Introduction to **Dense** Visual Camera Tracking

Richard Newcombe, University of Washington

CVPR 2014 Visual SLAM Tutorial

People and Recent Visual SLAM theses, Imperial College, London with Andrew Davison

Steven Lovegrove:
"Parametric Dense Visual SLAM", 2011

Ankur Handa:
"High Frame Rate, Dense Visual SLAM", 2013

Hauke Strasdat:
"Local Accuracy and Global Consistency for Efficient Visual SLAM", 2012

Richard Newcombe:
"Dense Visual SLAM", 2013

Thanks to Ankur, Steve and Hauke for images and slides I've incorporated here.

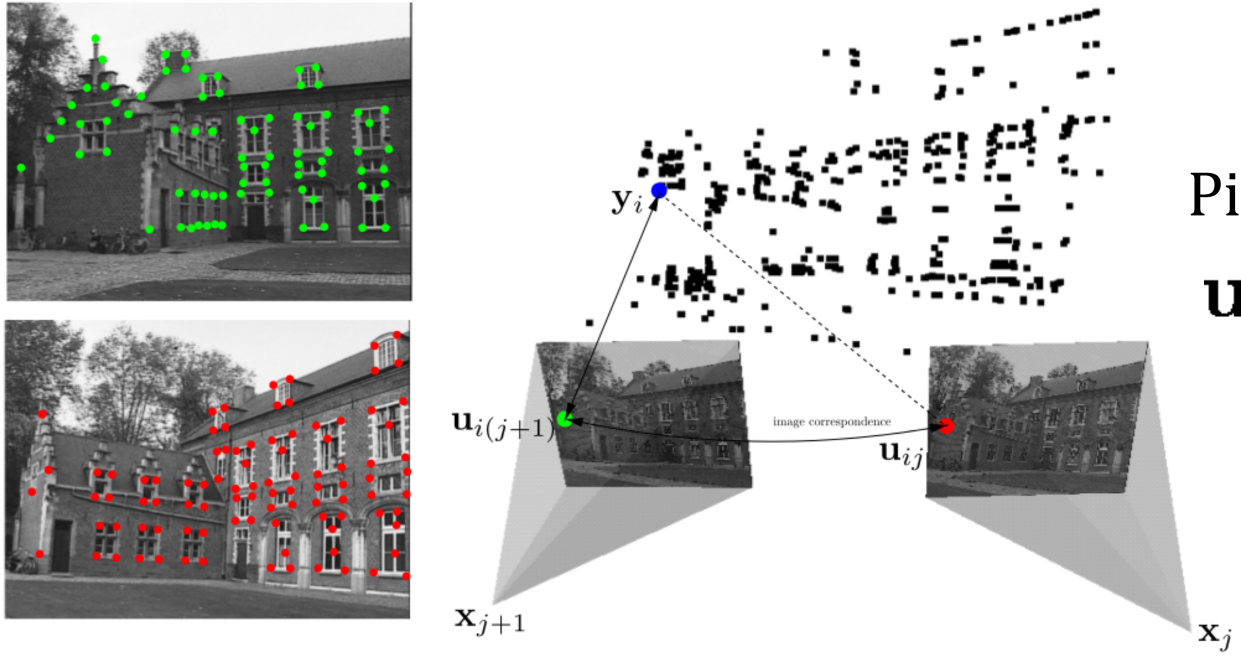
Dense Tracking Introduction Outline

1. Generative Models and the Dense Tracking advantage

2. Basic Gauss-Newton Optimisation for direct whole image alignment models

3. Example Dense tracking
   a. SO3 tracking of a passive camera
   b. SE3 tracking given RGB-D images
   c. SE3 tracking using Depth images

Recall the projection function:



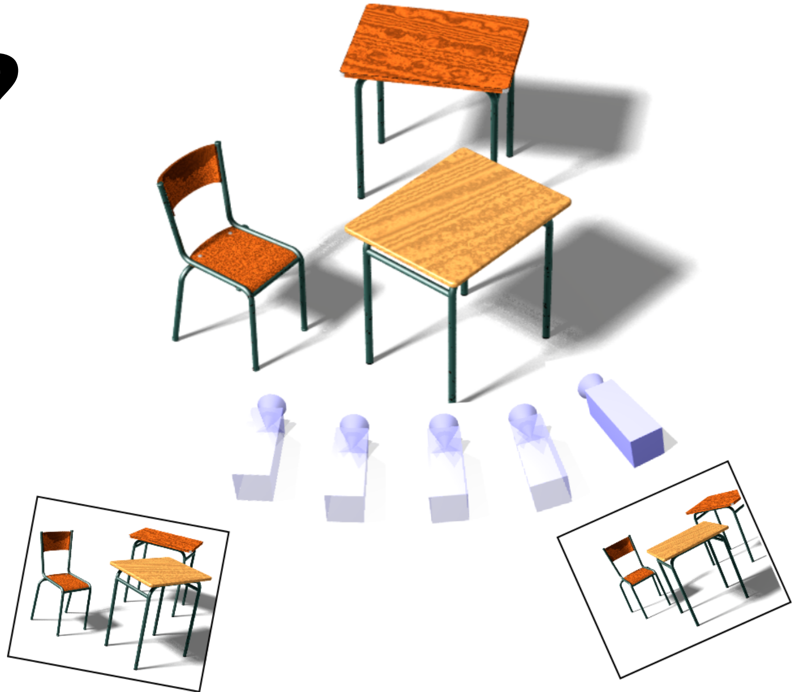
Pinhole Projection:

$$\mathbf{u} = \pi\left(KT(\mathbf{x})\mathbf{y}\right)$$

Thanks to Prof. Pollefeys for the original figure (3DIM ‘99).

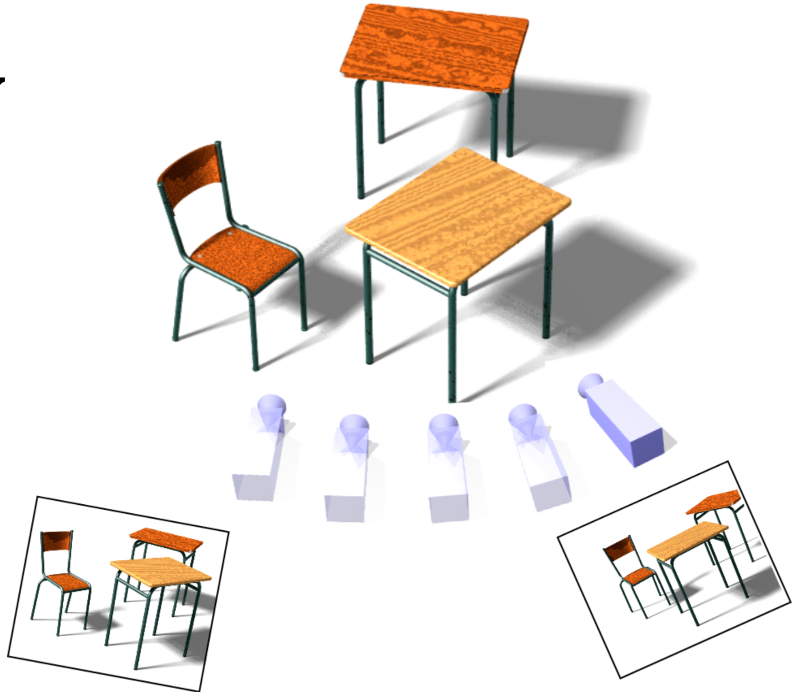
How can we use more of the image data?

We will contrast with explicit feature extraction and matching as used in sparse VO

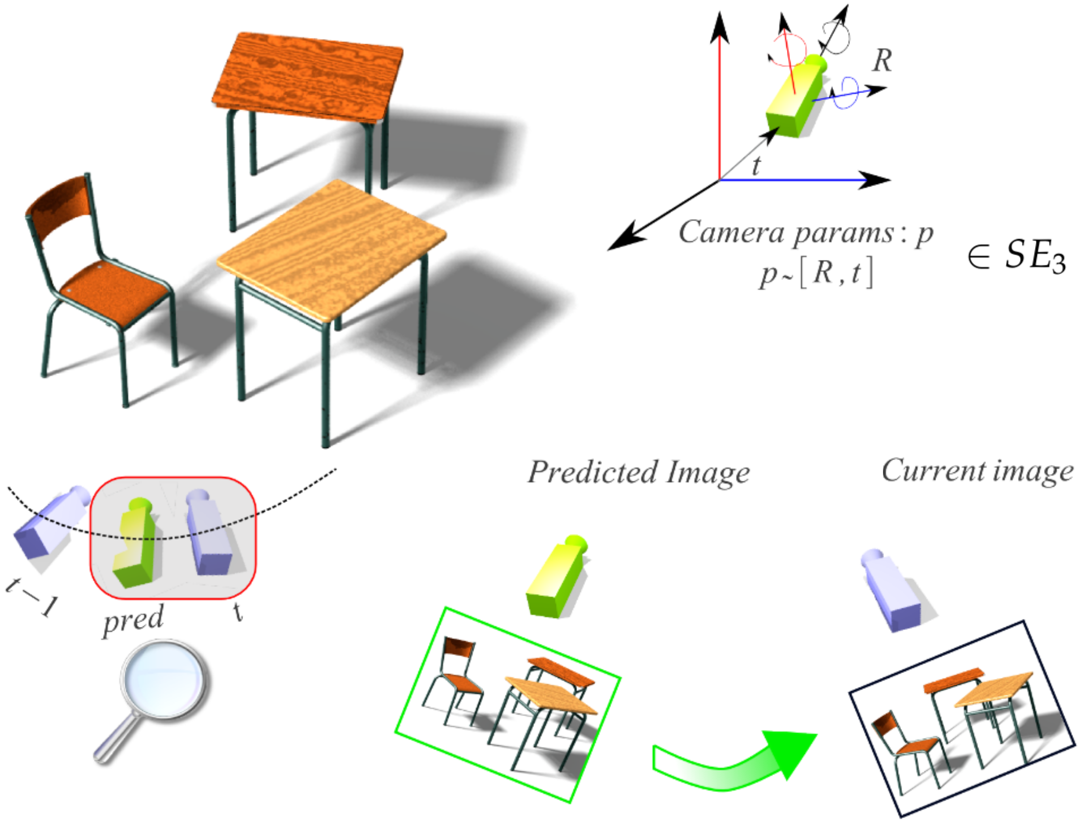


Solve for correspondence and camera motion simultaneously

Assumption: Observation function that can *render* a **dense** image prediction *given* a camera pose.



