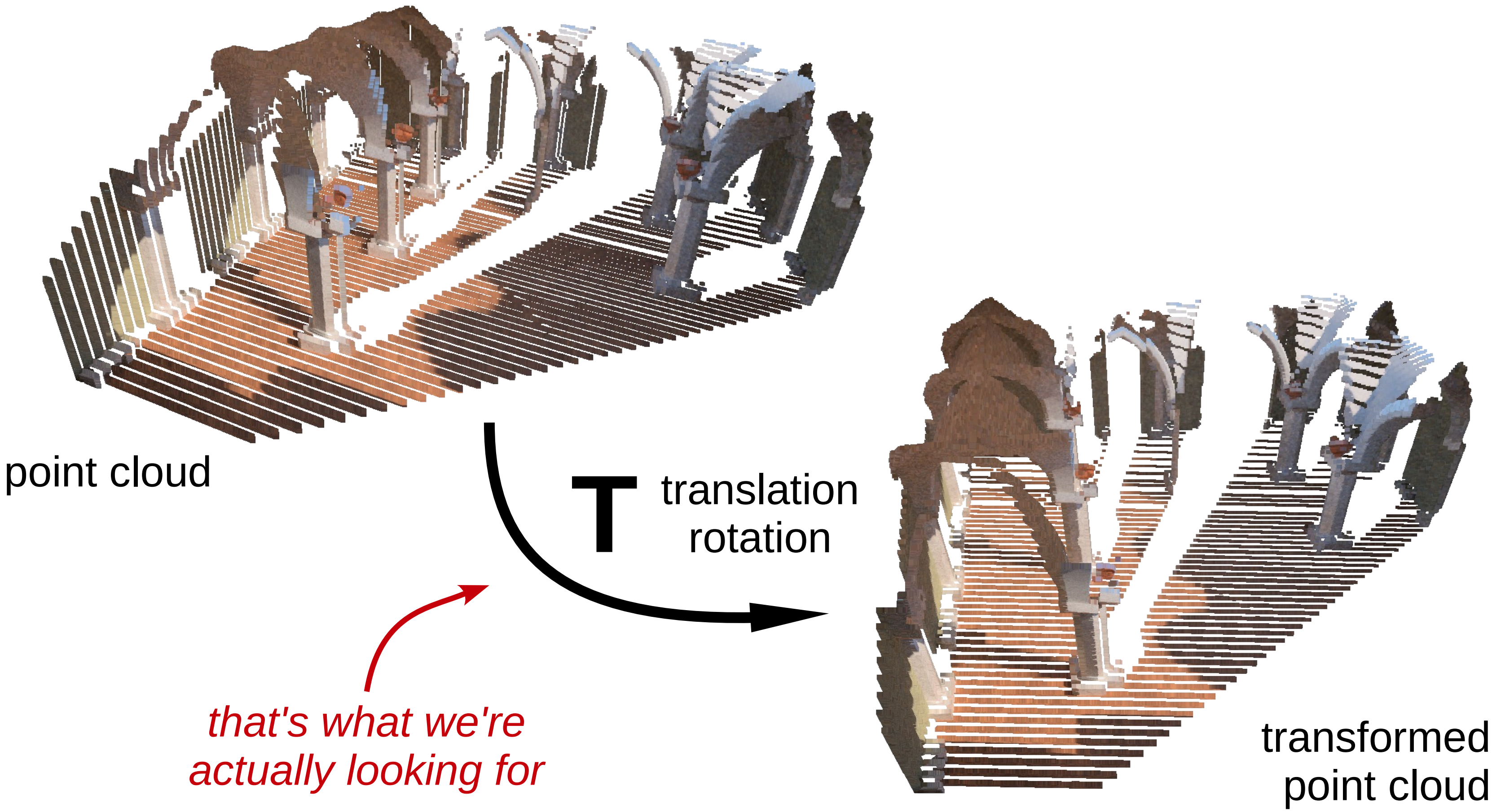

+


=

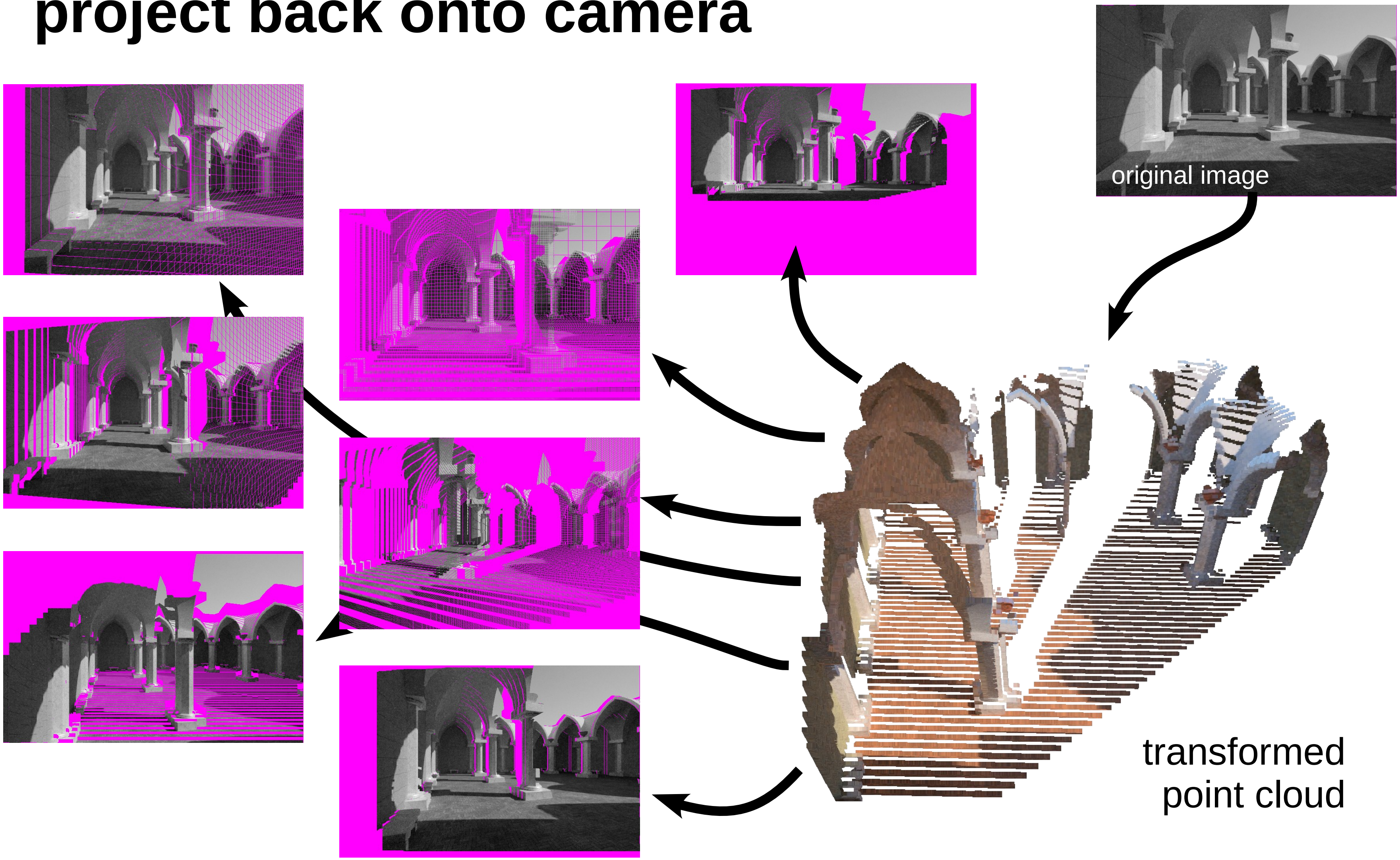


point cloud

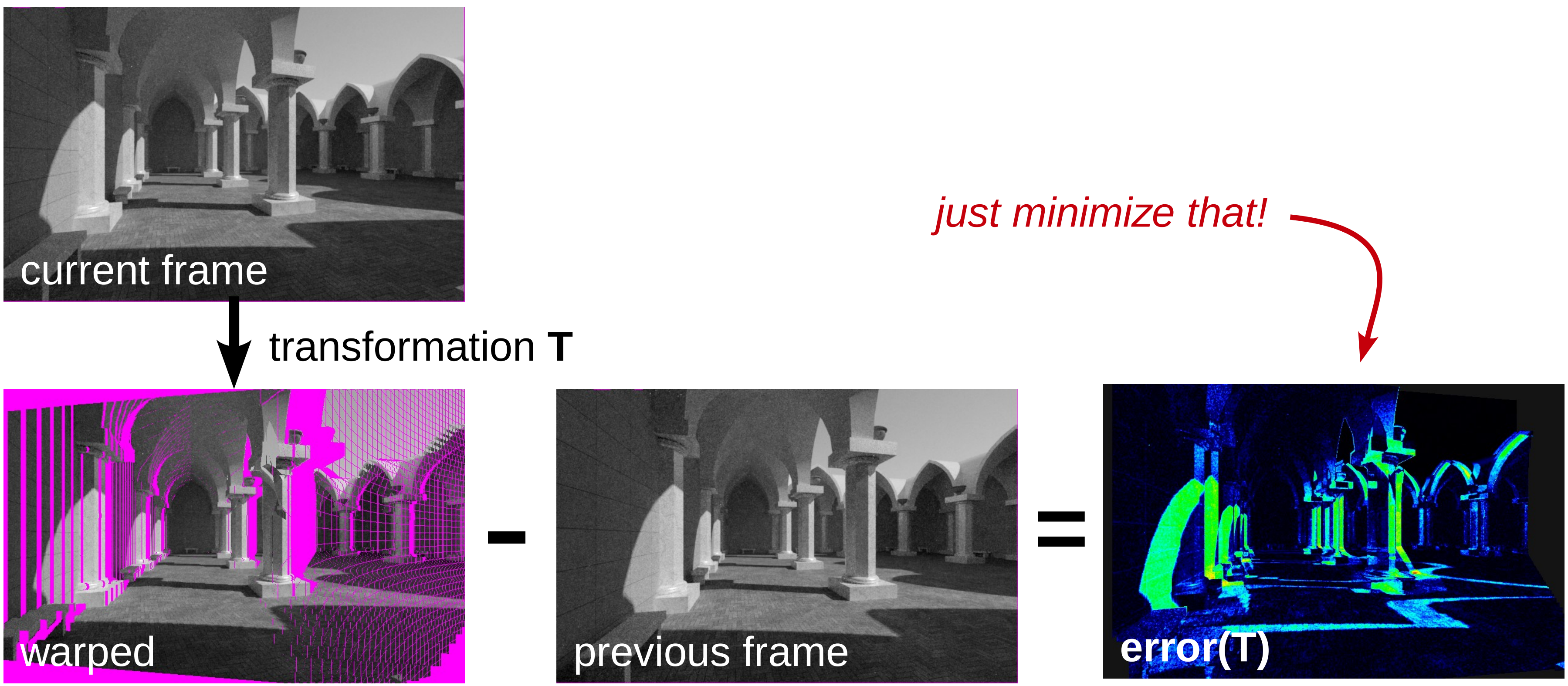
