Efficient probabilistic planar robot motion estimation given pairs of images

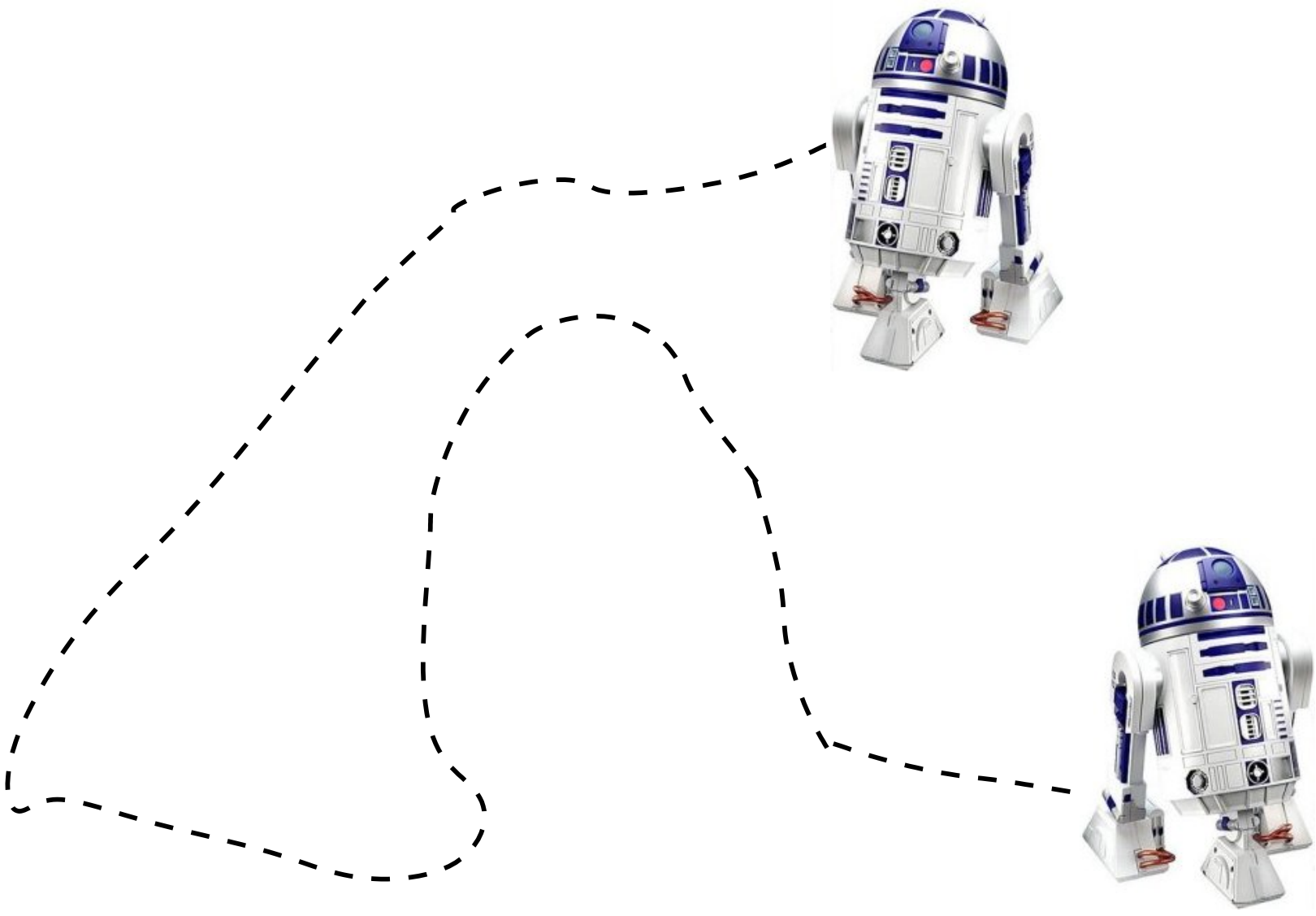
Olaf Booij, Zoran Zivkovic, Ben Kröse

Intelligent Systems Lab Amsterdam
University of Amsterdam, The Netherlands

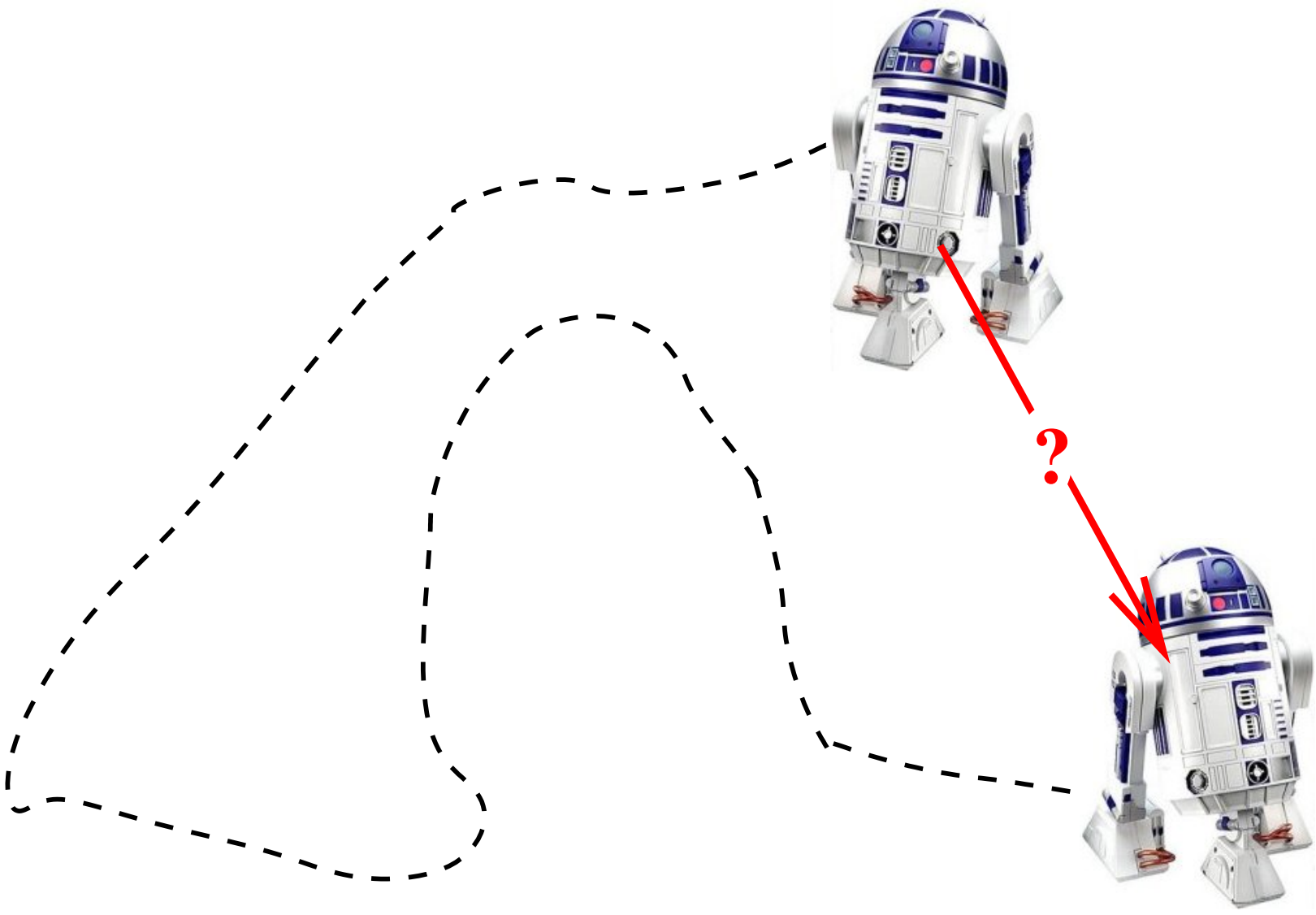
RSS 29-6-2010

Context

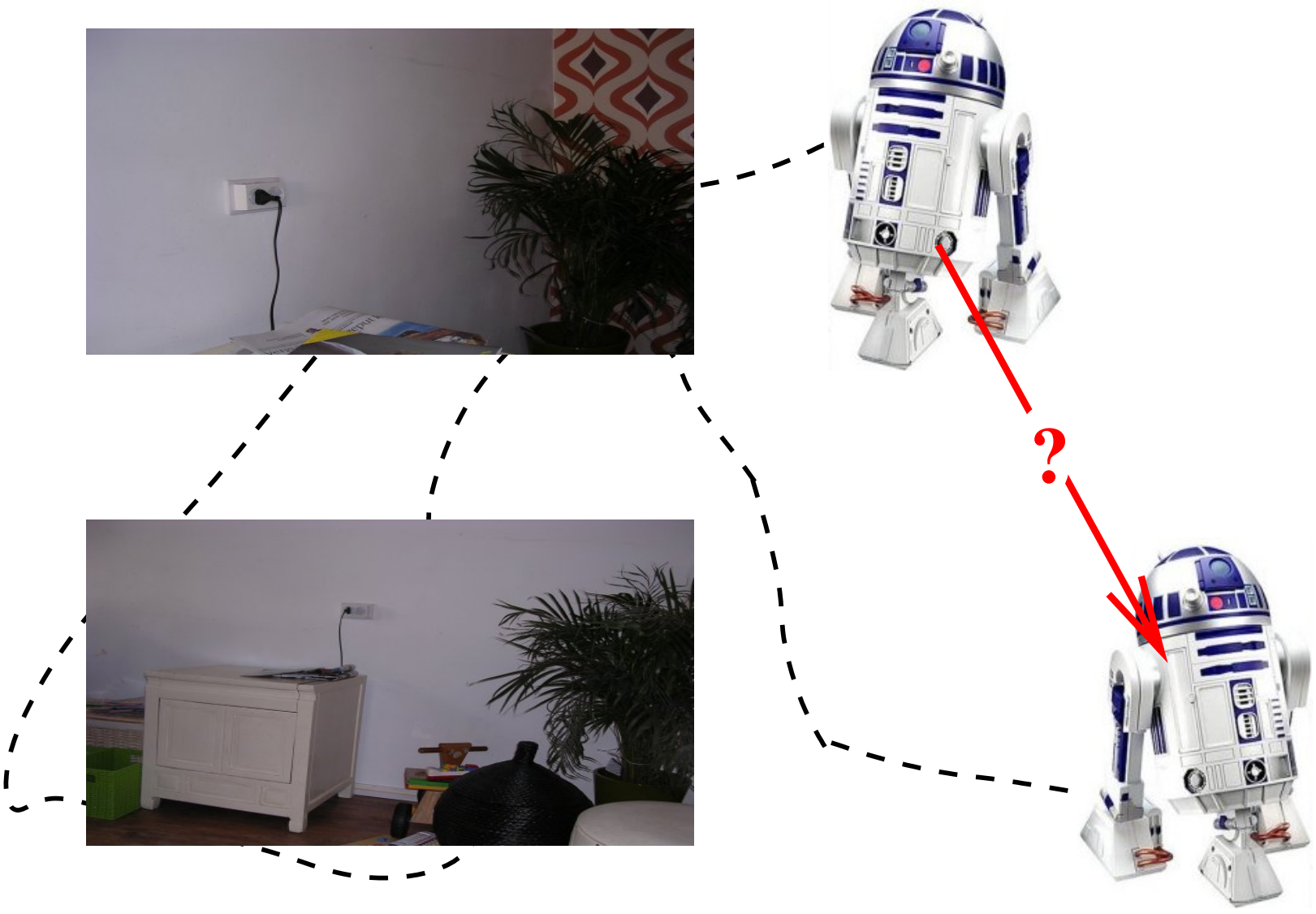
Context



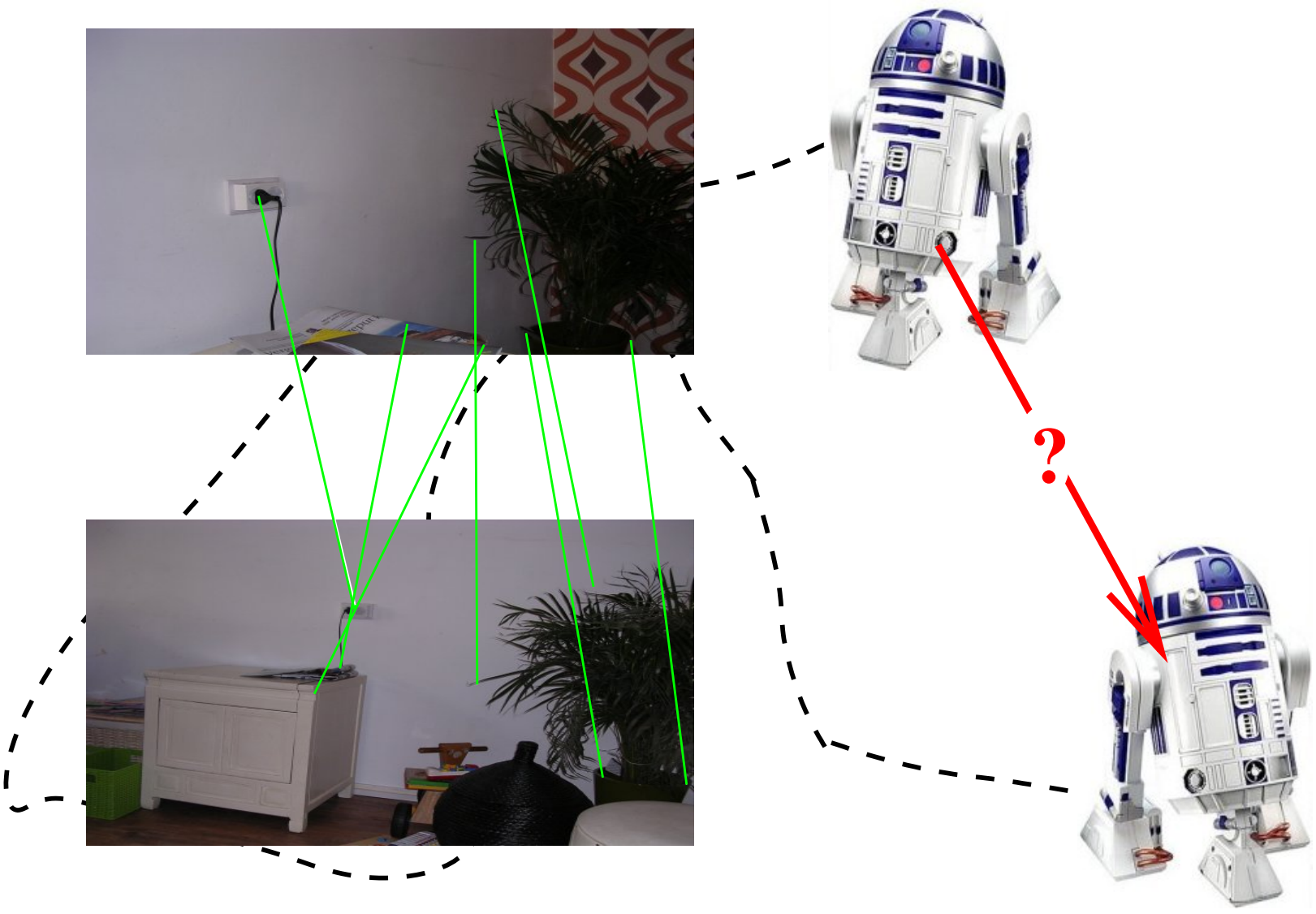
Context



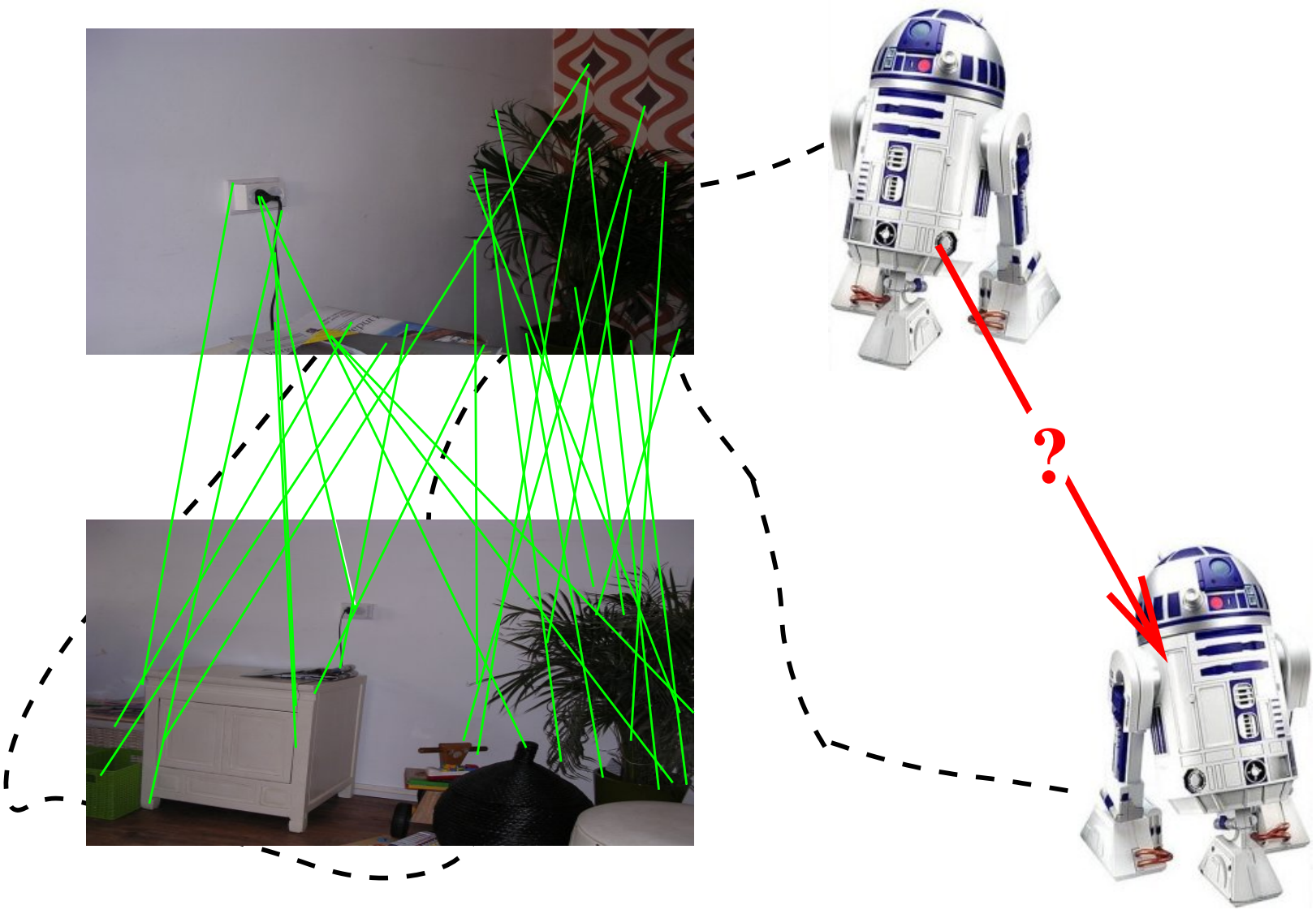
Context



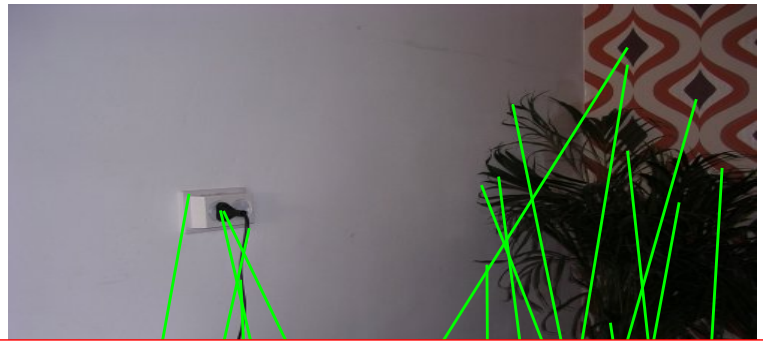
Context



Context



Context



How to compute the pose likelihood

$p(\text{correspondence} \mid \text{pose})$

Outline

Outline

Related work: solving $p(\text{correspondence} \mid \text{pose})$

Relation $\overline{\text{correspondence}}$, $\overline{\text{pose}}$

- non-linear 2 correspondences
- linear for >=3 correspondences

Observation error

- gaussian noise for correct matches (inliers)
- uniform noise for mismatches (outliers)

Find best fit

- RANSAC et al. + 3 or 2 point estimator
- M-Estimators/Bundle adjustment/EM
- -> ML solution (+ local uncertainty)

R. Hartley and A. Zisserman "Multiple view geometry in computer vision"
P. H. S. Torr and A. Zisserman "MLESAC: a new robust estimator with application to estimating image geometry"
D. Ortín and J. M. M. Montiel "Indoor robot motion based on monocular images"

Related work: solving $p(\text{correspondence} \mid \text{pose})$

Relation $\overline{\text{correspondence}}$, $\overline{\text{pose}}$

- non-linear 2 correspondences
- linear for >=3 correspondences

Observation error

- gaussian noise for correct matches (inliers)
- uniform noise for mismatches (outliers)

Find best fit

- RANSAC et al. + 3 or 2 point estimator
- M-Estimators/Bundle adjustment/EM
- -> ML solution (+ local uncertainty)

R. Hartley and A. Zisserman "Multiple view geometry in computer vision"
P. H. S. Torr and A. Zisserman "MLESAC: a new robust estimator with application to estimating image geometry"

D. Ortín and J. M. M. Montiel "Indoor robot motion based on monocular images"

Related work: solving $p(\text{correspondence} \mid \text{pose})$

Relation $\overline{\text{correspondence}}$, $\overline{\text{pose}}$

- non-linear 2 correspondences
- linear for >=3 correspondences

Observation error

- gaussian noise for correct matches (inliers)
- uniform noise for mismatches (outliers)

Find best fit

- RANSAC et al. + 3 or 2 point estimator