Overview of Dense Visual Tracking



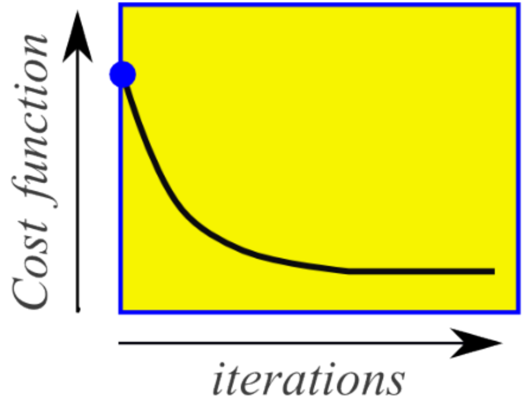
Dense VO Generative Model Intuition

- ➔ Given a dense, textured, *surface model* of a scene we can predict what should be seen in that camera by rendering

- ➔ If the *model* is good and the camera pose is correct, then the image prediction is close to true image observed

Whole Image Cost

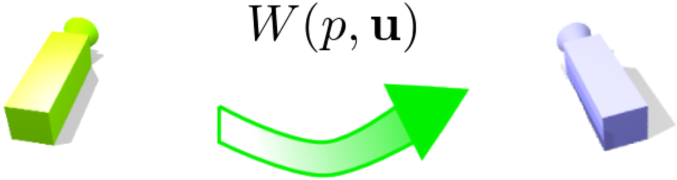
$$C = \min\left\{\sum\left(I(\mathbf{u}) - T(W(p+\Delta p, \mathbf{u}))\right)^2\right\}$$



$T(\mathbf{u})$

$I(\mathbf{u})$

$W(p, \mathbf{u})$



$T(\mathbf{u})$: *Predicted Image*

$I(\mathbf{u})$: *Current image*

p : *Camera params*

$W(p, \mathbf{u})$: *Warp*

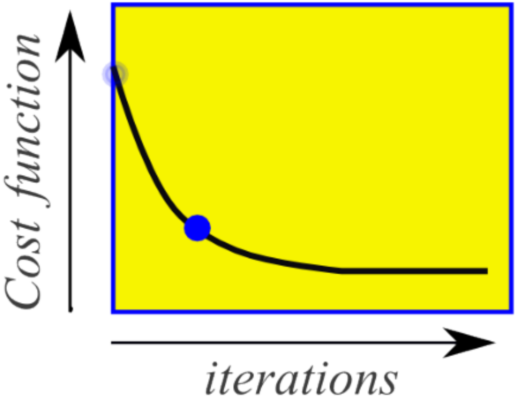
Dense 3D image alignment: Initialisation

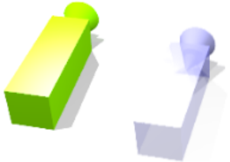
$$\Delta p_1 = argmin \sum \left(I(\mathbf{u}) - T(W(p + \Delta p, \mathbf{u})) \right)^2$$

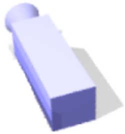
$$update: p \leftarrow p + \Delta p_1$$

$T(W(p + \Delta p_1, \mathbf{u}))$

$I(\mathbf{u})$







$T(\mathbf{u})$: *Predicted Image*

$I(\mathbf{u})$: *Current image*

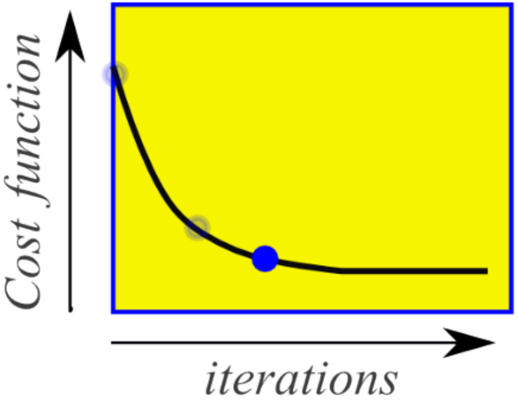
p : *Camera params*

$W(p, \mathbf{u})$: *Warp*

Dense 3D image alignment: Step 1

$$\Delta p_2 = argmin \sum \left(I(\mathbf{u}) - T(W(p + \Delta p, \mathbf{u})) \right)^2$$

$$update: p \leftarrow p + \Delta p_2$$



$T(W(p + \Delta p_2, \mathbf{u}))$

$I(\mathbf{u})$

$T(\mathbf{u})$: *Predicted Image*

$I(\mathbf{u})$: *Current image*

p : *Camera params*

$W(p, \mathbf{u})$: *Warp*

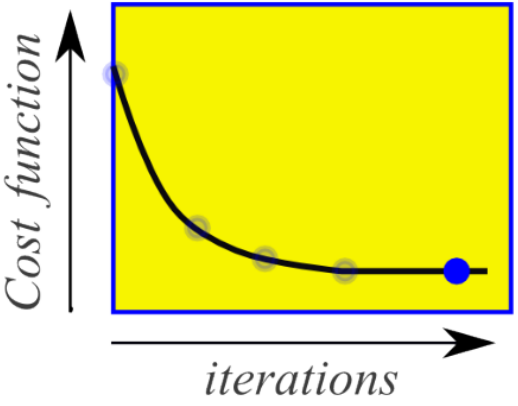
Dense 3D image alignment: Step 2

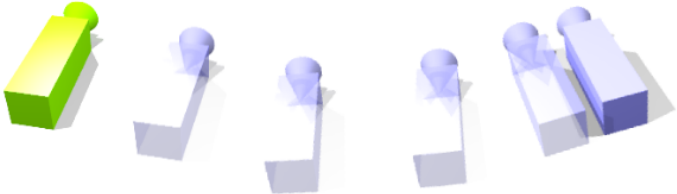
$$\Delta p_4 = \operatorname{argmin} \sum \left(I(\mathbf{u}) - T(W(p + \Delta p, \mathbf{u})) \right)^2$$

$update: p \leftarrow p + \Delta p_4$

$T(W(p + \Delta p_4, \mathbf{u}))$

$I(\mathbf{u})$





$T(\mathbf{u})$: *Predicted Image*

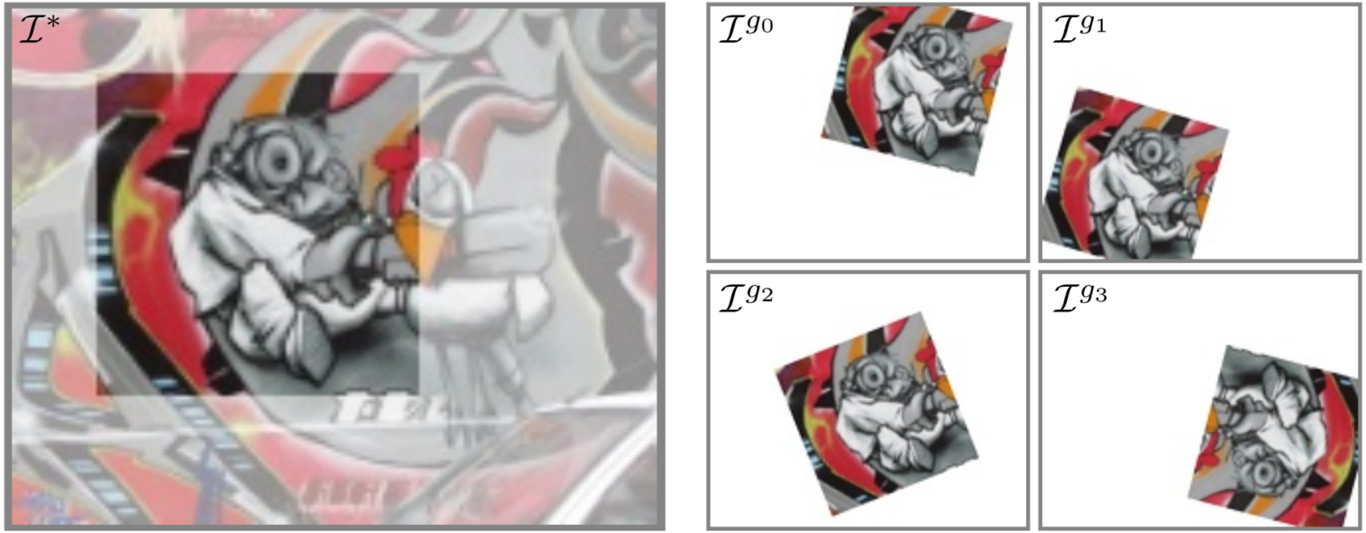
$I(\mathbf{u})$: *Current image*

p : *Camera params*

$W(p, \mathbf{u})$: *Warp*

Dense 3D image alignment: Step 4

Example Basic Generative Models



$$\mathtt{H}(\mathbf{x}) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{pmatrix}, \qquad \mathtt{I}^g \begin{pmatrix} u \\ v \end{pmatrix} = \mathtt{I}^* \left(\pi \left(\mathtt{H}(\mathbf{x}) \, (u, v, 1)^\top \right) \right).$$