# transform trough space



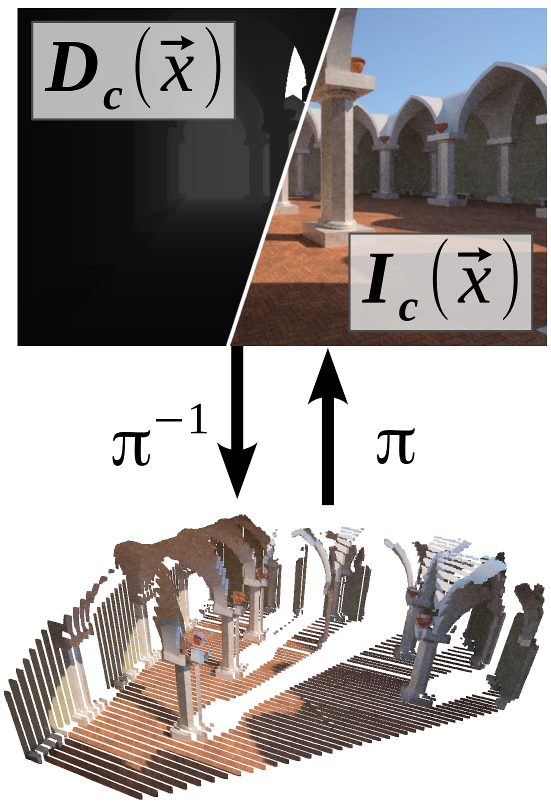
# project back onto camera



# but which