- M-Estimators/Bundle adjustment/EM
- -> ML solution (+ local uncertainty)

R. Hartley and A. Zisserman "Multiple view geometry in computer vision"
P. H. S. Torr and A. Zisserman "MLESAC: a new robust estimator with application to estimating image geometry"

D. Ortín and J. M. M. Montiel "Indoor robot motion based on monocular images"

Related work: problems

Inlier error is not gaussian

- modeling error
- calibration error
- discretization error
- non-planarity error
- scene dependent

Outlier error is not uniform

- SIFT +/ 30 degrees viewangle difference
- scene dependent (floor featureless?)

Solution is not gaussian

- multiple modes
- degenerate cases (unobservability)

Related work: problems

Inlier error is not gaussian

- modeling error
- calibration error
- discretization error
- non-planarity error
- scene dependent

Outlier error is not uniform

- SIFT +/ 30 degrees viewangle difference
- scene dependent (floor featureless?)

Solution is not gaussian

- multiple modes
- degenerate cases (unobservability)

Related work: problems

Inlier error is not gaussian

- modeling error
- calibration error
- discretization error
- non-planarity error
- scene dependent

Outlier error is not uniform

- SIFT +/ 30 degrees viewangle difference
- scene dependent (floor featureless?)

Solution is not gaussian

- multiple modes
- degenerate cases (unobservability)

Outline

Proposed method: overview

General idea

- model $p(\text{correspondence} \mid \text{pose})$ using non-parametric model
- discretize both correspondence and pose
- create look-up-table for all correspondence-pose combinations
- use existing data (learning!)

Problems to overcome

- look-up-table should be low dimensional
- i.e. bin-size should be small

Proposed method: overview

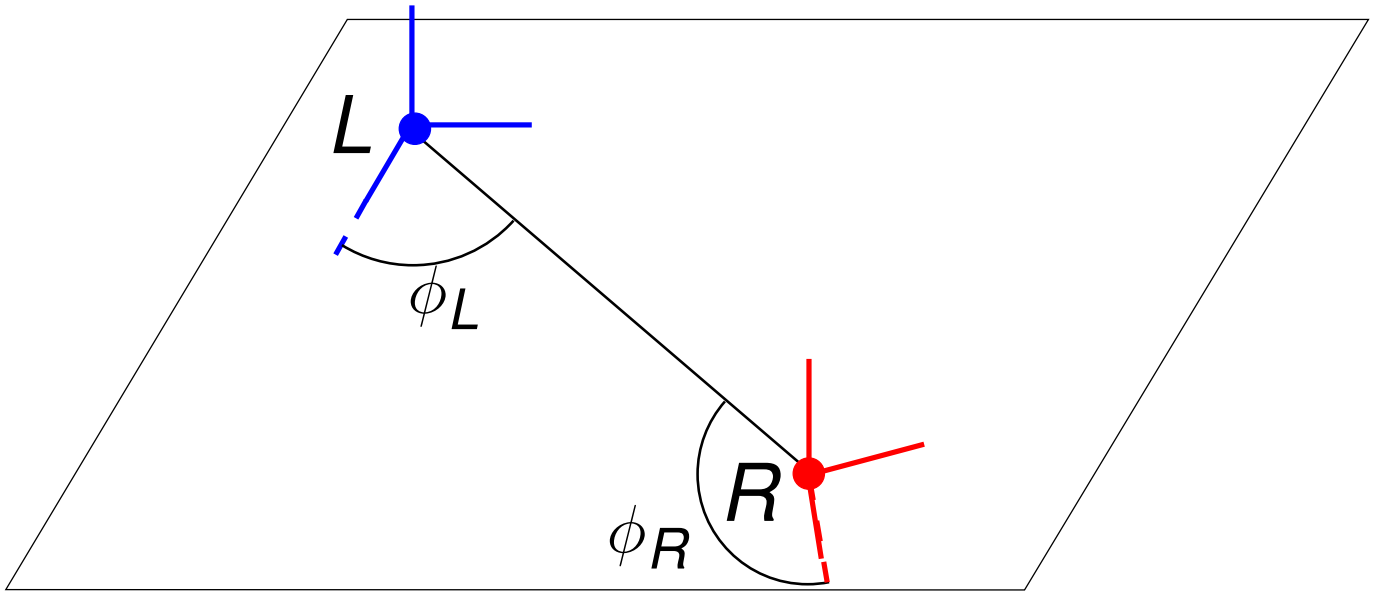
General idea

- model $p(\text{correspondence} \mid \text{pose})$ using non-parametric model
- discretize both correspondence and pose
- create look-up-table for all correspondence-pose combinations
- use existing data (learning!)

Problems to overcome

- look-up-table should be low dimensional
- i.e. bin-size should be small

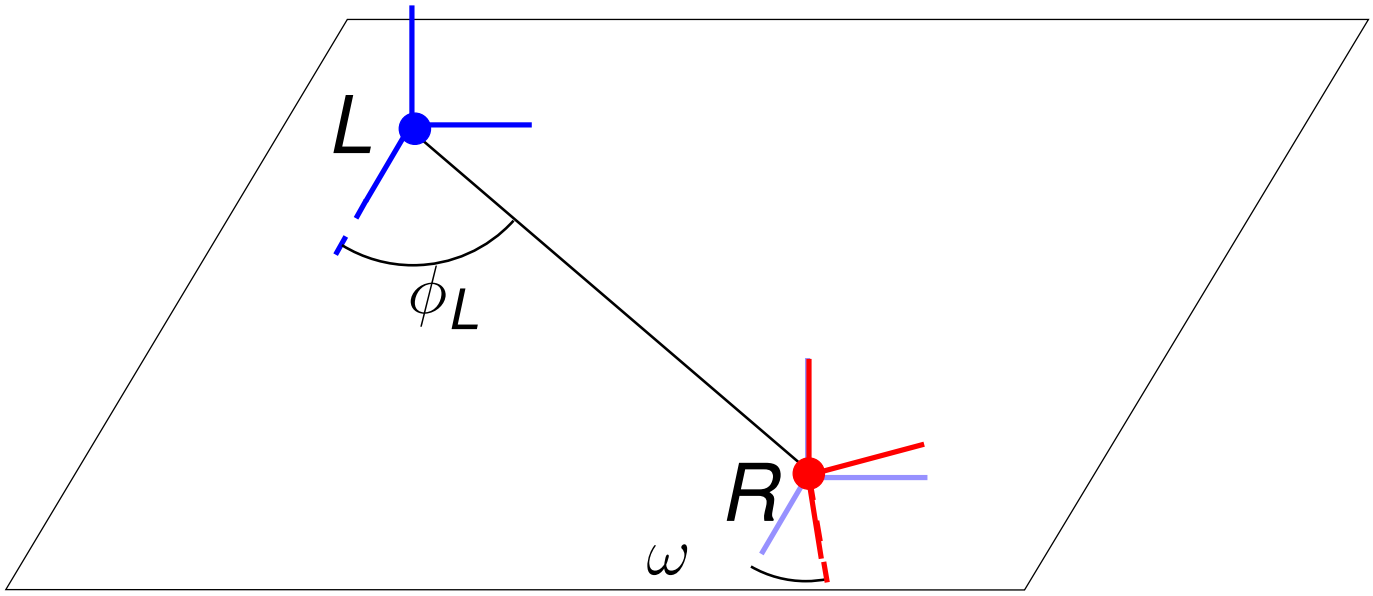
Proposed method: parameterization



Planar pose parameterization

- 2 angles are enough
- Note: scale is not parameterized
- Alternative representation using ω ($\omega = \pi + \phi_L - \phi_R$)