Dense whole image alignment technique: *warp*

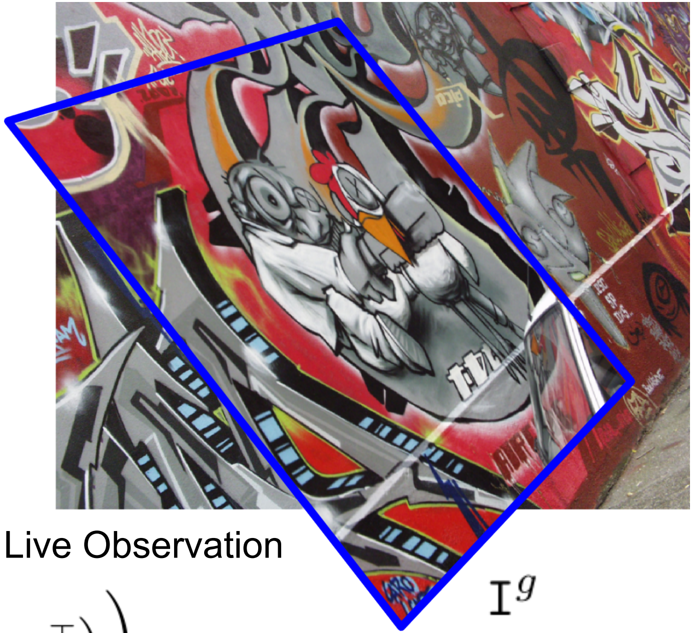
1. Define the geometric model W, with parameters $\mathbf{x}$, that transforms a pixel in one frame into another:

$$\mathtt{I}^g\begin{pmatrix} u \\ v \end{pmatrix} = \mathtt{I}^*\left(W\left(\mathbf{x};(u,v)^\top\right)\right),$$

Example generative model

Reference Image I^*

$\mathrm{I}^*\left(W\left(\mathbf{x};(u,v)^\top\right)\right)$ →



Live Observation I^g

$$\mathrm{I}^g\begin{pmatrix}u\\v\end{pmatrix} = \mathrm{I}^*\left(W\left(\mathbf{x};(u,v)^\top\right)\right)$$

Dense whole image alignment technique: *error*

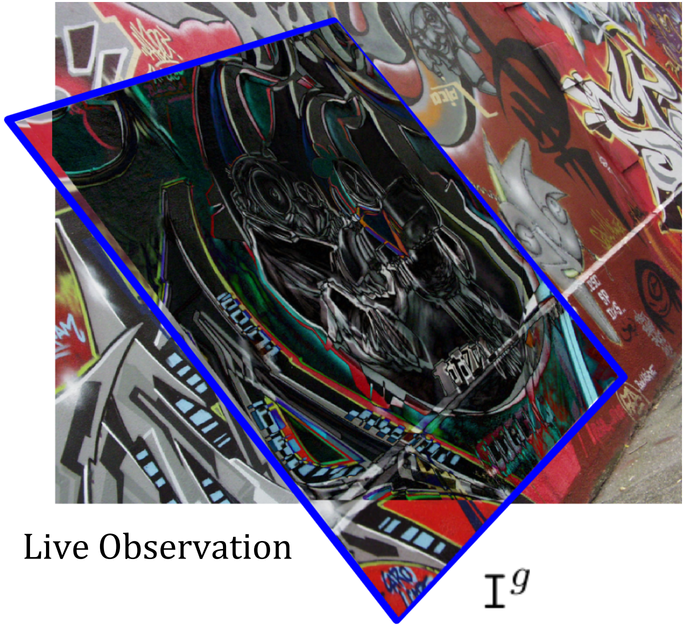
2. Define a Frame to Frame image alignment **error** and **cost function** that computes a similarity score between the image values $I^r(\mathbf{u})$ and $I^r(W(\mathbf{x},\mathbf{u}))$

$$e(u, \mathbf{x}) = \frac{1}{2} \sum_{\mathbf{u}_r \in \Omega_r} \left(I^l \left(W\left(\mathbf{x}; \mathbf{u}_r\right)\right) - I^r\left(\mathbf{u}_r\right) \right)^2 .$$

Error function computed at each pixel

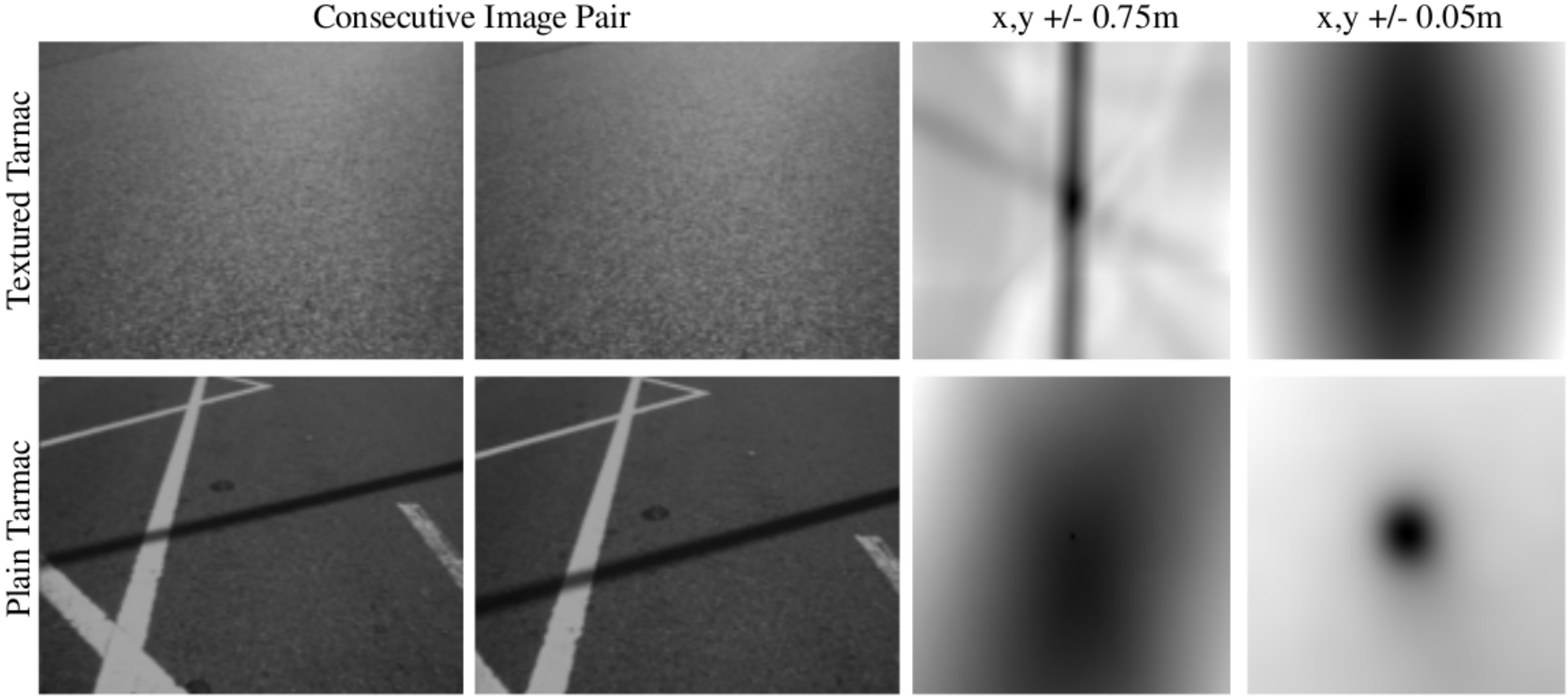
Reference Image I^*

$e(u, \mathbf{x})$



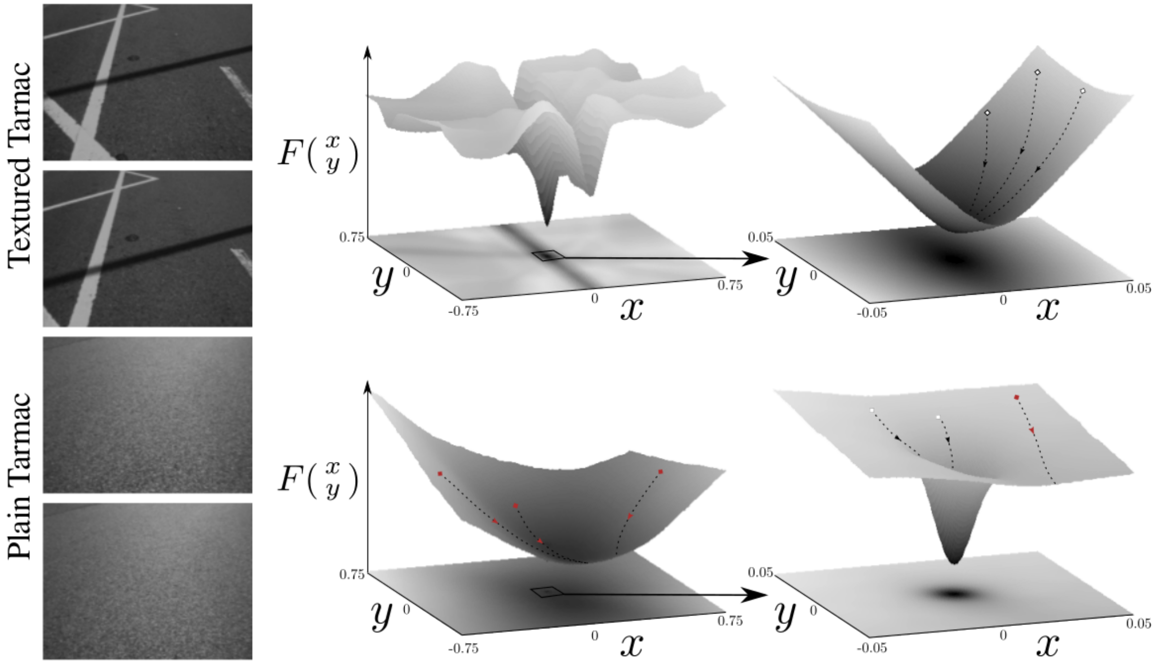
Live Observation I^g

Example Dense Frame to Frame Cost



Dense whole image alignment

3. We obtain the estimated alignment parameters **x** at the *minimum of* the photometric cost function: $\mathbf{x}^\circ = \arg\min_{\mathbf{x}\in\mathbb{R}^N} F(\mathbf{x})$