of these is the best?



# but how? with some math!




point in image: $\vec{x} := (u, v) \in \mathbb{R}^2$

intensity: $\boldsymbol{I}(\vec{x}) \in \mathbb{R}$

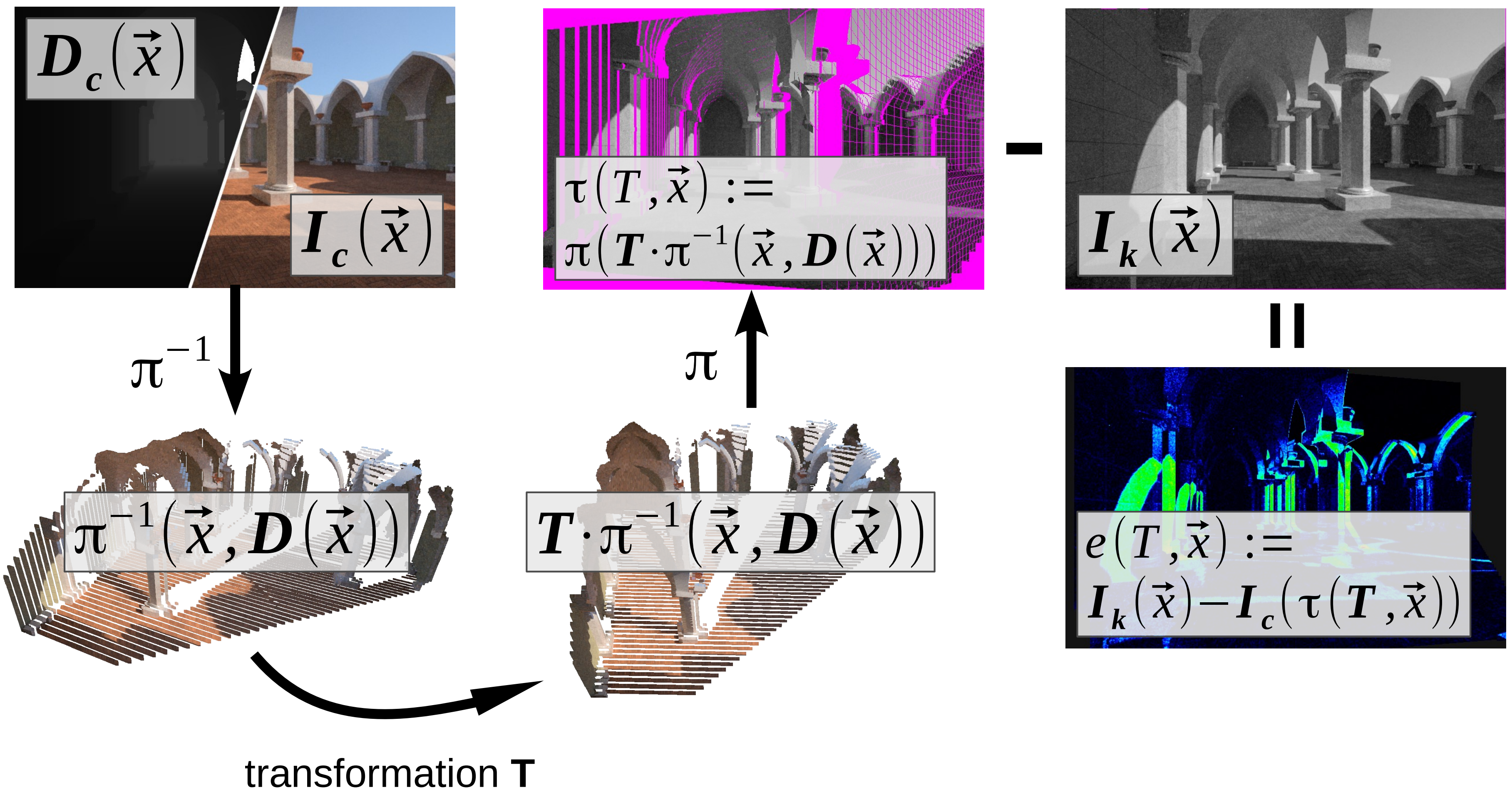
disparity: $\boldsymbol{D}(\vec{x}) \in \mathbb{R}$