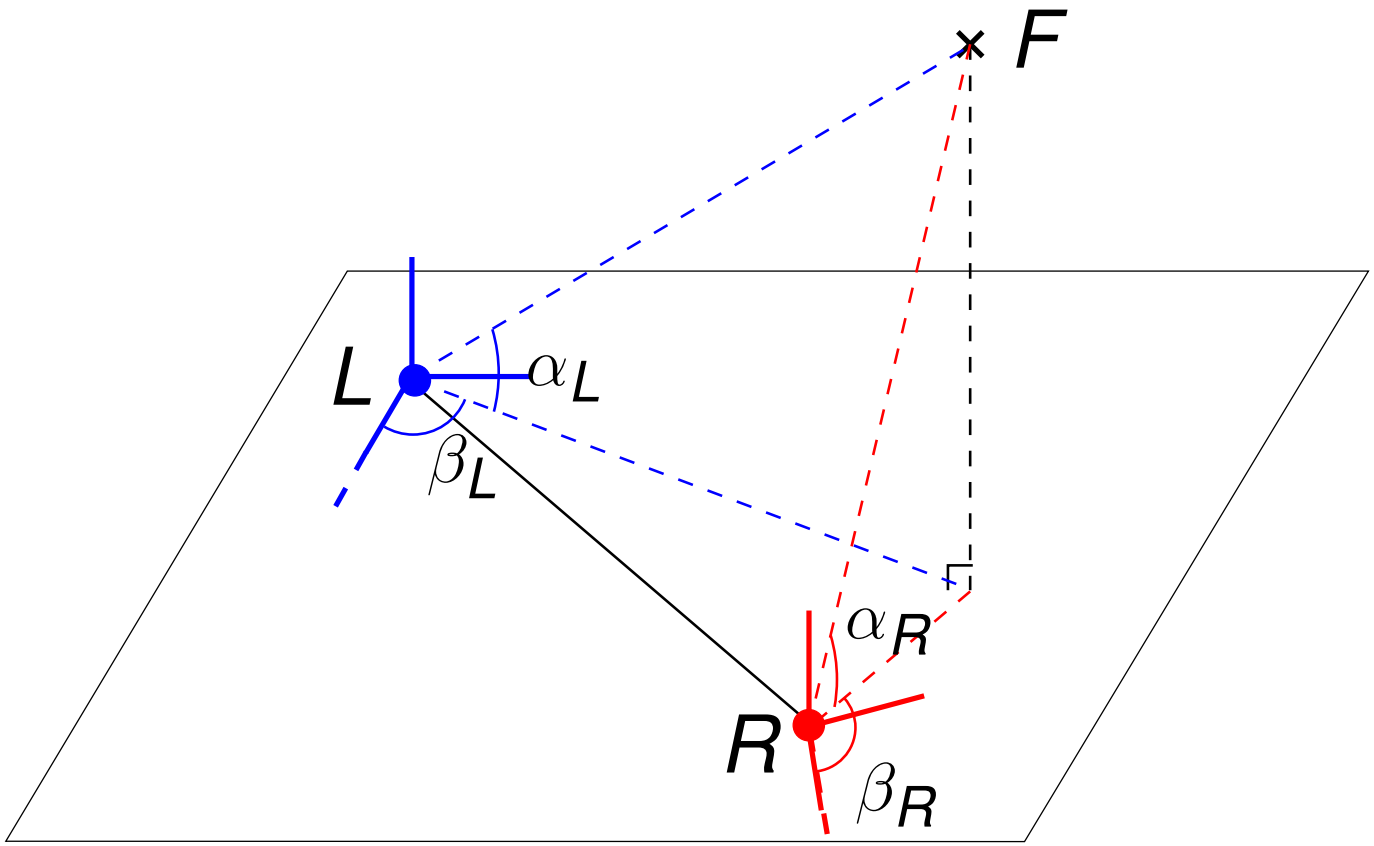
Proposed method: parameterization



Planar pose parameterization

- 2 angles are enough
- Note: scale is not parameterized
- Alternative representation using ω ($\omega = \pi + \phi_L - \phi_R$)

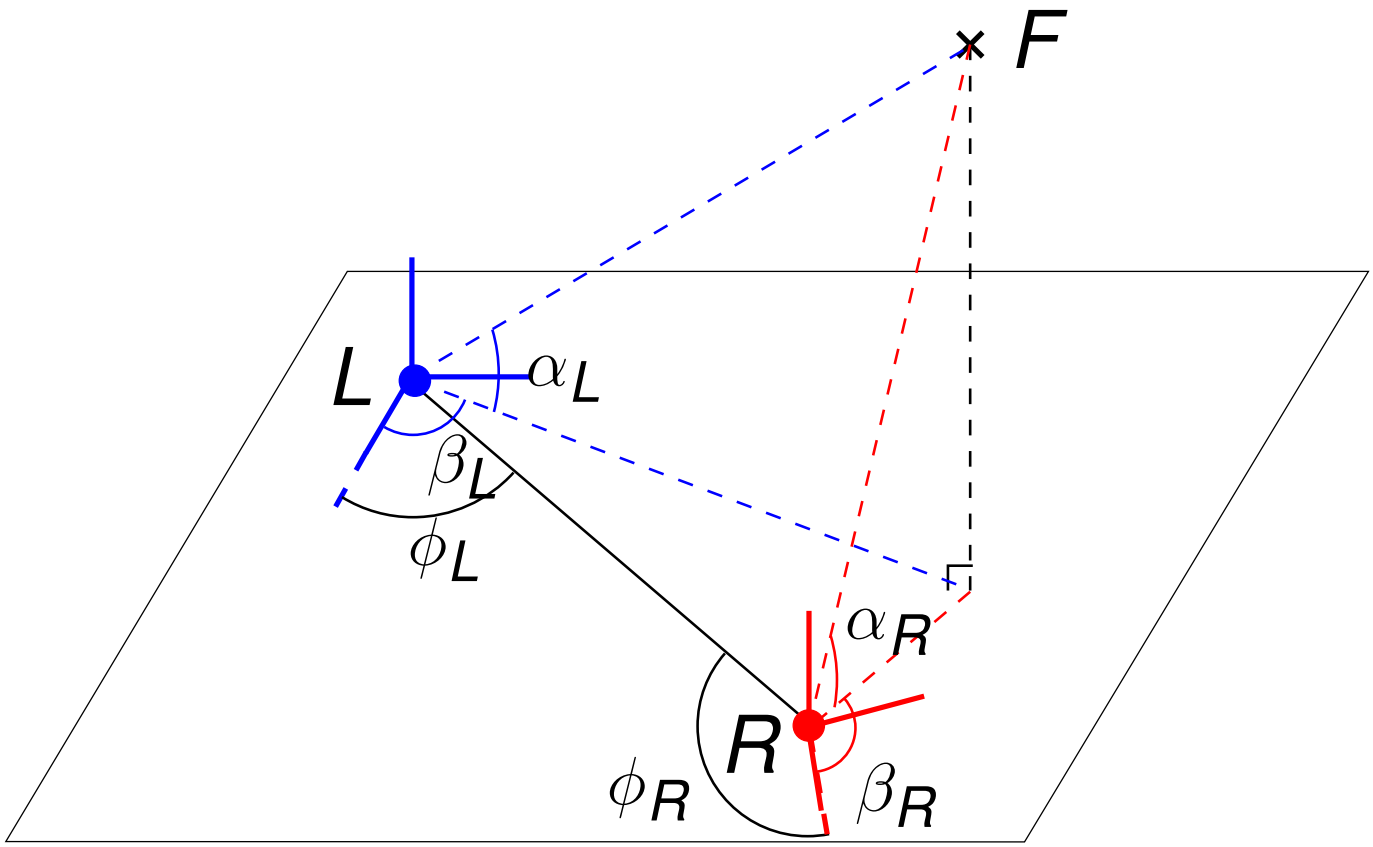
Proposed method



Correspondence parameterization (perspective case)

- Vertical angle $\alpha = \arctan(y)$
- Horizontal angle $\beta = \arctan(x)$

Proposed method



Function relating correspondence to pose

$$\phi_R - \beta_R = \arcsin\left(\frac{\tan(\alpha_R)}{\tan(\alpha_L)}\sin(\beta_L - \phi_L)\right).$$

Proposed method: 3d LUT

- the likelihood thus involves 6 parameters:

$$p(\alpha_L, \beta_L, \alpha_R, \beta_R | \phi_L, \phi_R)$$

- which would result in a 6 dimensional LUT
- using the correspondence-pose relation, we can approximate it:

$$\phi_R - \beta_R = \arcsin\left(\frac{\tan(\alpha_R)}{\tan(\alpha_L)} \sin(\beta_L - \phi_L)\right).$$

$$p\left(\frac{\tan(\alpha_R)}{\tan(\alpha_L)}, \phi_L - \beta_L, \phi_R - \beta_R \middle| \phi_L \quad , \phi_R \quad\right)$$

$$\propto p\left(\frac{\tan(\alpha_R)}{\tan(\alpha_L)}, \phi_L - \beta_L, \phi_R - \beta_R\right)$$

Proposed method: 3d LUT

- the likelihood thus involves 6 parameters:

$$p(\alpha_L, \beta_L, \alpha_R, \beta_R | \phi_L, \phi_R)$$

- which would result in a 6 dimensional LUT
- using the correspondence-pose relation, we can approximate it:

$$\phi_R - \beta_R = \arcsin\left(\frac{\tan(\alpha_R)}{\tan(\alpha_L)} \sin(\beta_L - \phi_L)\right).$$

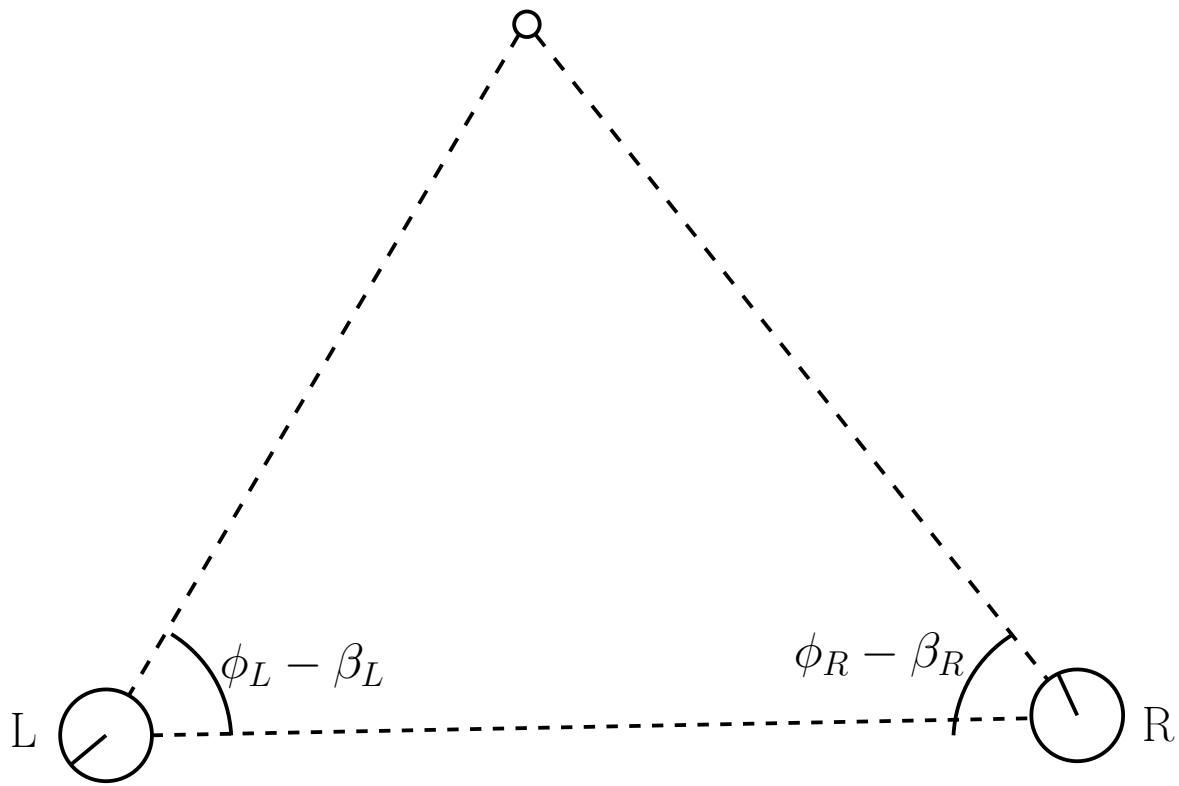
$$p\left(\frac{\tan(\alpha_R)}{\tan(\alpha_L)}, \phi_L - \beta_L, \phi_R - \beta_R \middle| \phi_L^{\text{uni}}, \phi_R^{\text{uni}}\right)$$

$$\propto p\left(\frac{\tan(\alpha_R)}{\tan(\alpha_L)}, \phi_L - \beta_L, \phi_R - \beta_R\right)$$

Proposed method: 3d LUT

So, what are we assuming?

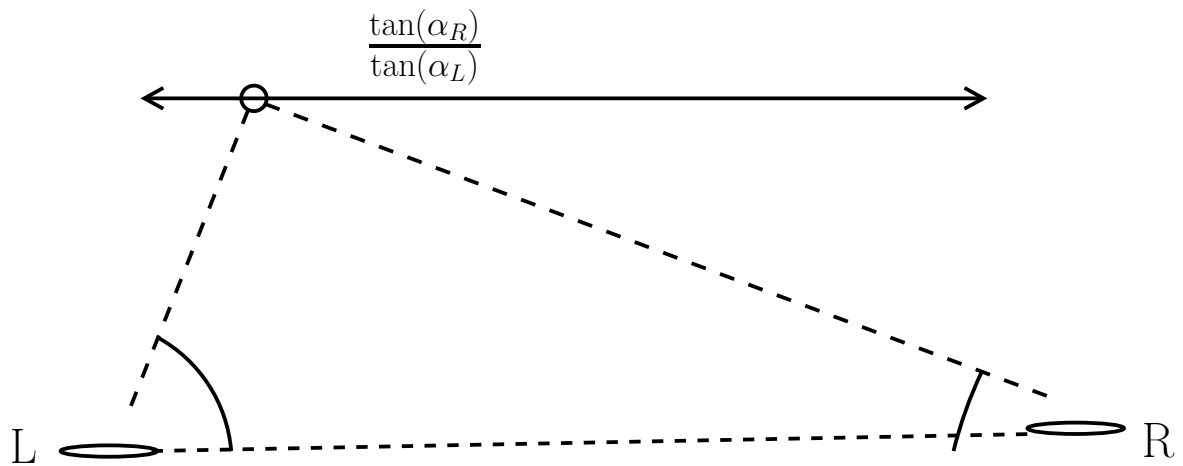
- $\phi - \beta$ models the horizontal angle with respect to the heading
- $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ models ratio of the vertical angle.



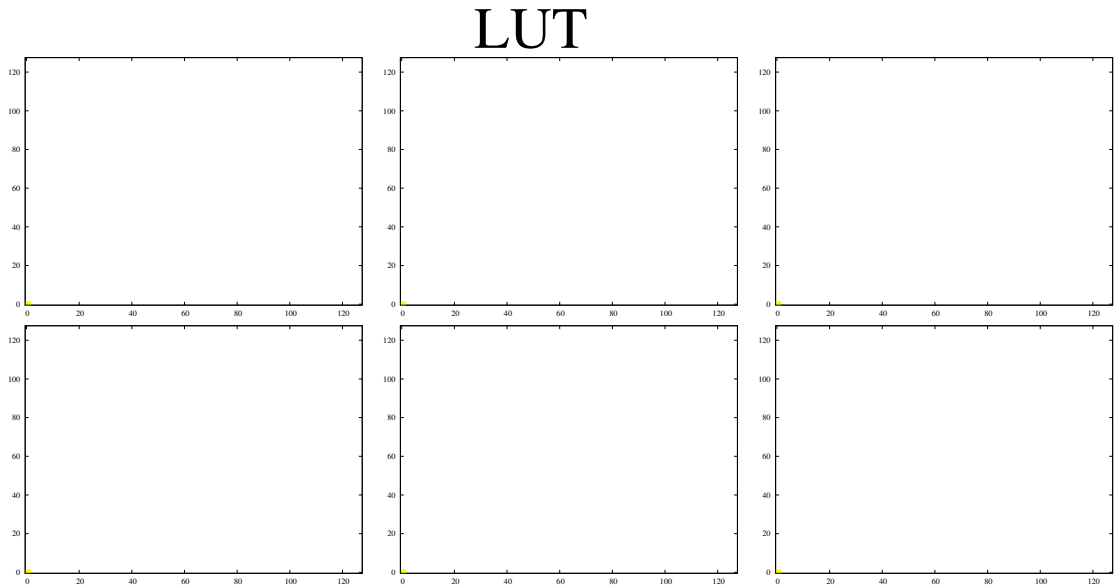
Proposed method: 3d LUT

So, what are we assuming?

- $\phi - \beta$ models the horizontal angle with respect to the heading
- $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ models ratio of the vertical angle.