Whole Image Alignment: another simple example

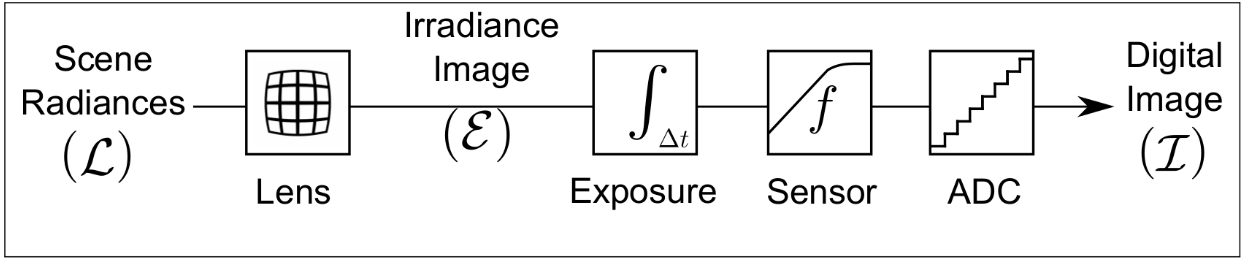
Image template:

Whole Image Alignment: another simple example

Live image with geometric warp:

Generative models beyond geometric

Warp function is for geometric error, but we can also think about modelling the rest of the image formation pipeline:



- ➔ Scene geometry, lighting structures and material reflectance properties, results in sample of the light ray for a given camera pose
- ➔ Geometric and radiometric distortion due to the camera lens, e.g. lens distortion, vignetting.
- ➔ Motion blur due to long exposures or image noise for short exposures and low lighting
- ➔ Nonlinear response of sensor for different exposures breaks brightness constancy assumption
- ➔ Feature descriptors enable sparse tracking techniques to become somewhat robust to these

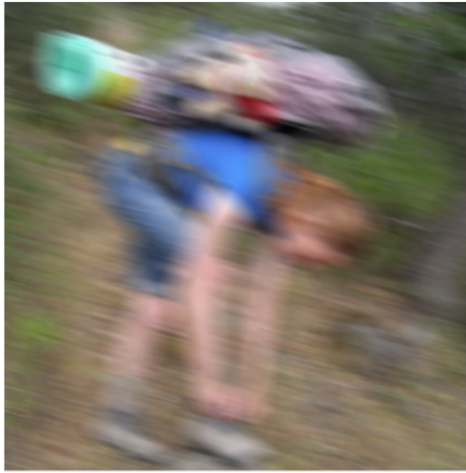
Example transformations: perils of feature matching

I.e. what if there is image degradation?

Reference Image

Geometric transformation and blur

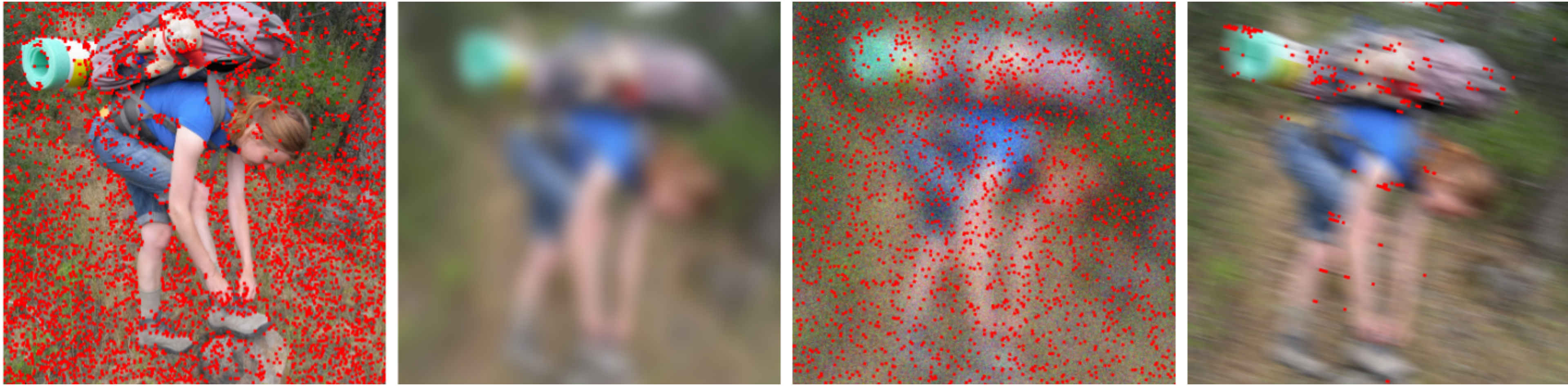
Geometric, blur and noise



Geometric, motion blur

Sparse pipelines need image features

Example FAST detections (Rosten and Drummond, ECCV 2006)



Reference Image

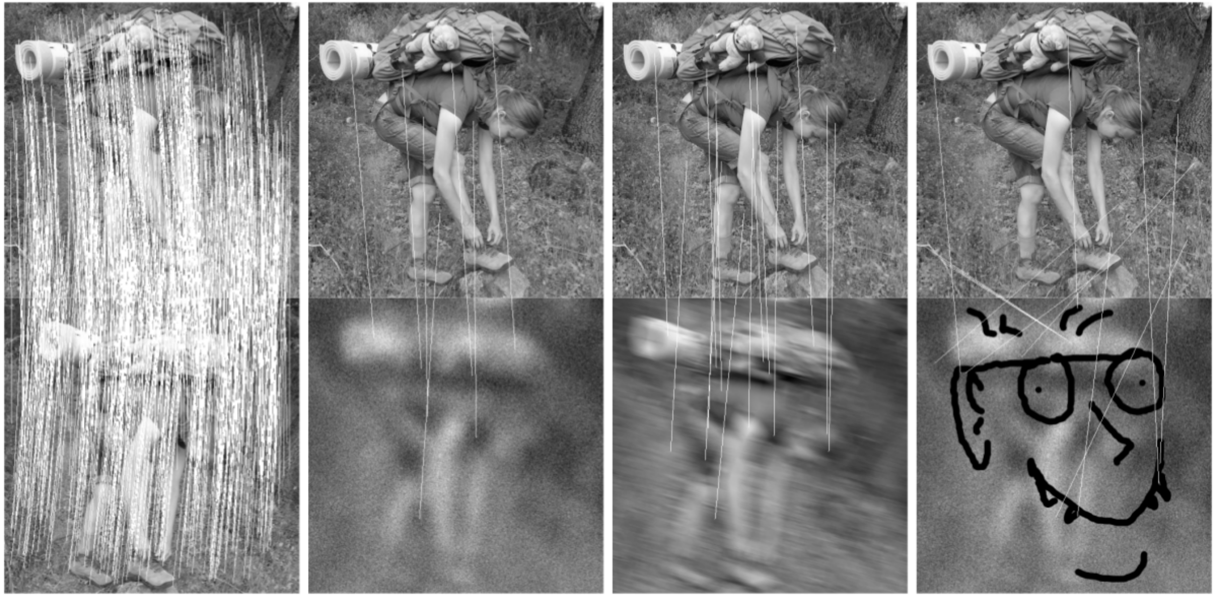
Geometric transformation and blur

Geometric, blur and noise

Geometric, motion blur

Sparse (1) extraction and (2) matching

Descriptor extraction and matching using naively applied SIFT (Lowe, ICCV 2004)



Geometric only

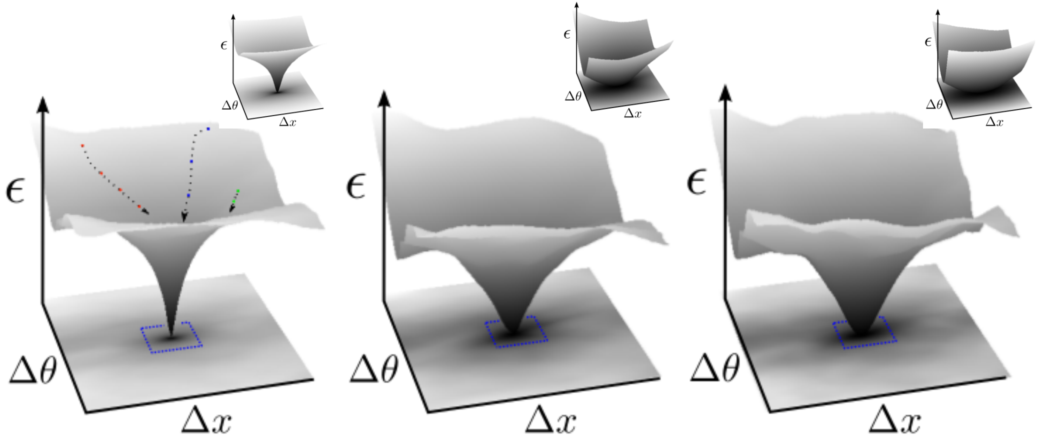
Geometric, blur and noise

Geometric, motion blur

Geometric, blur, noise, occlusion

Cost function using *dense* pixel errors

- ➔ Despite using a simple single pixel error term, there exists a clear global minimum
- ➔ However, there are local minima!