transformation: $\boldsymbol{T} \in \mathbb{R}^6$

project into $\mathbb{R}^3$:

$$\pi^{-1}(\vec{x}, \boldsymbol{D}(\vec{x})) := \frac{b}{\boldsymbol{D}(\vec{x})} \begin{bmatrix} u - c_u \\ v - c_v \\ f \end{bmatrix}$$

# full warping



# therefore:

minimize

$$e(T,\vec{x}) := \boldsymbol{I}_{\boldsymbol{k}}(\vec{x}) - \boldsymbol{I}_{\boldsymbol{c}}(\tau(\boldsymbol{T},\vec{x}))$$

for every pixel:

$$\hat{\boldsymbol{T}} = \underset{\boldsymbol{T}}{argmin} \sum_{\vec{x} \in \boldsymbol{I}_k} e(T,\vec{x})^2$$

using Gauss Newton:

$$\boldsymbol{J}^T \boldsymbol{J} \Delta T = -\boldsymbol{J} e(\boldsymbol{T})$$

*TODO: explain derivation of Jacobians?*

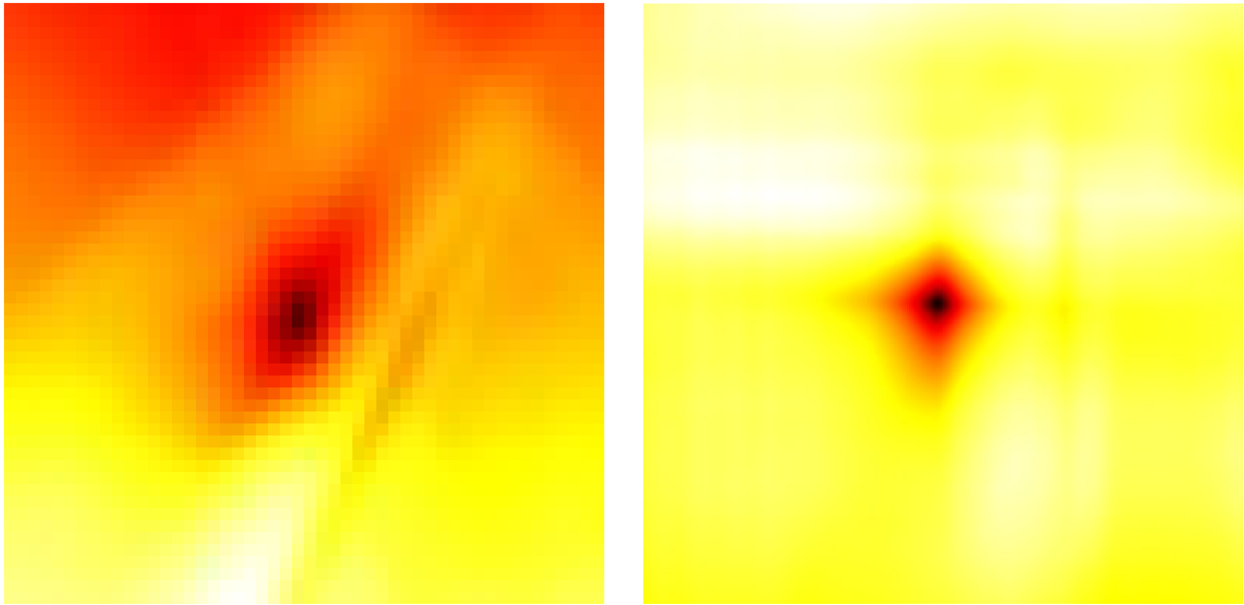
# TODO: Optimizations

*TODO: are these necessary?*

- use image pyramid
  - full resolution isn't really necessary
- only use pixels with strong gradient
- Huber weights?
- Levenberg-Marquardt?