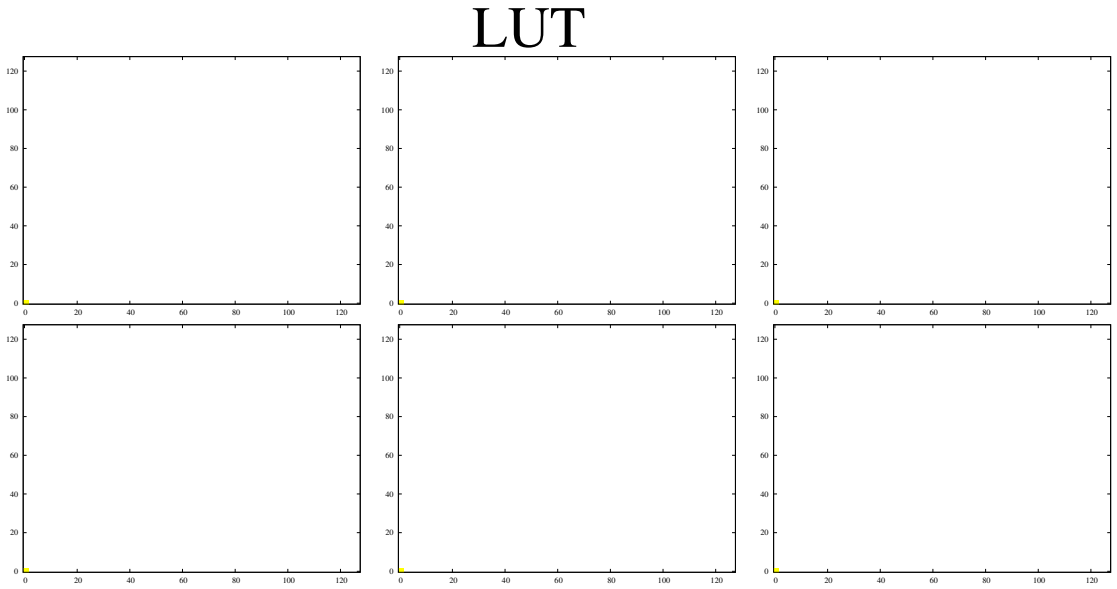


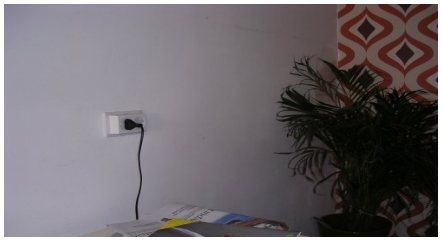
Proposed method: Building a LUT

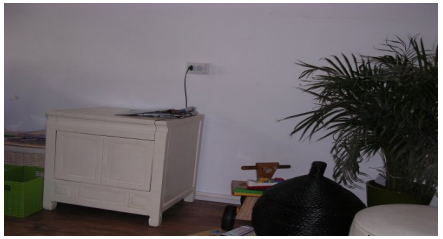


- For each image with ground truth pose
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT-slice.
- Compute $\phi_L - \beta_L$ and $\phi_R - \beta_R$ and add to the corresponding bin to pose prior $p(\phi_L, \phi_R)$
- Do not treat mismatches separately
- Finally normalize and log each slice

Proposed method: Building a LUT



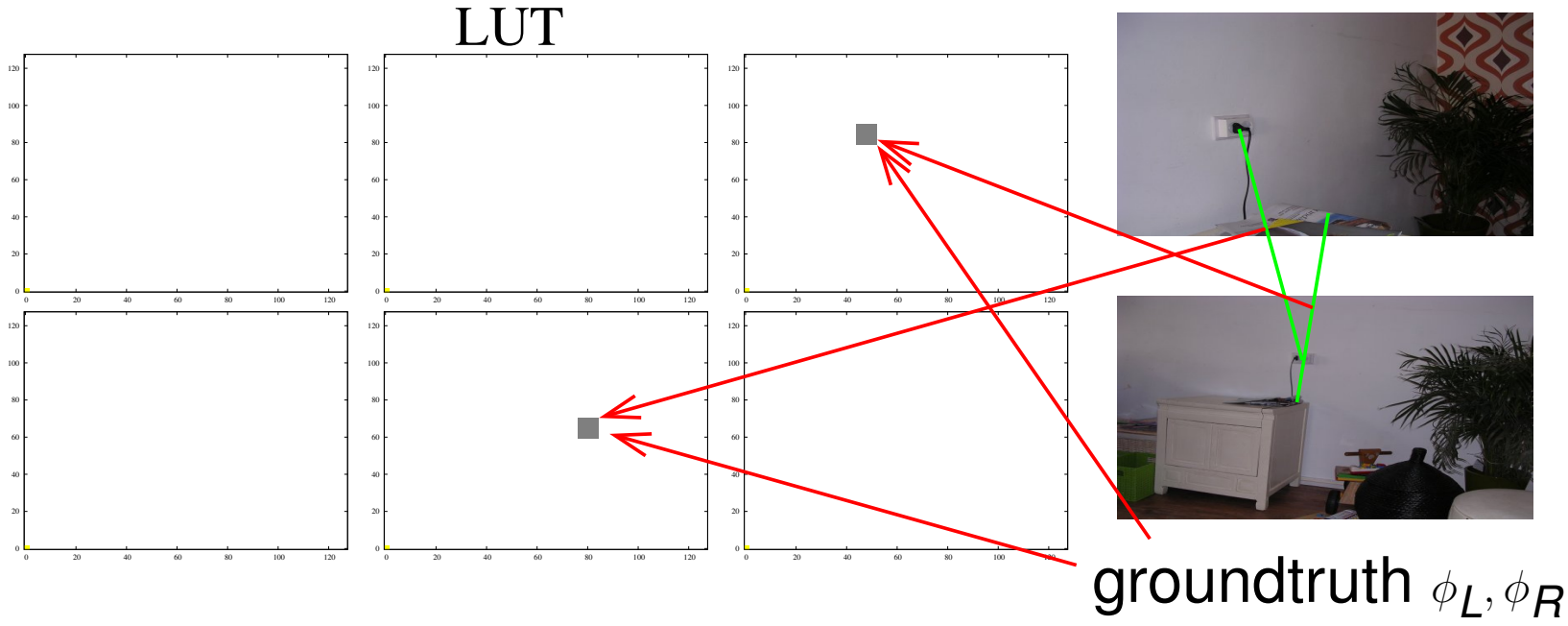




groundtruth ϕ_L, ϕ_R

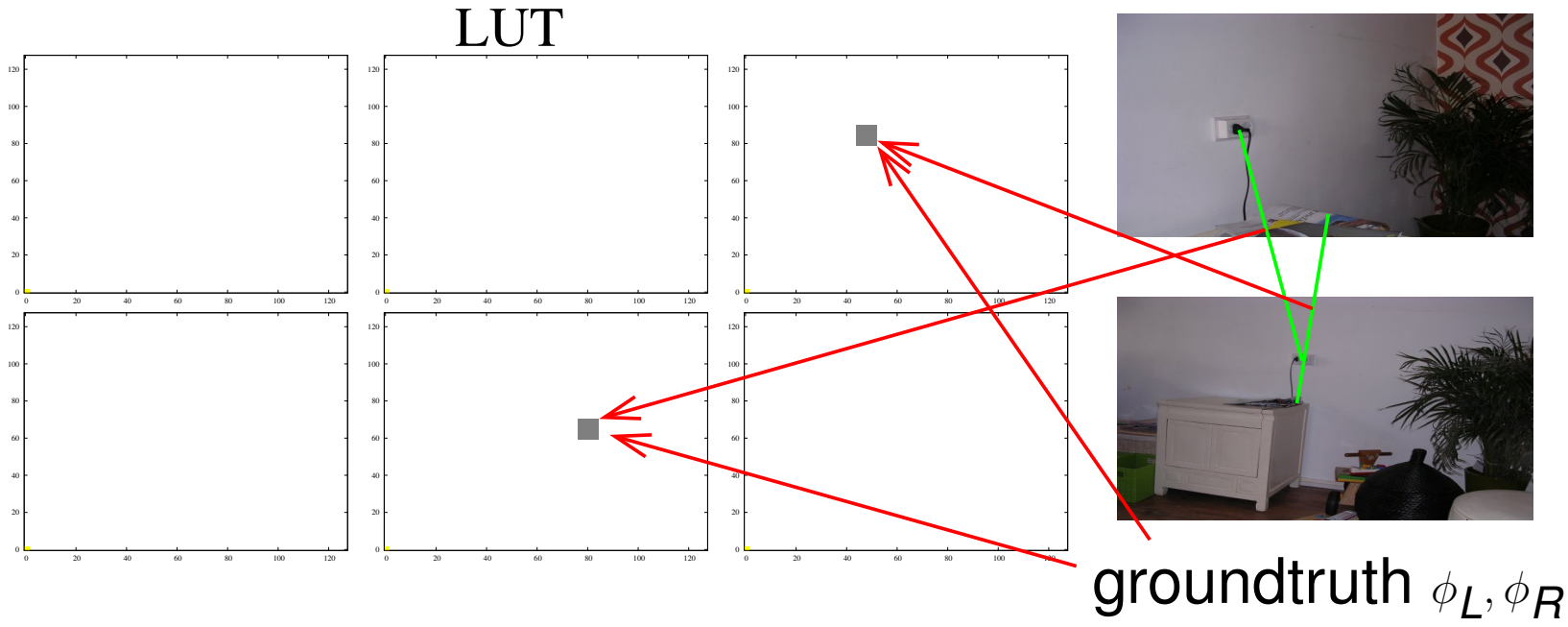
- For each image with ground truth pose
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT-slice.
- Compute $\phi_L - \beta_L$ and $\phi_R - \beta_R$ and add to the corresponding bin to pose prior $p(\phi_L, \phi_R)$
- Do not treat mismatches separately
- Finally normalize and log each slice

Proposed method: Building a LUT



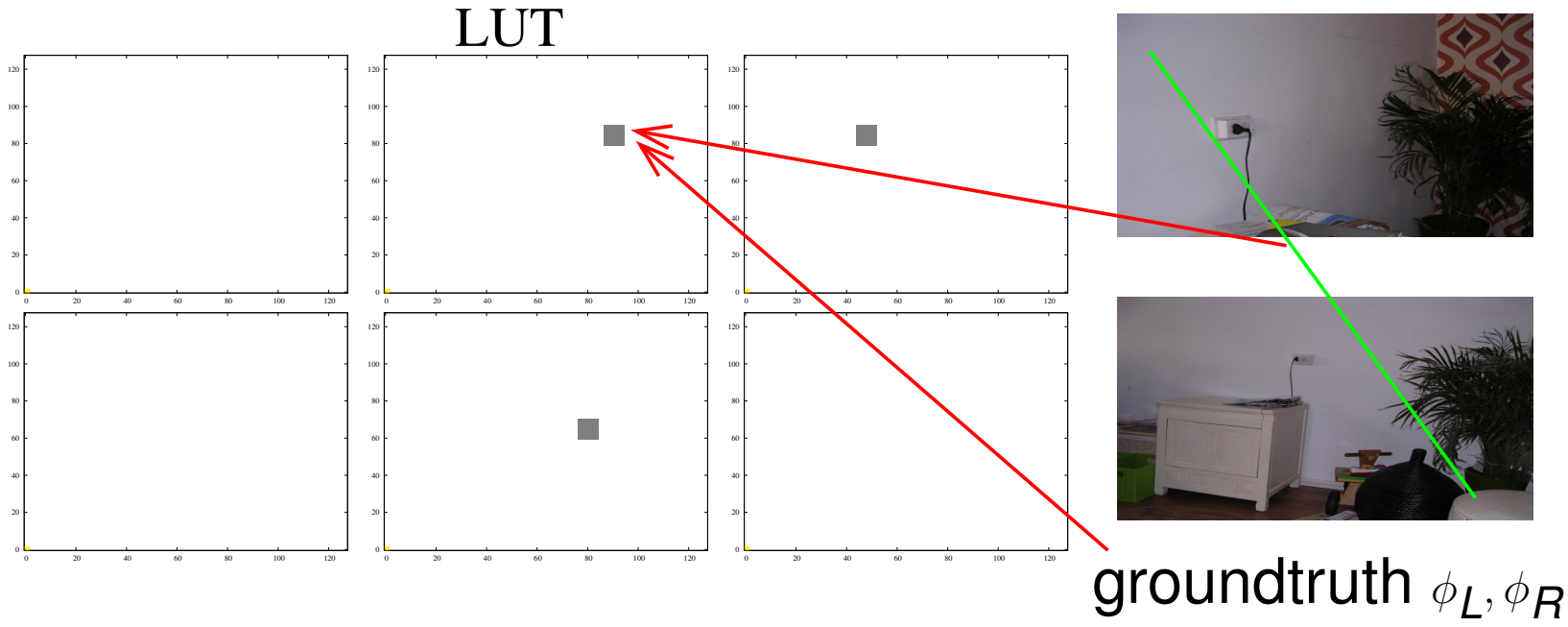
- For each image with ground truth pose
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT-slice.
- Compute $\phi_L - \beta_L$ and $\phi_R - \beta_R$ and add to the corresponding bin to pose prior $p(\phi_L, \phi_R)$
- Do not treat mismatches separately
- Finally normalize and log each slice

Proposed method: Building a LUT



- For each image with ground truth pose
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT-slice.
- Compute $\phi_L - \beta_L$ and $\phi_R - \beta_R$ and add to the corresponding bin to pose prior $p(\phi_L, \phi_R)$
- Do not treat mismatches separately
- Finally normalize and log each slice

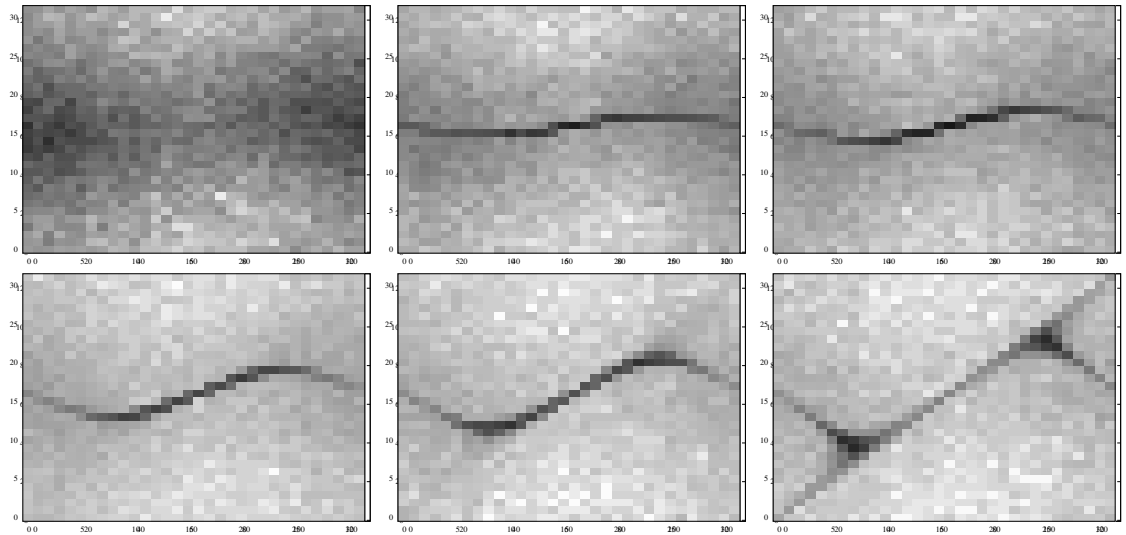
Proposed method: Building a LUT



- For each image with ground truth pose
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT-slice.
- Compute $\phi_L - \beta_L$ and $\phi_R - \beta_R$ and add to the corresponding bin to pose prior $p(\phi_L, \phi_R)$
- Do not treat mismatches separately
- Finally normalize and log each slice