Geometric only

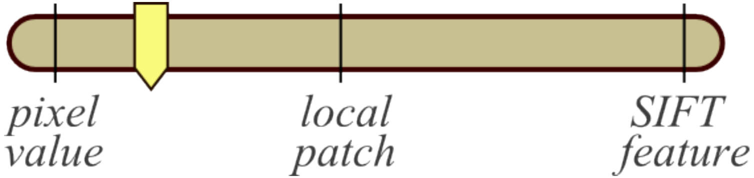
Geometric, blur, noise

Geometric, blur, noise, occlusion

Parameter range: $\Delta\theta \pm \pi/2$, $\Delta x \pm 100$ pixels

Dense and Sparse visual tracking

Generally we can trade off between complexity of the descriptor size and density of descriptor extraction to obtain a more robust error f:



- ➔ For whole image alignment, there is great redundancy for the few parameters being estimated, which can increase tracking robustness
- ➔ But gradient descent on the whole image cost function requires initialisation near to the global minimum (i.e. not for wide baseline)
- ➔ Many variations on how robustify against, or model photometric transformations

Basic Optimisation for Whole Image Alignment

Iterative Gauss-Newton Optimisation of the Dense cost function

Lucas-Kanade (1981)

Direct alignment for 2D image translation with warp function ***w***(u) = u+t, and with a quadratic penalty function:

$$\underset{t \in \mathrm{R}^2}{\operatorname{argmin}} \left\{ E(t) = \sum_{u \in \Omega} \left(\mathcal{I}_l(u+t) - \mathcal{I}_r(u) \right)^2 \right\} .$$

Direct Non-linear Optimisation

We want to estimate the unknown transform between two image frames by minimising a whole image error:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \{E_w(\mathbf{x})\} \ ,$$

$$E_w(\mathbf{x}) = \sum_{u \in \Omega} \psi\left(e(u, \mathbf{x})\right) \ .$$

Where is the chosen penalty, i.e. $\psi(e)=e^2$, and Ω is the image domain.

Direct Non-linear Optimisation

The image error, given the generative warp model is simply the per pixel difference *given* parameters ***x***:

$$e(u, \mathbf{x}) = \mathcal{I}_l(\mathbf{w}(u, \mathbf{x})) - \mathcal{I}_r(u)$$

We will use an Iterative Gauss-Newton Gradient descent on E_{w} to estimate the parameters **x**.

Taylor series expansion of $E_w(\mathbf{x}_0 + \Delta x)$

$$\tilde{E}_w(\mathbf{x}_0 + \Delta x) \approx E_w(\mathbf{x}_0) + \boldsymbol{\nabla}_{\mathbf{x}} E_w(\mathbf{x}_0)\Delta x + \tfrac{1}{2}\Delta x^\top \boldsymbol{\nabla}^2_{\mathbf{x}} E_w(\mathbf{x}_0)\Delta x \ ,$$

Solve convex form at Stationary Point:

$$\boldsymbol{\nabla}_{\Delta x}\tilde{E}_w = 0.$$

Gauss-Newton Approximation

Approximate the Hessian by truncating to the first order components:

$$\frac{1}{2}\sum_{u\in\Omega}\Delta x^{\top}J(u,\mathbf{x}_0)^{\top}J(u,\mathbf{x}_0)\Delta x\ ,$$

The result is an approximated 2nd order linearisation:

$$\tilde{E}_w(\mathbf{x}_0+\Delta x)=E_w(\mathbf{x}_0)+\sum_{u\in\Omega}\psi'(e(u,\mathbf{x}_0))J(u,\mathbf{x}_0)\Delta x+\frac{1}{2}\sum_{u\in\Omega}\Delta x^{\top}J(u,\mathbf{x}_0)^{\top}J(u,\mathbf{x}_0)\Delta x\ ,$$

Gradient of the cost function

Derivative of the penalty function:

$$\psi'(e(u,\mathbf{x}_0)) = \left.\frac{\partial \psi(e(u,\mathbf{x}))}{\partial e(u,\mathbf{x})}\right|_{\mathbf{x}_0},$$

i.e. $2e(u, \boldsymbol{x}_0)$ for $\psi(e(u,\boldsymbol{x}))=e(u,\boldsymbol{x})^2$

$$\tilde{E}_w(\mathbf{x}_0 + \Delta x) = E_w(\mathbf{x}_0) + \sum_{u\in\Omega} \psi'(e(u,\mathbf{x}_0)) J(u,\mathbf{x}_0) \Delta x + \frac{1}{2} \sum_{u\in\Omega} \Delta x^\top J(u,\mathbf{x}_0)^\top J(u,\mathbf{x}_0) \Delta x ,$$

Derivative of the observation prediction function:

$$J(u,\mathbf{x}_0) = \left.\frac{\partial \mathcal{I}_l(\mathbf{w}(u,\mathbf{x}))}{\partial \mathbf{x}}\right|_{\mathbf{x}_0}.$$

Examples to follow for SO3 RGB, SE3 RGB, RGB-D and Depth only camera tracking

Solve for the linearised Cost function

Remember, a minimising argument is achieved as a function extremum: $\nabla_{\Delta x} \tilde{E}_w = 0.$

Taking the derivative of the linearised cost function:

$$\cancel{E_w(\mathbf{x}_0)} + \sum_{u \in \Omega} \psi'(e(u, \mathbf{x}_0)) J(u, \mathbf{x}_0) \cancel{\Delta x} + \cancel{\tfrac{1}{2}} \sum_{u \in \Omega} \cancel{\Delta x}^\top J(u, \mathbf{x}_0)^\top J(u, \mathbf{x}_0) \Delta x \ ,$$

$$\sum_{u \in \Omega} \psi'(e(u, \mathbf{x}_0)) J(u, \mathbf{x}_0) \quad + \quad \sum_{u \in \Omega} J(u, \mathbf{x}_0)^\top J(u, \mathbf{x}_0) \Delta x$$

Solving for the incremental update

Resulting in the *normal equations*:

$$\sum_{u\in\Omega} J(u,\mathbf{x}_0)^\top J(u,\mathbf{x}_0)\Delta x = -\sum_{u\in\Omega} \psi'(e(u,\mathbf{x}_0))J(u,\mathbf{x}_0)\ ,$$

$$\Rightarrow \Delta x = -\Big(\sum_{u\in\Omega} J(u,\mathbf{x}_0)^\top J(u,\mathbf{x}_0)\Big)^{-1} \sum_{u\in\Omega} \psi'(e(u,\mathbf{x}_0))J(u,\mathbf{x}_0)\ .$$

The parameter vector is then updated:

$$\mathbf{x} \leftarrow \mathbf{x}_0 + \Delta x\ .$$

See **A13** for more details on trust region techniques for improved stability.

Basic Dense VO algorithm outline

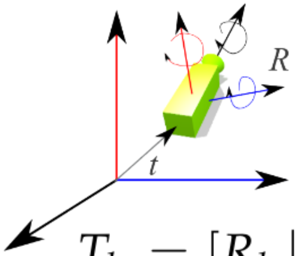
Input: relative transform estimate, a template.and a live frame.

output: updated relative transform estimate that warps the template into the live frame.

1. Compute dense cost function error and derivatives
2. Minimise dense cost function error by iterative Gauss-Newton minimisation.
3. Iterate until *convergence criteria*.

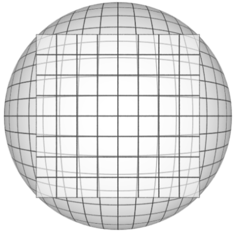
Incremental Camera Tracking

For RGB and
RGB-D Cameras



$$T_{lr} = [R_{lr} | t_{lr}] \in SE_3$$

Incremental Transformations



We can parameterise the relative camera motion between referen
live frames:

$$T_{lr} = [R_{lr} | t_{lr}] \in SE_3$$

A minimal parameterisation of a rigid body transform is given by:

$$\mathbf{x} = \begin{pmatrix} \omega \in \mathbb{R}^3 \\ v \in \mathbb{R}^3 \end{pmatrix} ,$$

Where the parameters define an element of the Lie Algebra as (see A13):

$$\hat{\mathbf{x}} = \begin{pmatrix} [\omega]_\times & v \\ 0 & 0 \end{pmatrix} \in \mathfrak{se}_3 \quad \text{where} \quad [\omega]_\times = \begin{pmatrix} 0 & -\omega_2 & \omega_1 \\ \omega_2 & 0 & -\omega_0 \\ -\omega_1 & \omega_0 & 0 \end{pmatrix} .$$

Incremental Transformations

The derivative of the non-linear exponential map that takes $[\omega]_x$ to the SO3 rotation matrix can be obtained by truncating to the linear term of the matrix exponential:

$$\exp([\omega]_\times) \mapsto I + [\omega]_\times + \frac{1}{2}[\omega]_\times^2 + \cdots + \frac{1}{k!}[\omega]_\times^k + \cdots$$

The linearisation of the exponential map to first order for ω around 0 is useful in practice, i.e. cos(θ)~1 and sin(θ)~0.

We will compose resulting incremental small SO3 (or SE3) transformations together via the exponential map:

$$T_{lr} = \exp(\hat{\mathbf{x}}^n) \exp(\hat{\mathbf{x}}^{n-1}) \ldots \exp(\hat{\mathbf{x}}^0) \tilde{T}_{lr}$$

A *rotating* RGB Camera

The transformation of a pixel from one frame into another is *independent* of the scene geometry if $t = (0\ 0\ 0)^T$:

$$u_l = \pi\left(K R_{lr} K^{-1} \dot{u}_r\right) .$$

Here $K^{-1}u_r$ defines a ray through pixel u_r and the camera center that is rotated and projected into the live frame.