*TODO: provide data on how much a difference these things make*

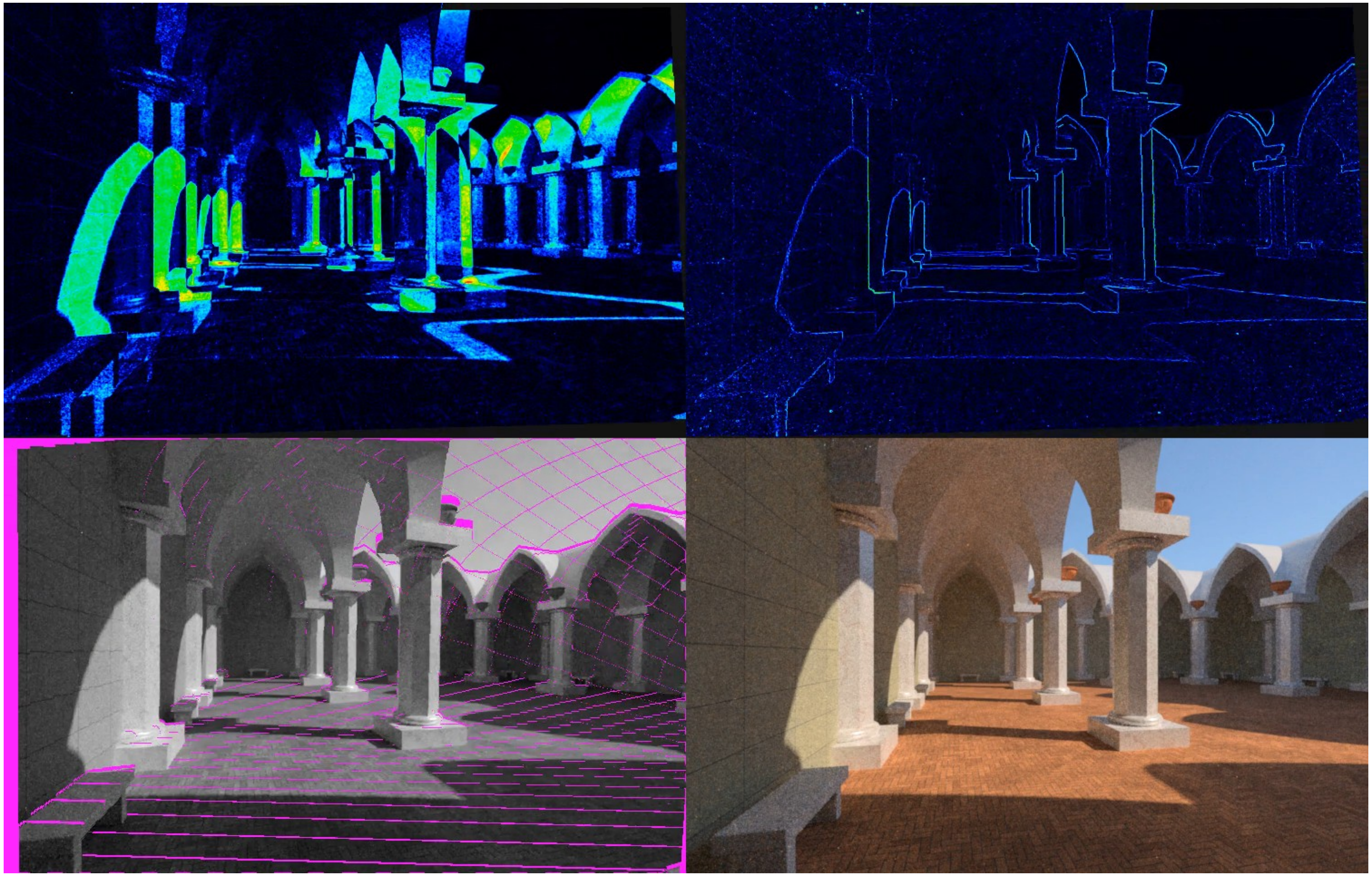
# TODO: COST SURFACE



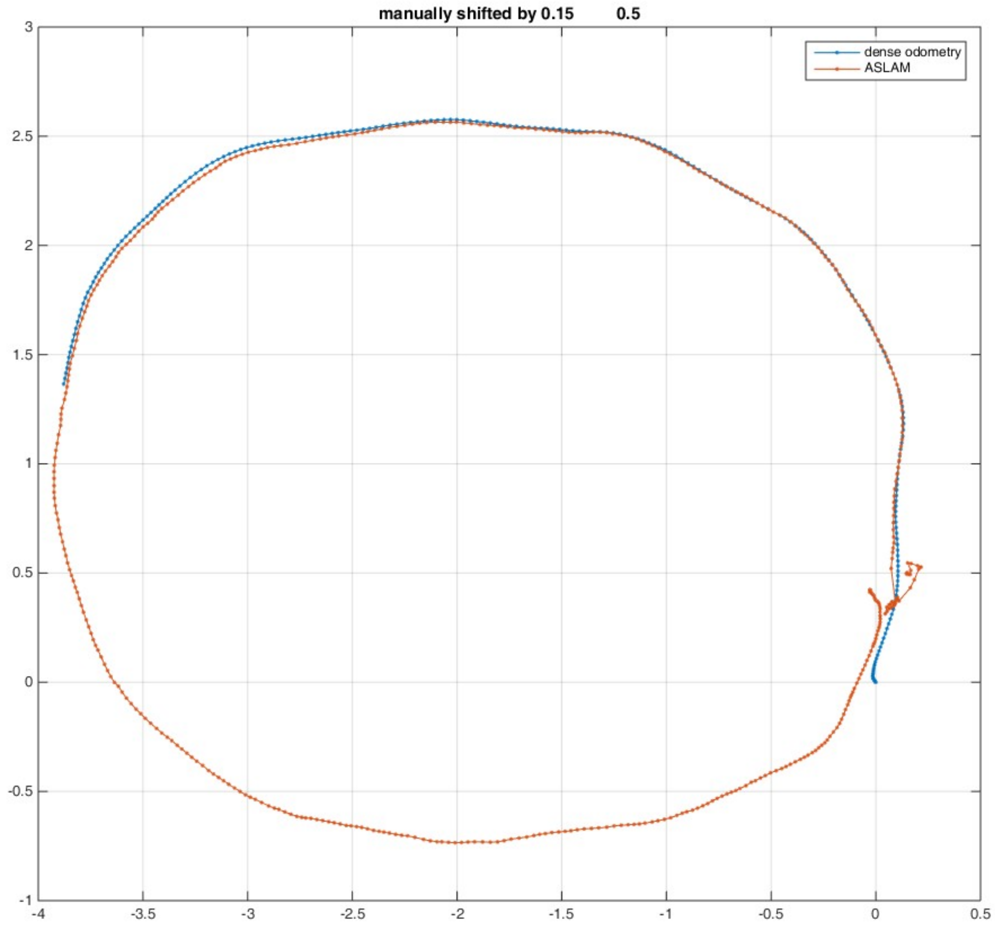
*TODO: add better error plots here (with labeled axes and stuff)*

# Gauss-Newton In Action

*TODO: make that thing fullscreen and click-to-play*



# TODO: RESULTS




*TODO: use full trajectory here and bigger plot*

# TODO: MORE RESULTS

- simulated circular trajectory
- timing data
- comparison of various parameters
  - does pyramid help with convergence radius or just with speed?
  - precision loss by not using full image resolution or filtering out 'boring' pixels

# TODO: CONCLUSION