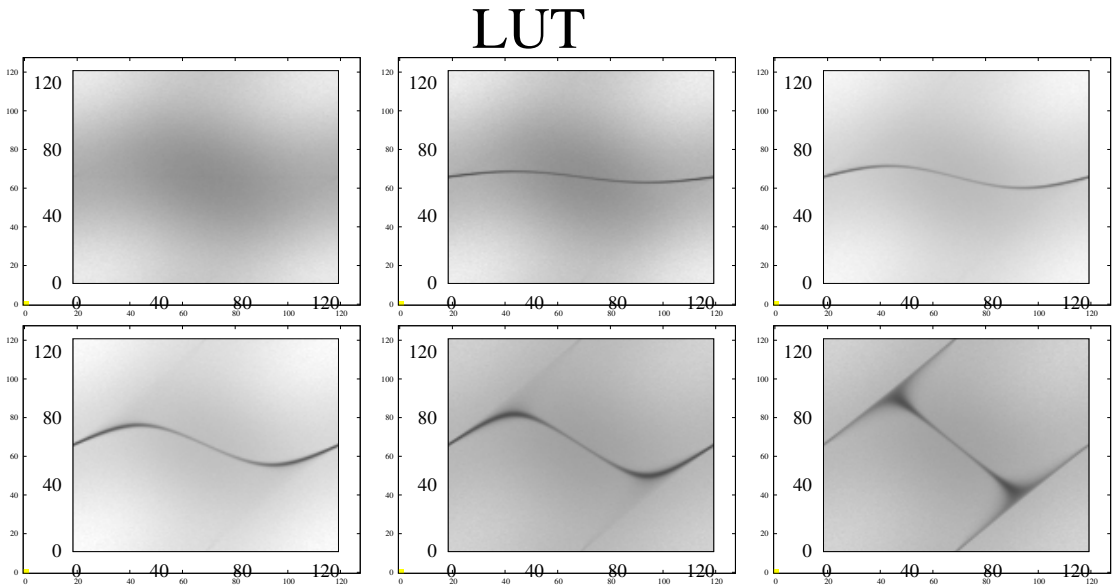
Proposed method: Building a LUT

LUT



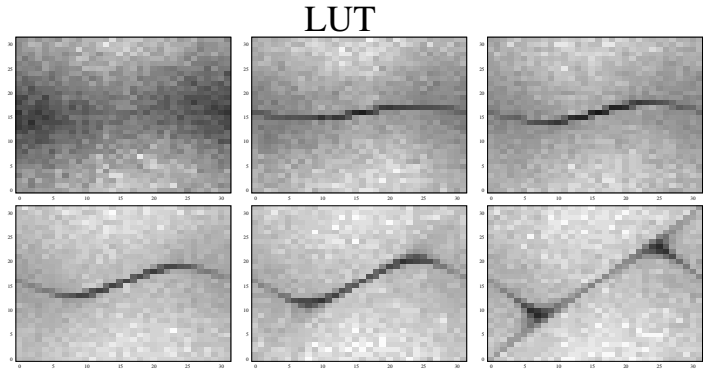
- For each image with ground truth pose
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT-slice.
- Compute $\phi_L - \beta_L$ and $\phi_R - \beta_R$ and add to the corresponding bin to pose prior $p(\phi_L, \phi_R)$
- Do not treat mismatches separately
- Finally normalize and log each slice

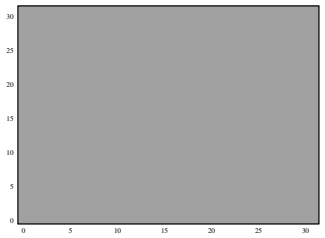
Proposed method: Building a LUT



- For each image with ground truth pose
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT-slice.
- Compute $\phi_L - \beta_L$ and $\phi_R - \beta_R$ and add to the corresponding bin to pose prior $p(\phi_L, \phi_R)$
- Do not treat mismatches separately
- Finally normalize and log each slice

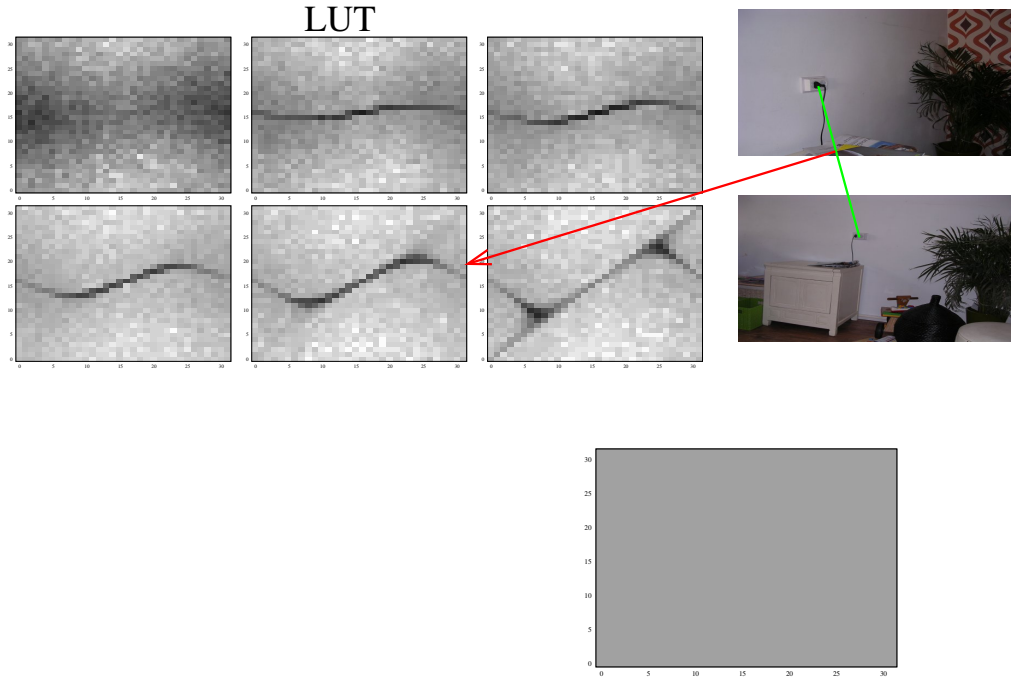
Proposed method: Using a LUT





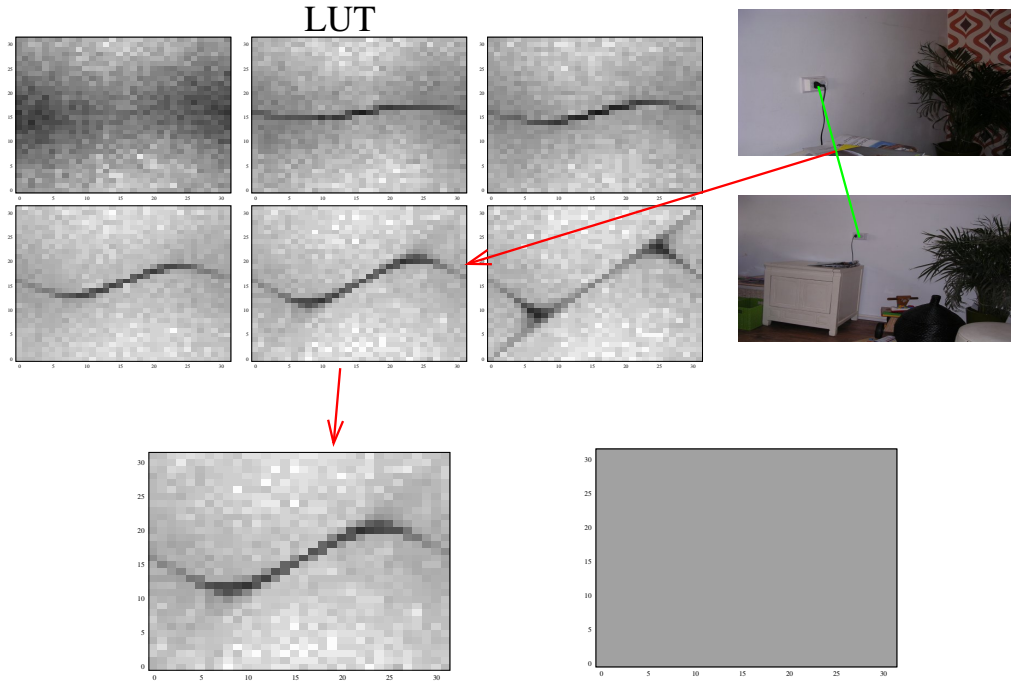
- Begin with a log pose prior (eg. uniform)
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT slice
- Shift it by β_L and β_R
- Add it to the pose prior
- Perform exp() and normalize to get proper distribution

Proposed method: Using a LUT



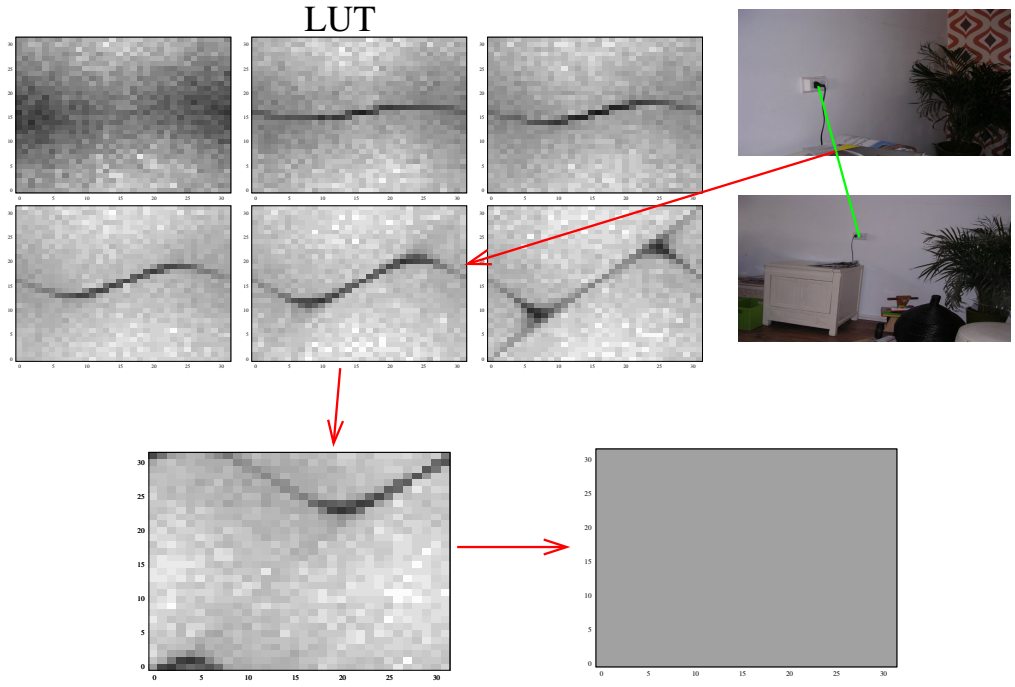
- Begin with a log pose prior (eg. uniform)
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT slice
- Shift it by β_L and β_R
- Add it to the pose prior
- Perform exp() and normalize to get proper distribution

Proposed method: Using a LUT



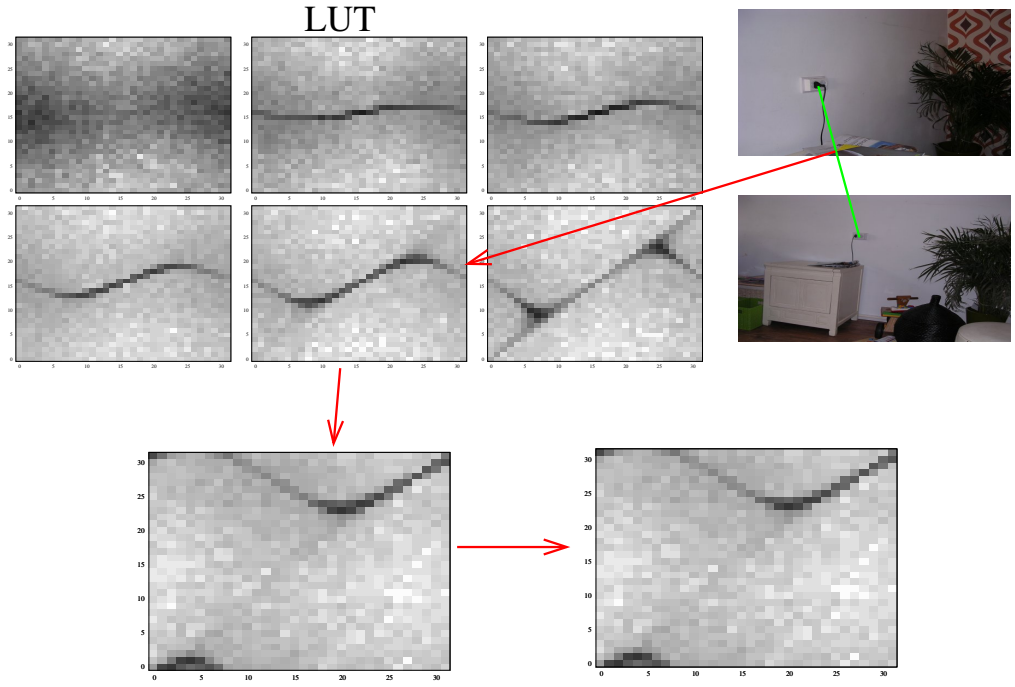
- Begin with a log pose prior (eg. uniform)
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT slice
- Shift it by β_L and β_R
- Add it to the pose prior
- Perform exp() and normalize to get proper distribution

Proposed method: Using a LUT



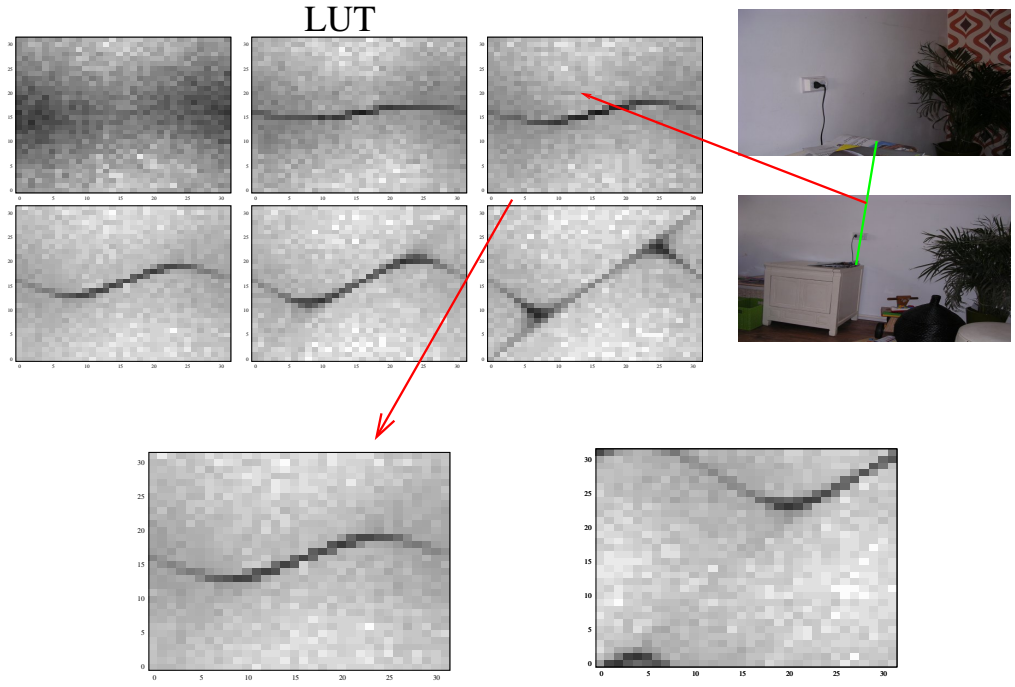
- Begin with a log pose prior (eg. uniform)
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT slice
- Shift it by β_L and β_R
- Add it to the pose prior
- Perform exp() and normalize to get proper distribution