Given an incremental compositional update to the rotation between the reference and live frames, the **warp function** is therefore:

$$\mathbf{w}_{SO_3}(u_r, \omega) = \pi\left(K \exp([\omega]_\times) \tilde{R}_{lr} K^{-1} \dot{u}_r\right) .$$

A *rotating* RGB Camera

$$E_w(\mathbf{x}) = \sum_{u \in \Omega} \psi\left(e(u, \mathbf{x})\right)$$

$$e(u, \mathbf{x}) = \mathcal{I}_l(\mathbf{w}(u, \mathbf{x})) - \mathcal{I}_r(u)$$

Whole Image Error: E

Inserting $\mathbf{w}_{SO3}$ into the whole image error we now perform the linearisation of $E_w(\mathbf{x_0} + \Delta)$ with $\Delta = \omega$, hence we compute the per pixel image **error derivative** as:

$$J(u, \omega) = \frac{\partial I_l(\mathbf{w}_{SO3})}{\partial \mathbf{w}_{SO3}} \frac{\partial \mathbf{w}_{SO3}(u, \omega)}{\partial K \exp([\omega]_\times) \tilde{R}_{lr} K^{-1} \dot{u}_r} \frac{\partial \mathrm{K} \exp([\omega]_\times) \tilde{R}_{lr} K^{-1} \dot{u}_r}{\partial \omega}$$

A *rotating* RGB Camera

Pre-computing the currently rotated ray

$$(x, y, z)^\top = \tilde{R}_{lr} K^{-1} u_r$$

The resulting **error gradient** vector for pixel *u* is:

$$J(u, \omega) = \begin{pmatrix} \nabla_x I_l \\ \nabla_y I_l \end{pmatrix}^\top \begin{pmatrix} \frac{f_x}{z} & 0 & -\frac{x f_x}{z^2} \\ 0 & \frac{f_y}{z} & -\frac{y f_y}{z^2} \end{pmatrix} \begin{pmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{pmatrix}.$$

A *rotating* RGB Camera

Evaluating the total Jacobian together with the chosen penalty function, we solve the resulting *normal equations:*

$$\Delta x = -\left(\sum_{u\in\Omega} J(u,\mathbf{x}_0)^\top J(u,\mathbf{x}_0)\right)^{-1} \sum_{u\in\Omega} \psi'(e(u,\mathbf{x}_0))J(u,\mathbf{x}_0)$$

Finally, form the SO3 matrix by exponentiation, and compose onto the initial transform:

$$\tilde{R}_{lr} \leftarrow \exp([\omega]_\times)\tilde{R}_{lr} \ .$$

Application: Real-time mosaicing, (Lovegrove & Davison, ECCV 2010)

General *rigid body* RGB-D tracking

When a depth map is also available in one frame, we can compute pixel transfer of points in one frame given the relative **SE3** transform T_{lr}:

$$u_l = \pi \left(K T_{lr} K^{-1} \mathcal{D}_r(u_r) \dot{u}_r \right)$$

Given an incremental compositional update to the rotation between the reference and live frames, the **warp function** is therefore:

$$\mathbf{w}_{SE_3}(u, \mathbf{x}) = \pi \left(K \exp(\hat{\mathbf{x}}) \tilde{T}_{lr} K^{-1} \mathcal{D}_r(u_r) \dot{u}_r \right)$$

General *rigid body* RGB-D tracking

Inserting $\mathbf{w}_{SE3}$ into the whole image error we now perform the **linearisation** of $E_w(\mathbf{x_0} + \Delta)$ with rigid body parameters $\Delta = \mathbf{x}$:

$$J(u, \mathbf{x}) = \frac{\partial I_l(\mathbf{w}_{SE_3})}{\partial \mathbf{w}_{SE_3}} \frac{\partial \mathbf{w}_{SE_3}(u, \mathbf{x})}{\partial K \exp(\hat{\mathbf{x}}) \tilde{T}_{lr} K^{-1} \mathcal{D}_r(u) \dot{u}_r} \frac{\partial K \exp(\hat{\mathbf{x}}) \tilde{T}_{lr} K^{-1} \mathcal{D}_r(u) \dot{u}_r}{\partial \mathbf{x}}$$

Pre-computing the currently transformed per pixel vertex:

$$(x, y, z)^\top = \tilde{T}_{lr} K^{-1} \mathcal{D}_r(u) \dot{u}_r$$

The resulting image **error gradient** vector for pixel *u* is*:*

$$J(u, \mathbf{x}) = \begin{pmatrix} \nabla_x I_l \\ \nabla_y I_l \end{pmatrix}^\top \begin{pmatrix} \frac{f_x}{z} & 0 & -\frac{x f_x}{z^2} \\ 0 & \frac{f_y}{z} & -\frac{y f_y}{z^2} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & z & -y \\ 0 & 1 & 0 & -z & 0 & x \\ 0 & 0 & 1 & y & -x & 0 \end{pmatrix}$$

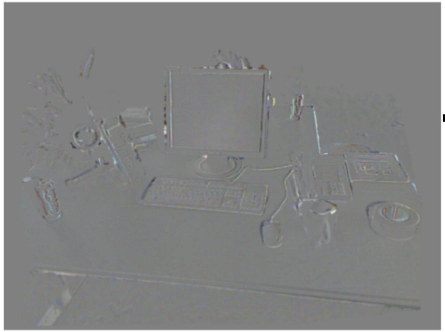
Solve normal equations and compose: $\tilde{T}_{lr} \leftarrow \exp(\hat{\mathbf{x}}) \tilde{T}_{lr}$

Example Dense Pixel Transfer

(a) First input image

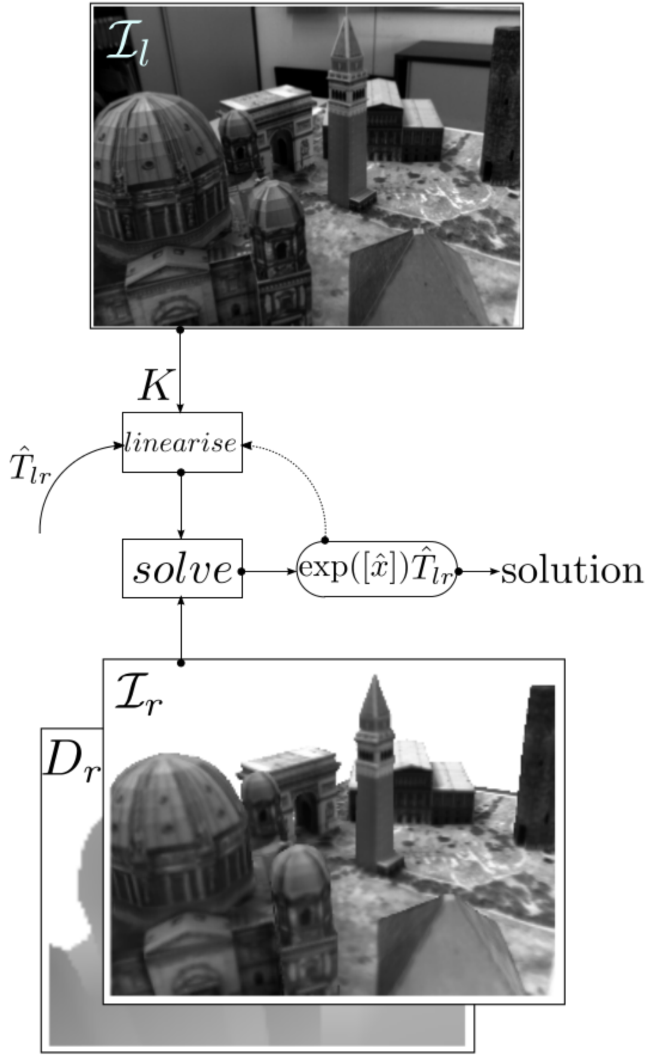
(b) Second input image

(c) Warped second image



(d) Difference image

- ➔ Note: we can use rendering engine (e.g. OpenGL) to achieve the observation prediction.
- ➔ Requires a triangle mesh representation of the depth map.
- ➔ Can correctly predict self occlusion since it is a surface.



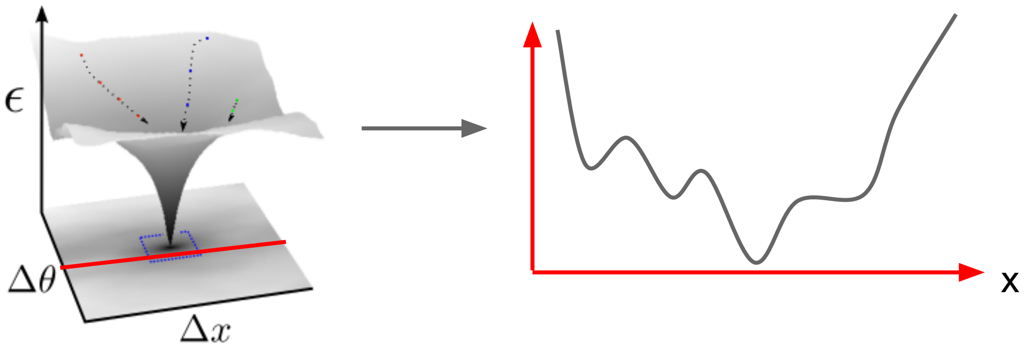
The linearisation assumption:

$$\mathcal{I}_l(\mathbf{w}(u,\Delta\mathbf{x})) \approx \mathcal{I}_l(\mathbf{w}(u,\mathbf{0})) + J(\mathbf{0})\Delta\mathbf{x} \ ,$$

$$\Rightarrow \mathcal{I}_r(u) - \mathcal{I}_l(\mathbf{w}(u,\mathbf{0})) \approx J(\mathbf{0})\Delta\mathbf{x} \ .$$

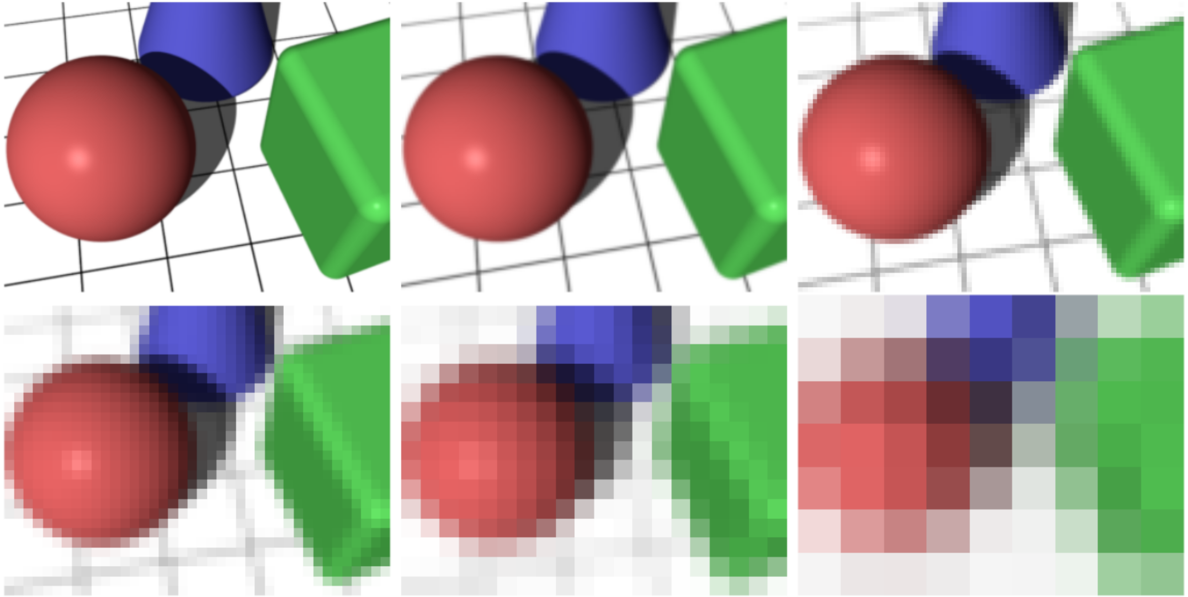
Coarse to fine optimisation

The linearisation assumption is easily broken in real images, as the transformation magnitude increases, the cost function becomes clearly non-convex.