Proposed method: Using a LUT



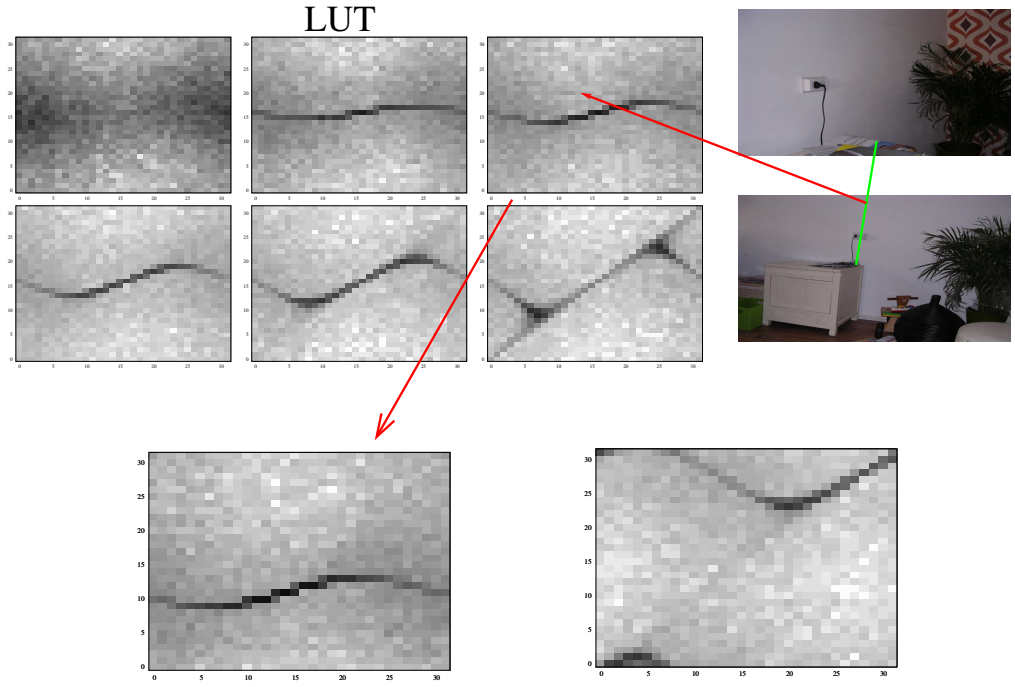
- Begin with a log pose prior (eg. uniform)
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT slice
- Shift it by β_L and β_R
- Add it to the pose prior
- Perform exp() and normalize to get proper distribution

Proposed method: Using a LUT



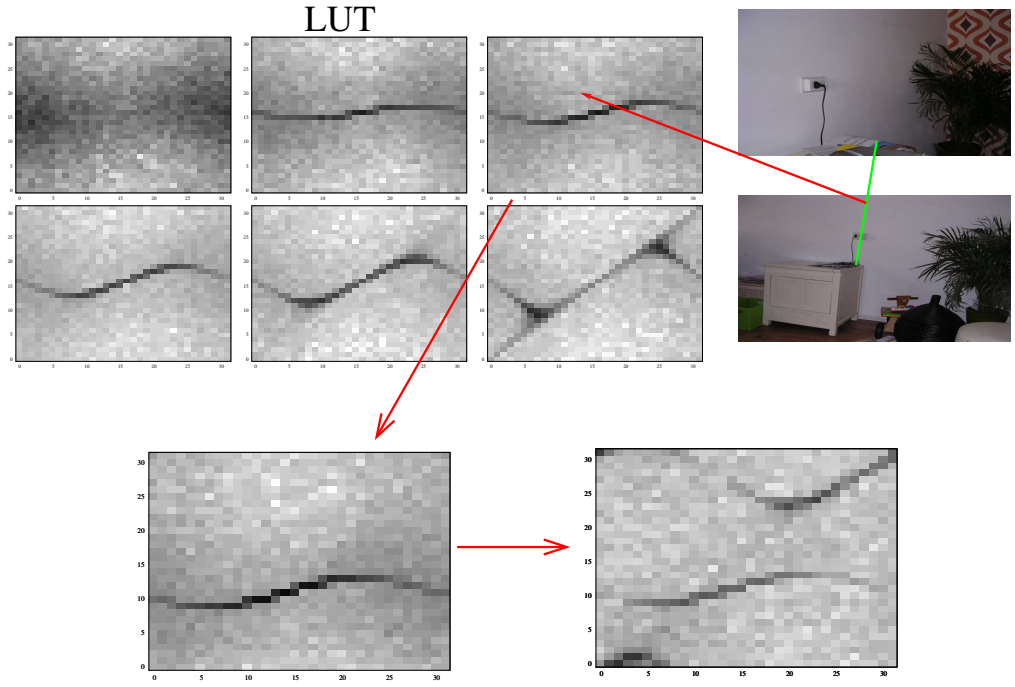
- Begin with a log pose prior (eg. uniform)
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT slice
- Shift it by β_L and β_R
- Add it to the pose prior
- Perform exp() and normalize to get proper distribution

Proposed method: Using a LUT



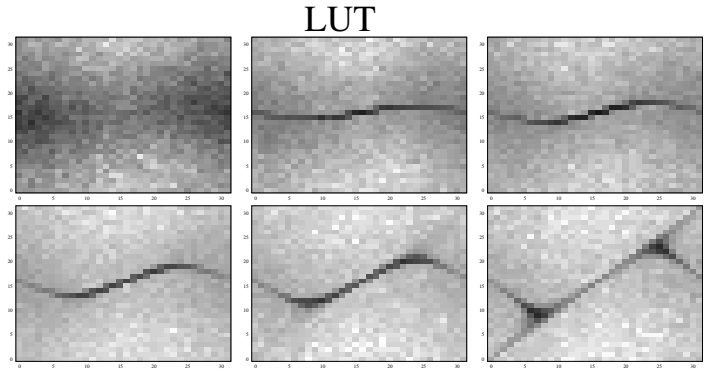
- Begin with a log pose prior (eg. uniform)
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT slice
- Shift it by β_L and β_R
- Add it to the pose prior
- Perform exp() and normalize to get proper distribution

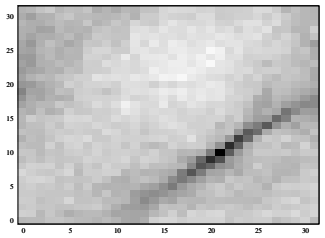
Proposed method: Using a LUT



- Begin with a log pose prior (eg. uniform)
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT slice
- Shift it by β_L and β_R
- Add it to the pose prior
- Perform exp() and normalize to get proper distribution

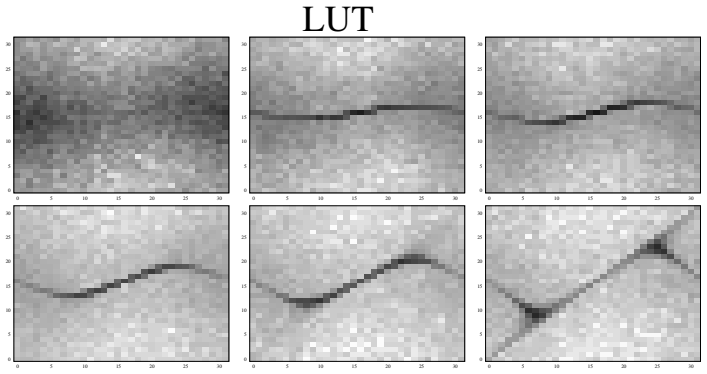
Proposed method: Using a LUT

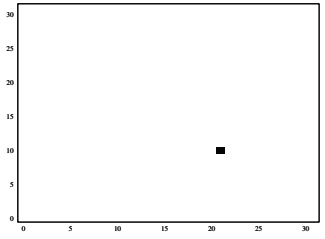




- Begin with a log pose prior (eg. uniform)
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT slice
- Shift it by β_L and β_R
- Add it to the pose prior
- Perform exp() and normalize to get proper distribution

Proposed method: Using a LUT





- Begin with a log pose prior (eg. uniform)
- For each correspondence compute $\frac{\tan(\alpha_R)}{\tan(\alpha_L)}$ to pick LUT slice
- Shift it by β_L and β_R
- Add it to the pose prior
- Perform exp() and normalize to get proper distribution

Outline

Evaluation: Qualitative

Pro's & Con's of LUT based method

pro

- no explicit error model
- small cpu usage
- full likelihood
  - multiple modes
  - unbiased estimate of confidence interval

con

- discretization error
- large memory usage

Evaluation: Qualitative

Pro's & Con's of LUT based method

con

- discretization error
- large memory usage

pro

- no explicit error model
- small cpu usage
- full likelihood
  - multiple modes
  - unbiased estimate of confidence interval

Evaluation: Qualitative

Pro's & Con's of LUT based method

pro

- no explicit error model
- small cpu usage
- full likelihood
  - multiple modes
  - unbiased estimate of confidence interval

con

- discretization error
- large memory usage

Evaluation: Qualitative

Pro's & Con's of LUT based method

pro

- no explicit error model
- small cpu usage
- full likelihood
  - multiple modes
  - unbiased estimate of confidence interval

con

- discretization error
- large memory usage

Evaluation: Qualitative

Pro's & Con's of LUT based method

con

- discretization error
- large memory usage

pro

- no explicit error model
- small cpu usage
- full likelihood
  - multiple modes
  - unbiased estimate of confidence interval

Evaluation: Simulator

Model

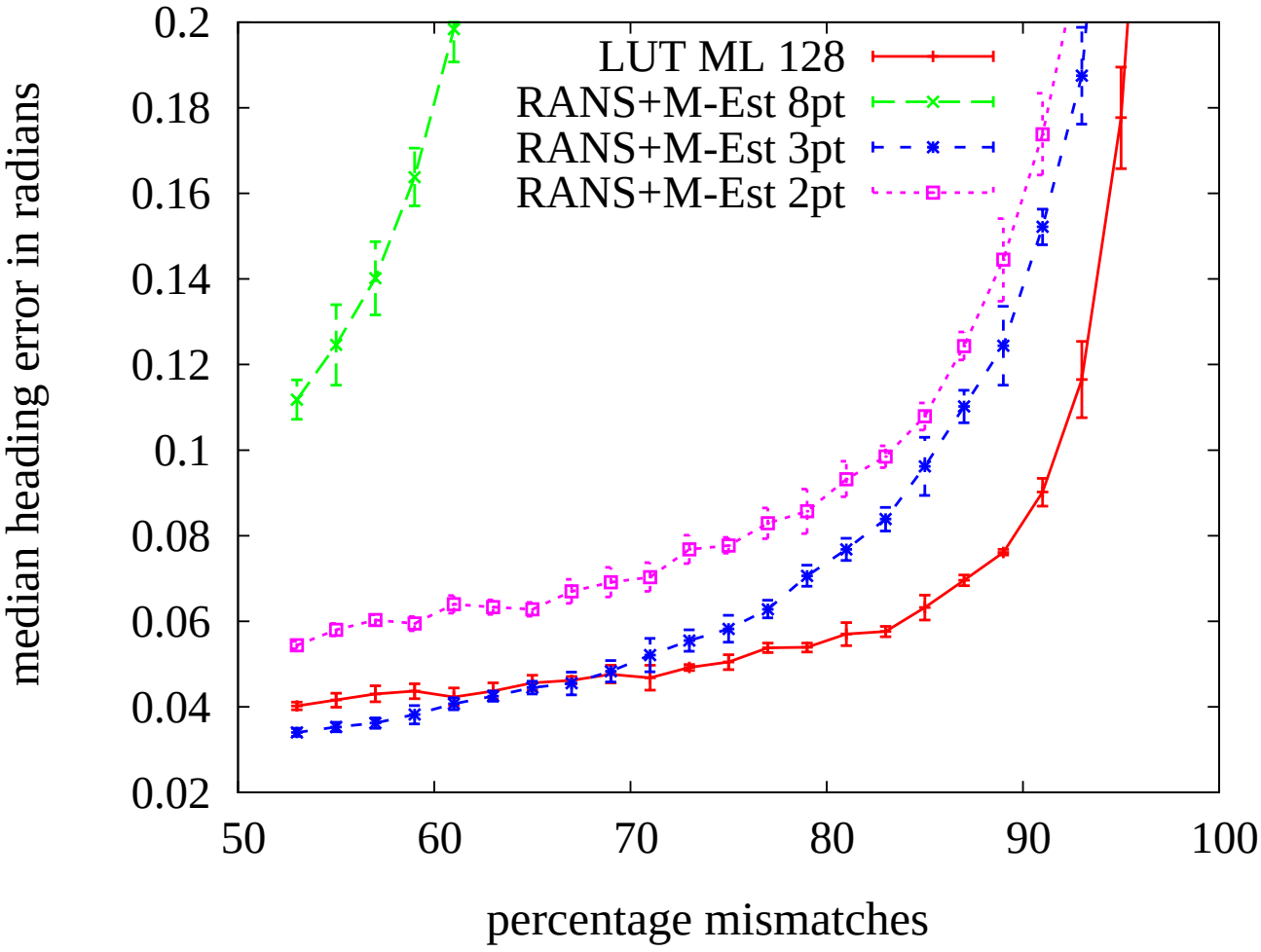
- unlimited omnidirectional view
- only planar motion
- 100 points around camera, average distance 2*camera distance
- image projection noise of +/- .5 degrees

LUT

- 128^3 bins
- 10^{10} samples

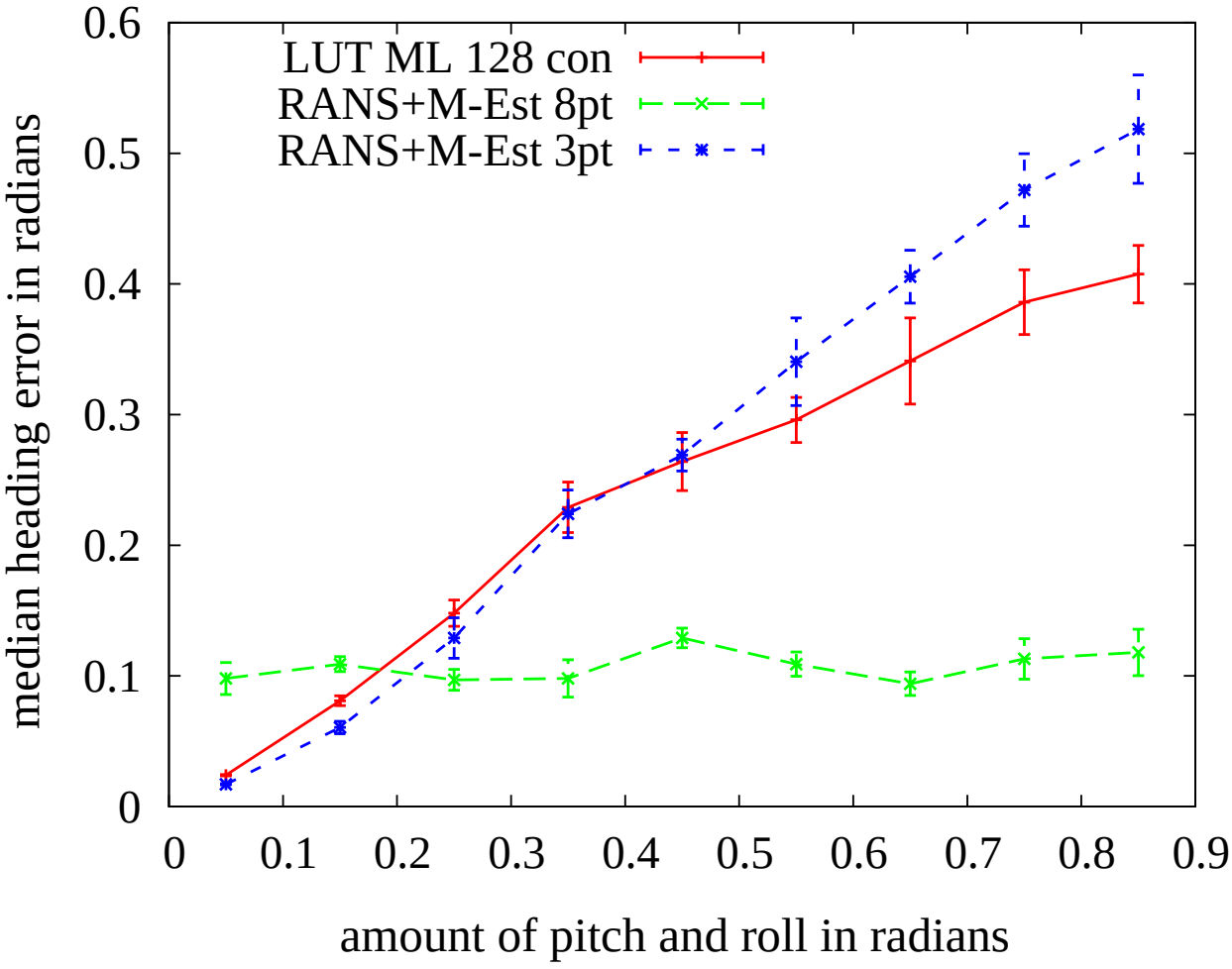
Evaluation: Simulator

Simulation - vary number of mismatches



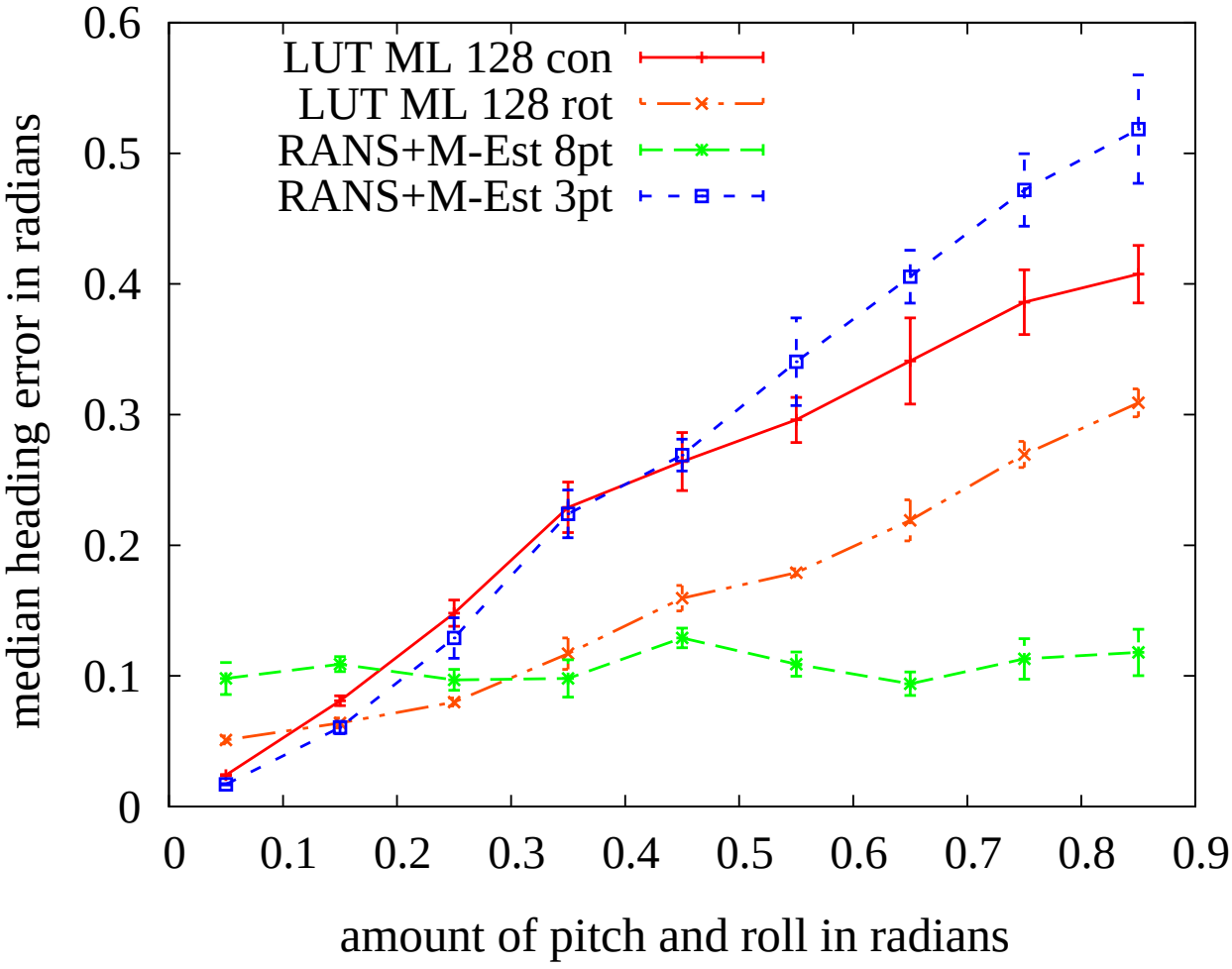
Evaluation: Simulator

Simulation - vary out of plane rotation



Evaluation: Simulator

Simulation - vary out of plane rotation



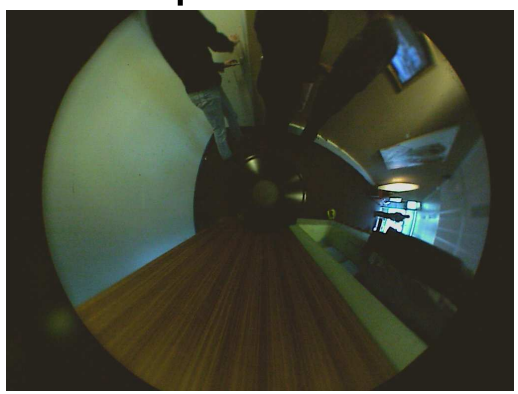
Evaluation: Image datasets

Real home data

- omnidirectional camera mounted on Nomad Scout or Biron
- 3 real homes
- +/- 10^4 images
- > 10^6 image pairs
- ground truth from odometry+laser based SLAM