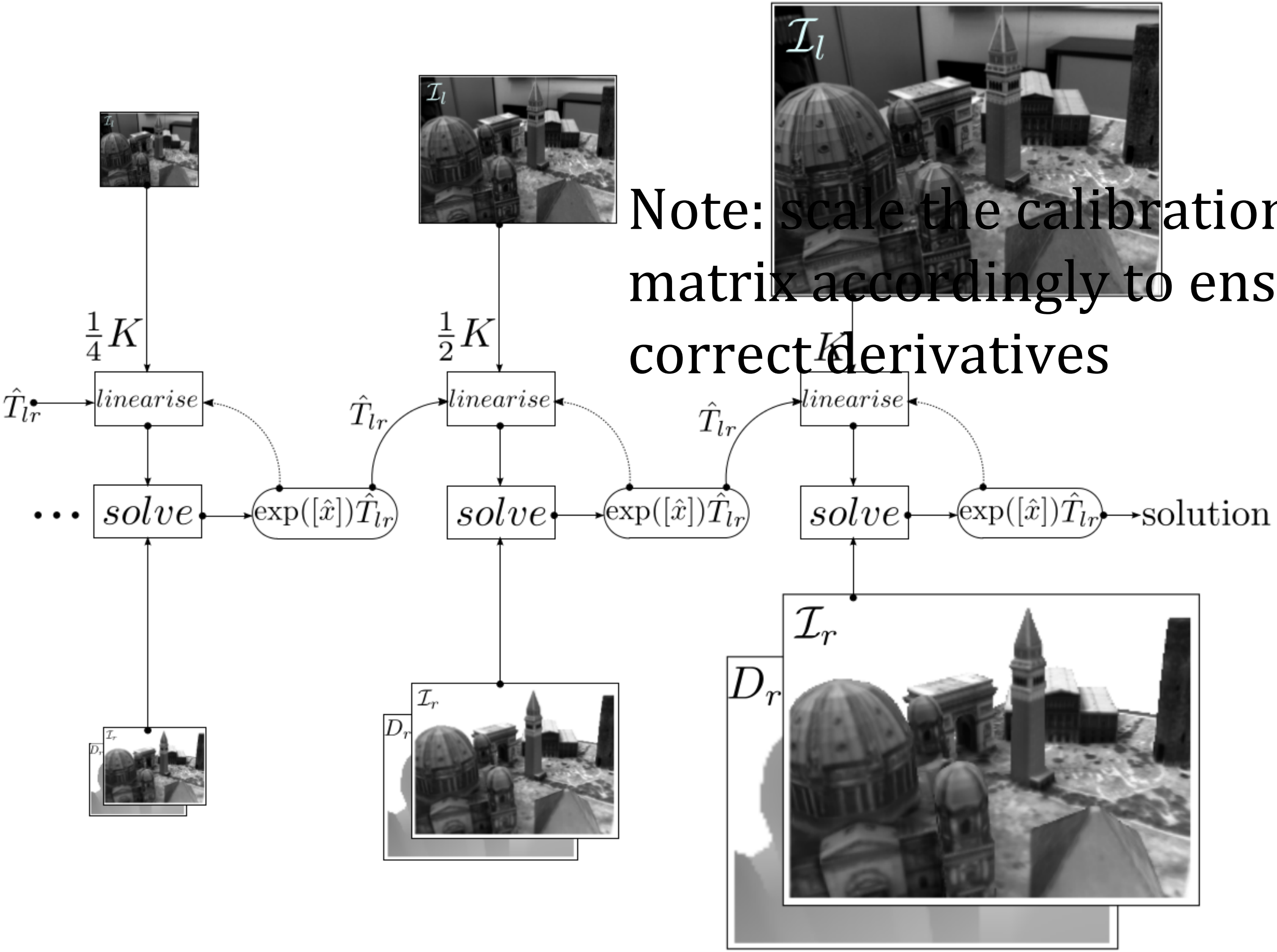


Coarse to fine optimisation: downsampling

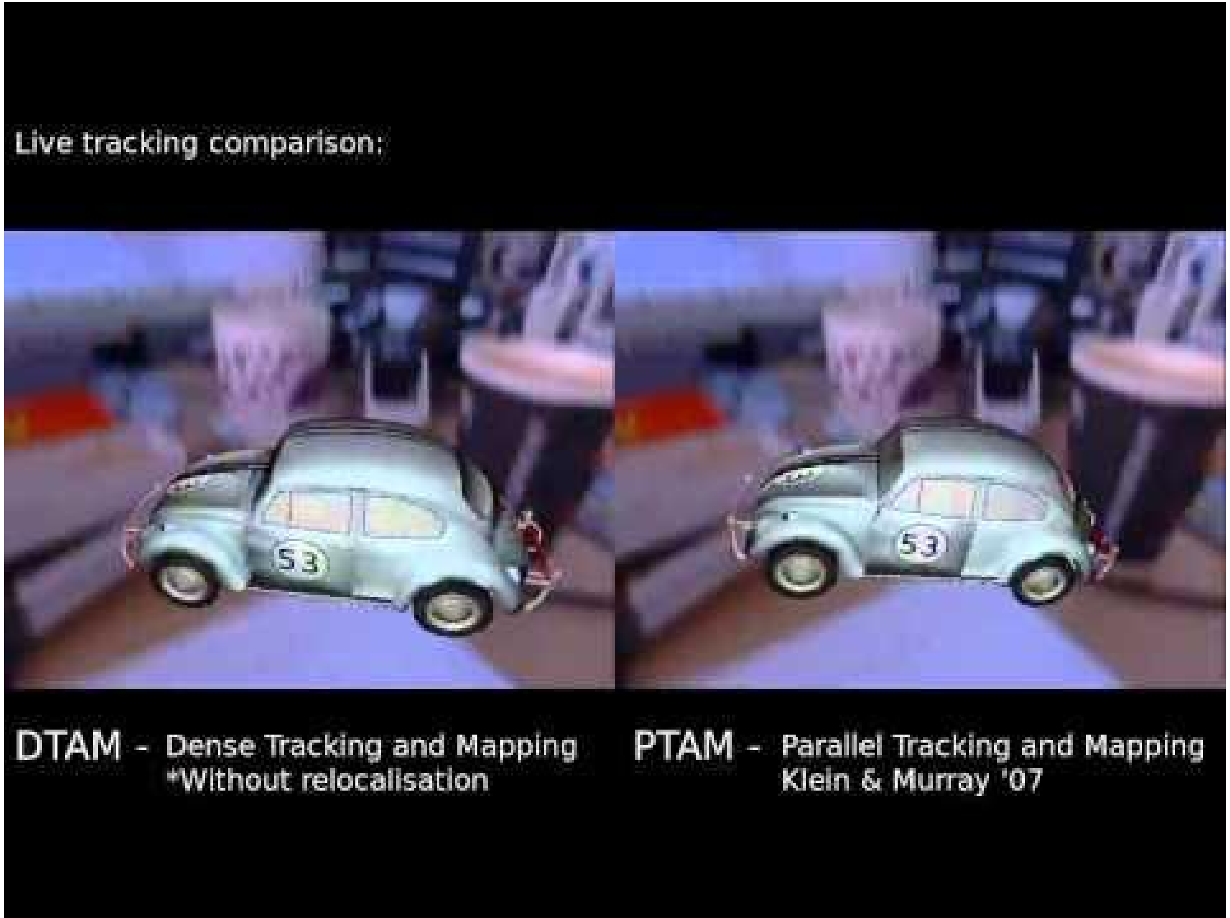
Removing higher frequency components in the images increases the parameter range for which the linearisation holds.