Almere

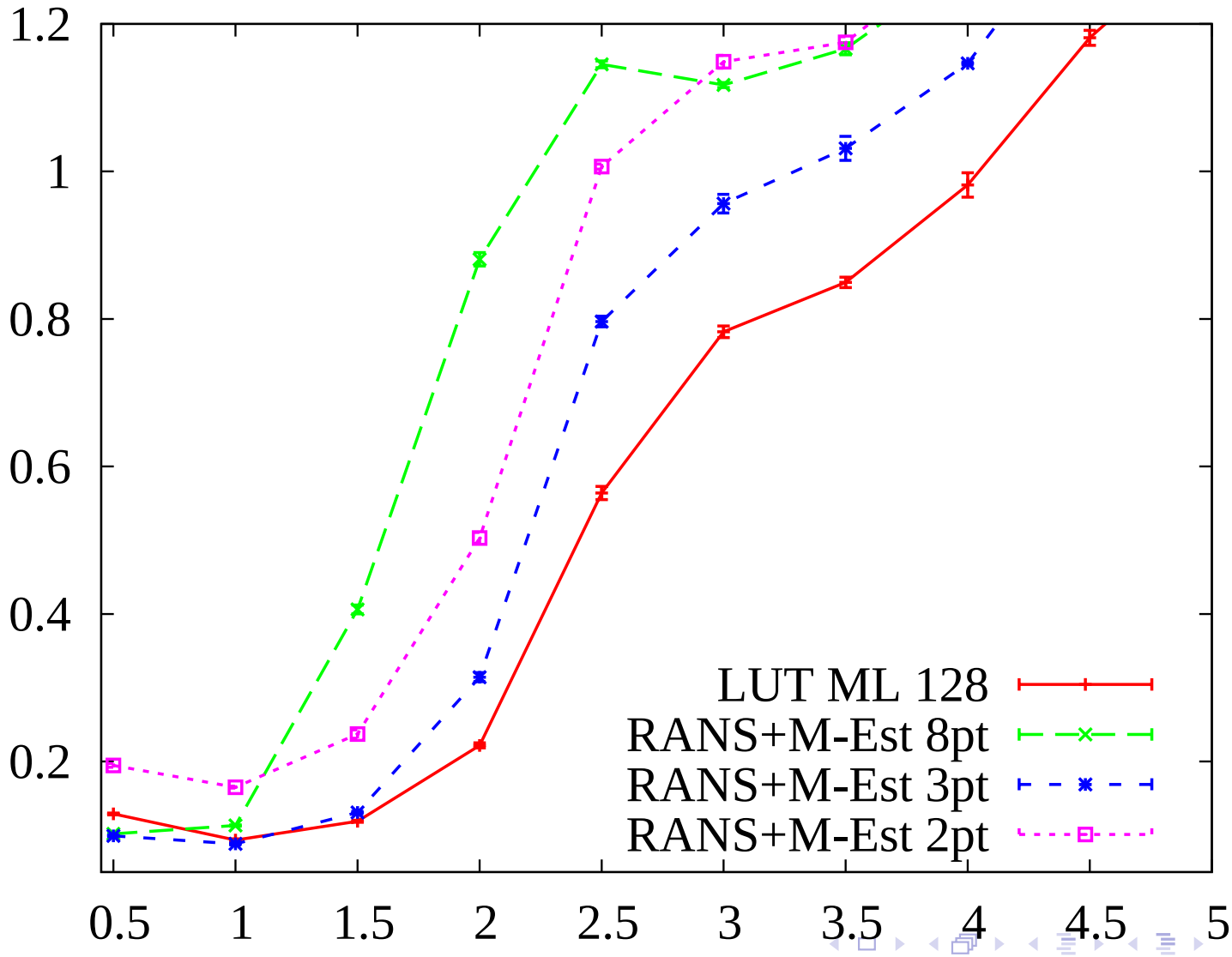
Spaan



Biron

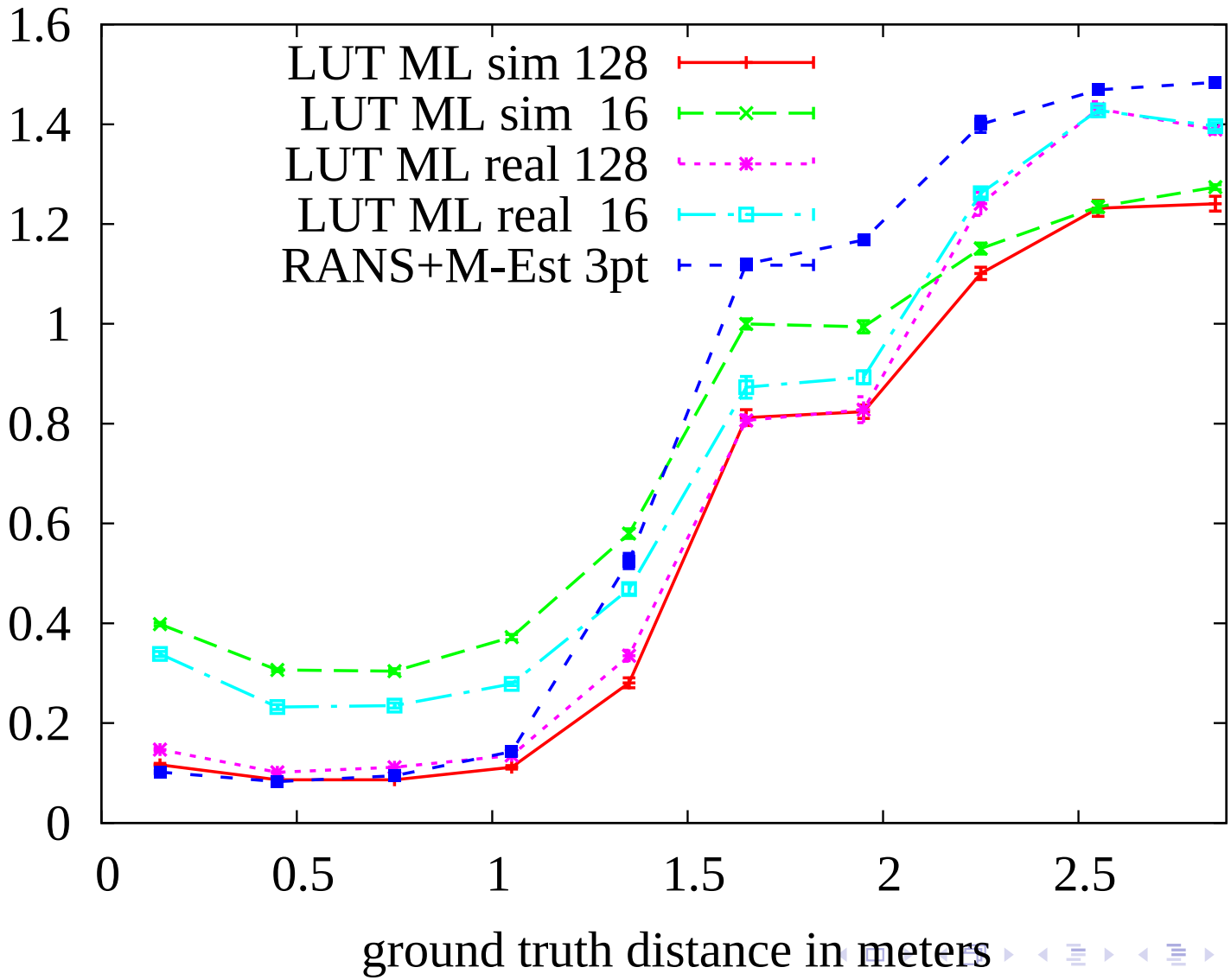
Evaluation: Image datasets

Almere set using simulated LUT vary distance



Evaluation: Image datasets

Spaan set using learned/simulated LUT vary distance



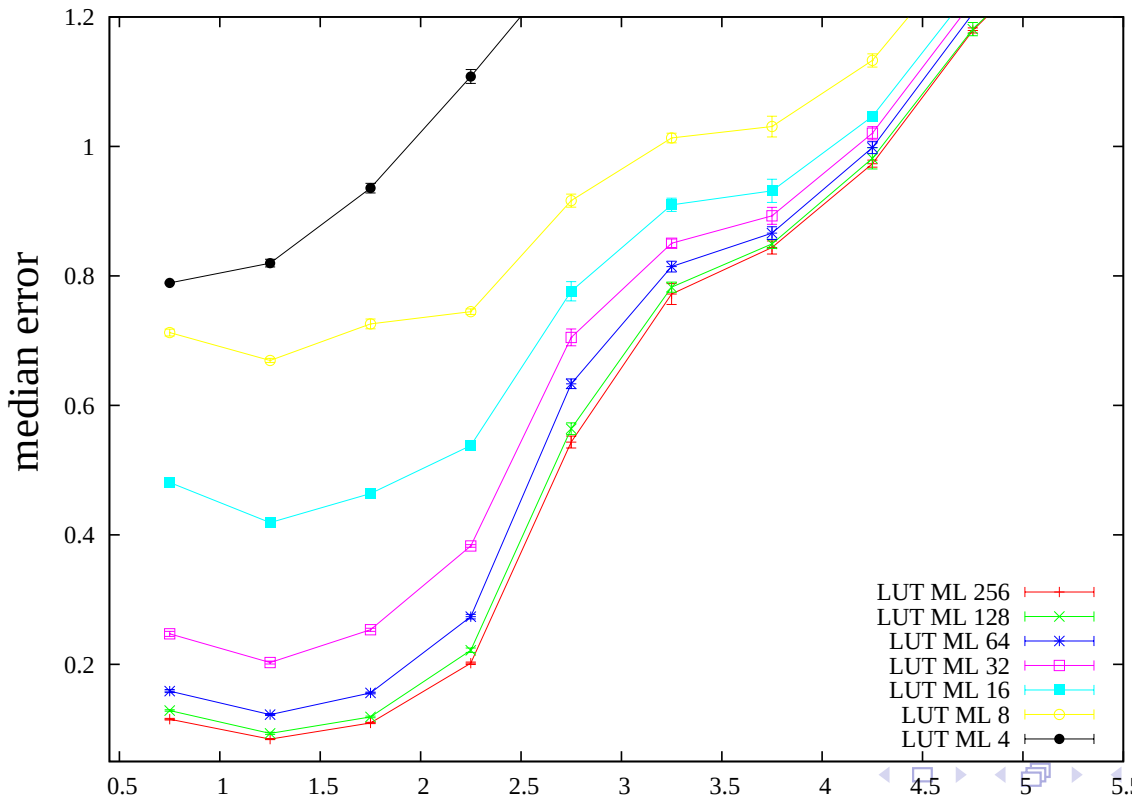
Evaluation: Image datasets

cpu-time in ms

M-Est 8pt	M-Est 3pt	M-Est 2pt
3.6	3.8	0.68

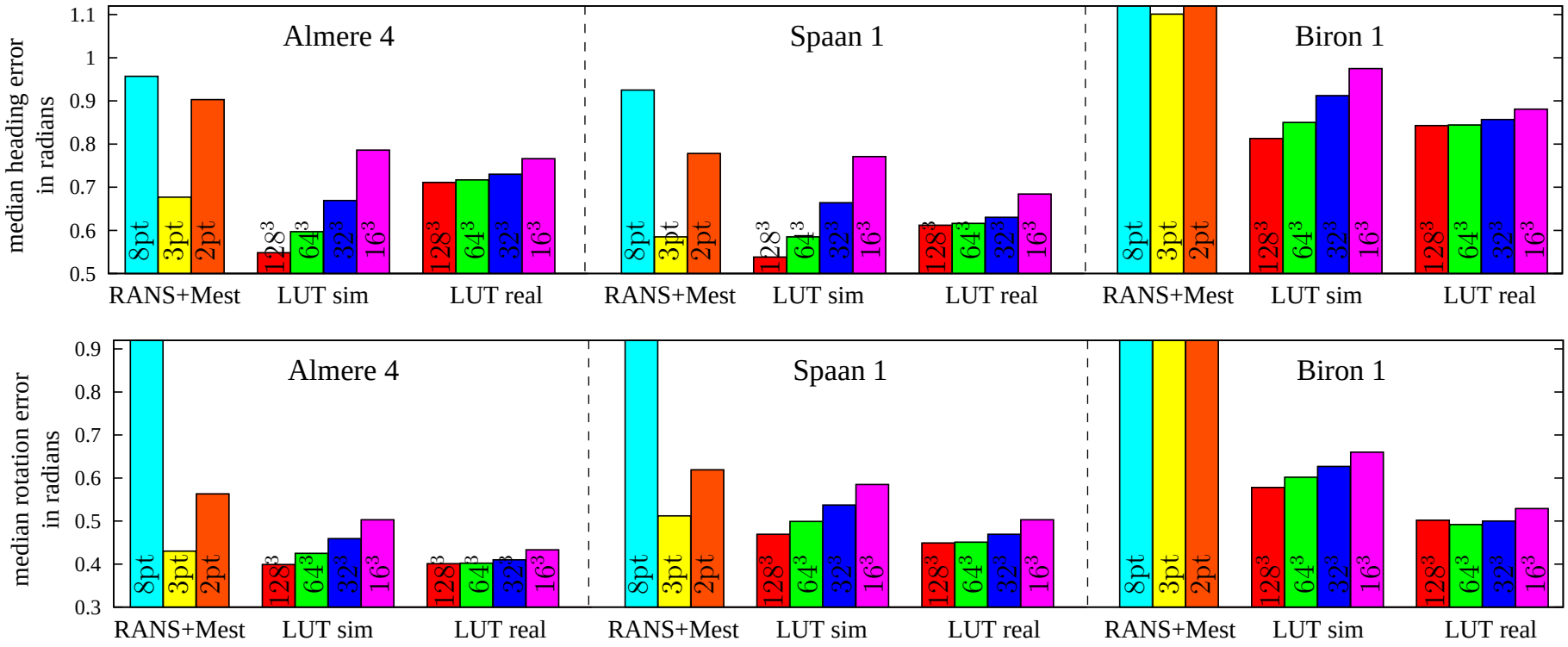
LUT binsize vs cpu-time

256	128	64	32	16	8	4
10.1	1.3	0.25	0.070	0.034	0.022	0.016



Evaluation: Image datasets

Real home data - overview

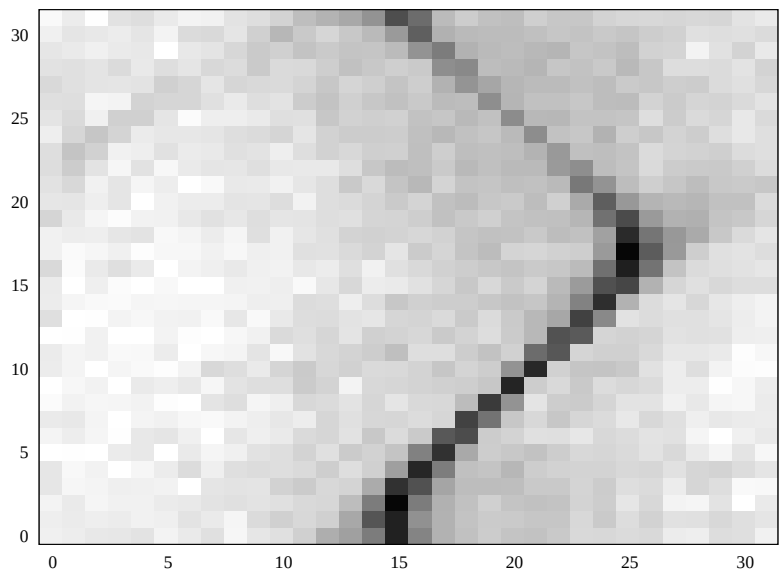


Outline

Discussion: Full likelihood

Advantage of having full likelihood

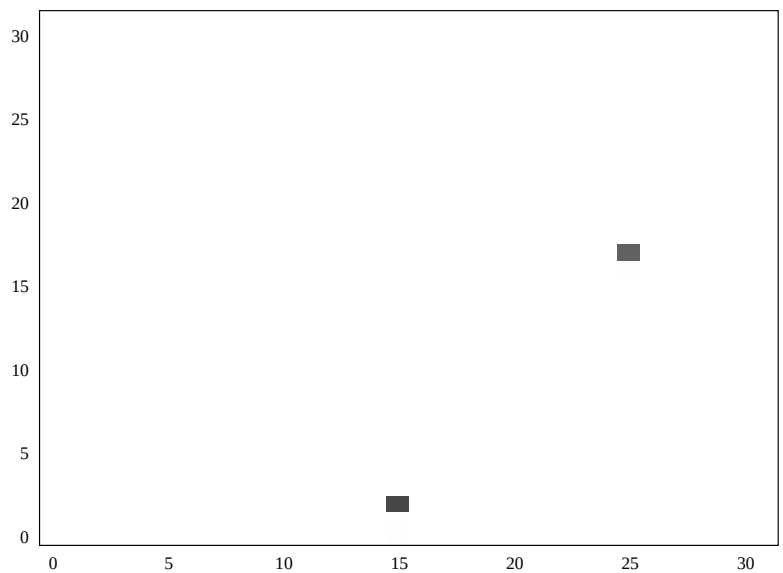
- Multiple modes
- Degenerate cases
- Uncertainty estimation, e.g. for SLAM
- Topological mapping using proper probability rather than the "geometric "



Discussion: Full likelihood

Advantage of having full likelihood

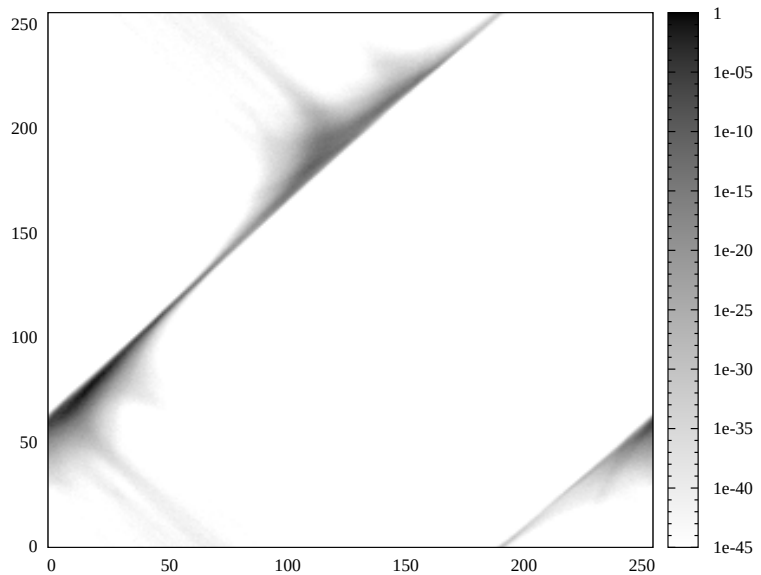
- Multiple modes
- Degenerate cases
- Uncertainty estimation, e.g. for SLAM
- Topological mapping using proper probability rather than the "geometric "



Discussion: Full likelihood

Advantage of having full likelihood

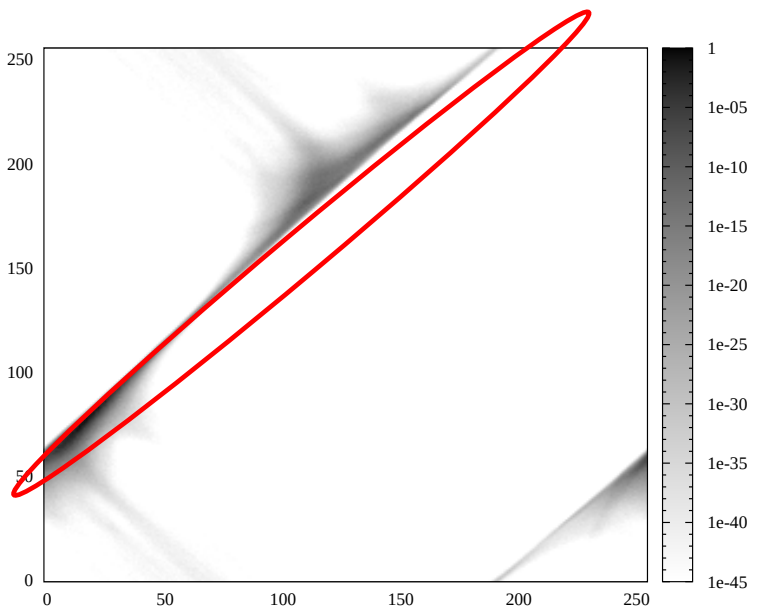
- Multiple modes
- Degenerate cases
- Uncertainty estimation, e.g. for SLAM
- Topological mapping using proper probability rather than the "geometric "



Discussion: Full likelihood

Advantage of having full likelihood

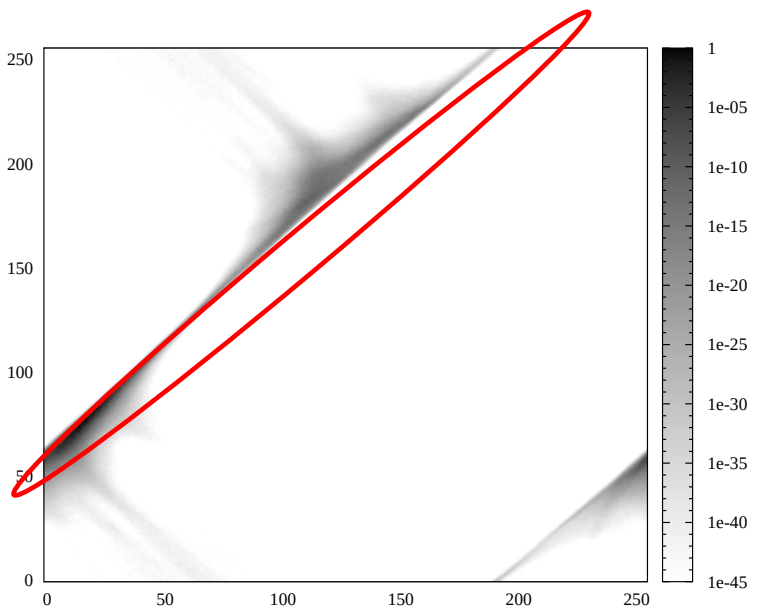
- Multiple modes
- Degenerate cases
- Uncertainty estimation, e.g. for SLAM
- Topological mapping using proper probability rather than the "geometric "



Discussion: Full likelihood

Advantage of having full likelihood

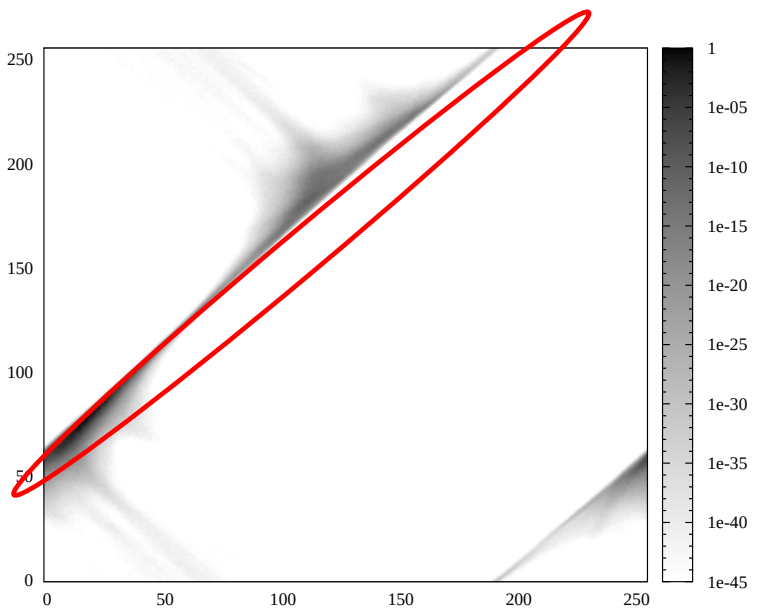
- Multiple modes
- Degenerate cases
- Uncertainty estimation, e.g. for SLAM
- Topological mapping using proper probability rather than the "geometric check"



Discussion: Full likelihood

Advantage of having full likelihood

- Multiple modes
- Degenerate cases
- Uncertainty estimation, e.g. for SLAM
- Topological mapping using proper probability rather than the "geometric hack"



Outline

Conclusions

The developed LUT based pose estimator is

- more accurate
- more efficient
- easy to understand and implement

but also raises questions:

- How could we use a full likelihood in a SLAM algorithm?
- Is it as good for perspective images?
- Could non-uniform discretization be used?
- Is it applicable on non-calibrated cameras
- Does it work on features cheaper than SIFT

Conclusions

The developed LUT based pose estimator is

- more accurate
- more efficient
- easy to understand and implement

but also raises questions:

- How could we use a full likelihood in a SLAM algorithm?
- Is it as good for perspective images?
- Could non-uniform discretization be used?
- Is it applicable on non-calibrated cameras
- Does it work on features cheaper than SIFT

Thanks

Questions....