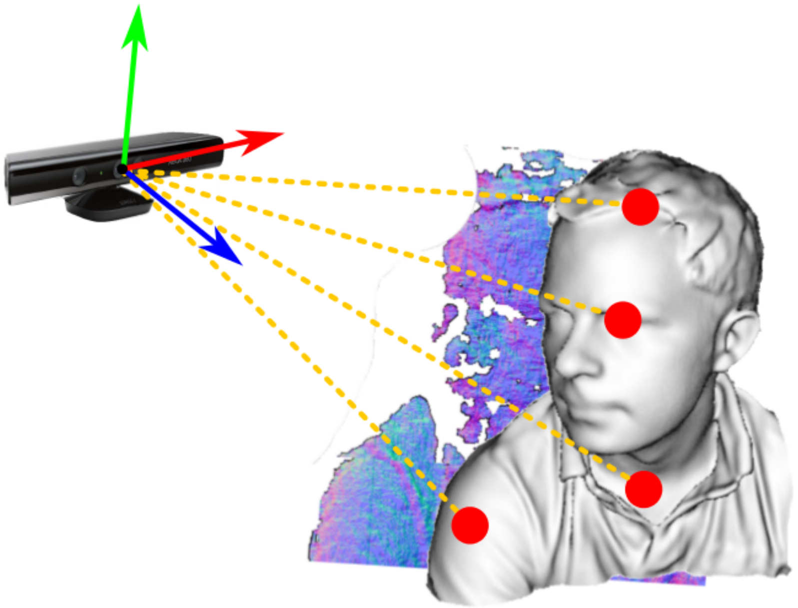
Note: scale the calibration matrix accordingly to ensure correct derivatives



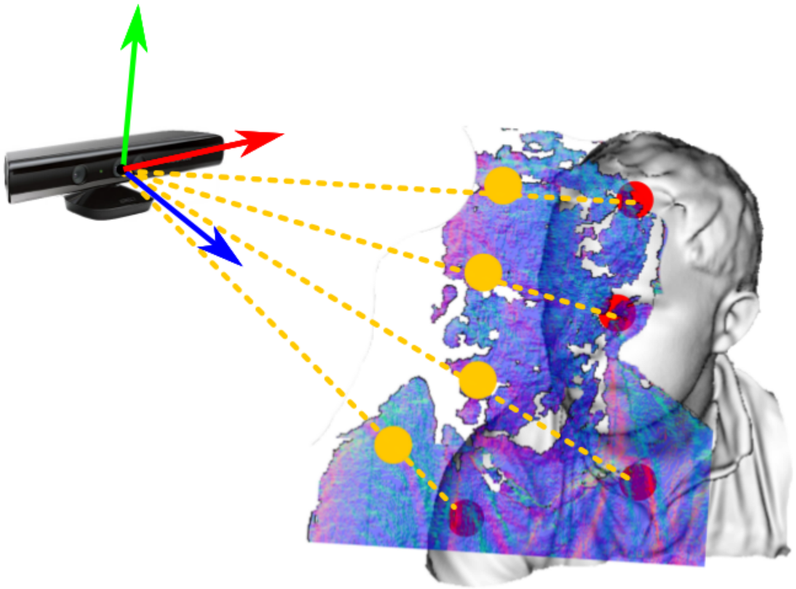
Single RGB dense visual odometry from a keyframe (Newcombe et al, ICCV 2011)



General *rigid body* depth tracking (ICP)

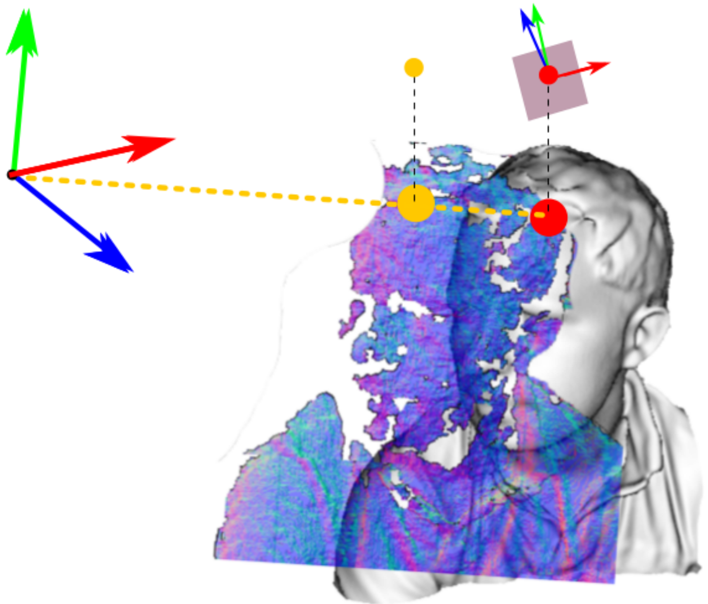


(a) The model (grey) is rendered into the estimated frame. We can sample points from this model in image space (red dots).

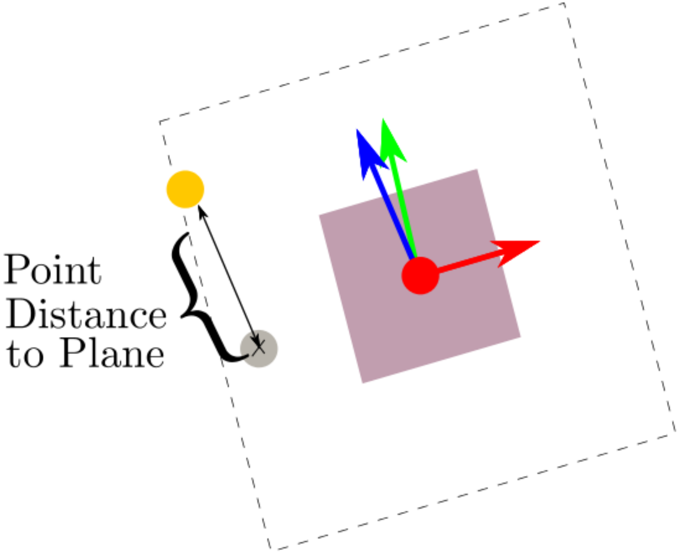


(b) Projective data-association with the live frame: Corresponding are selected by pairing points which lie on the same ray (red-yellow dots).

General *rigid body* depth tracking (ICP)

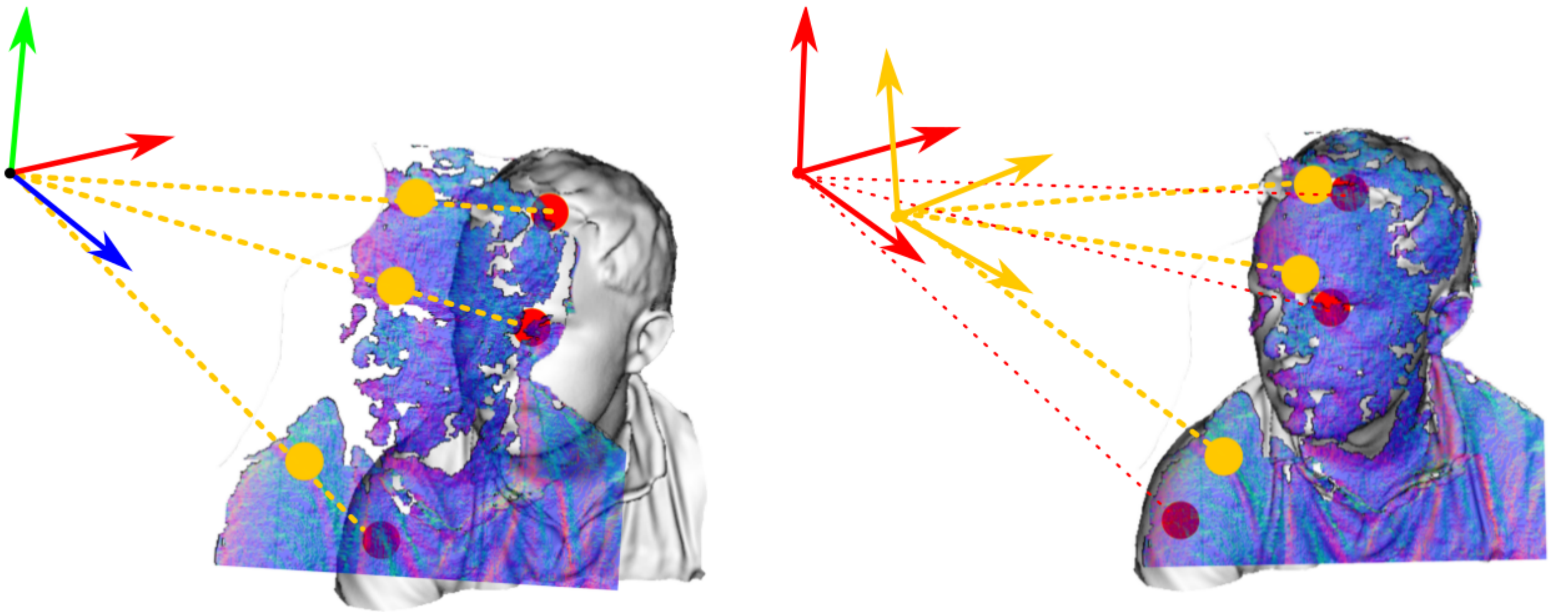


(c) Each pair, has a point-plane constraint: the surface normal estimated from the model provides the normal since it is higher quality.



(d) Each point-plane constraint provides an error measure as the shortest distance of the live image point to the tangent plane of the corresponding model point.

General *rigid body* depth tracking (ICP)



(e) Pairs fail a point-plane compatibility if the point-point Euclidean distance or normal-normal angle exceed thresholds.

(f) A Gauss-Newton based iterative gradient descent minimisation of the sum of point-plane error induced by the remaining pairs results in the new pose estimate.

Whole image depth image tracking (dense ICP)

Given 2 depth images, we define a generative model over the vertex maps:

$$v_r(u) = K^{-1}\dot{u}\mathcal{D}_r(u) \ ,$$

Warp the surface in the reference image into the live image given the relative SE3 transform:

$$v_l(u') = \tilde{T}_{rl}K^{-1}\dot{u}'\mathcal{D}_l(u') \ ,$$

$$u' = \mathbf{w}_{se3}(u, \mathbf{x}_0) = \pi(K\tilde{T}_{lr}v_r) \ .$$

We can use the per depth pixel point-plane error, instead of a euclidean distance of the vertices:

$$e(u, \mathbf{x}) = N_r^\top(u)(\exp(\hat{\mathbf{x}})v_l(u') - v_r(u)) \ ,$$

Whole image depth image tracking (dense ICP)

Plugging the point-plane error into the whole image cost function, we again perform **linearisation** of $E_w(\mathbf{x_0} + \Delta)$ with rigid body parameters $\Delta = \mathbf{x}$. Pre-computing the currently transformed per pixel vertex:

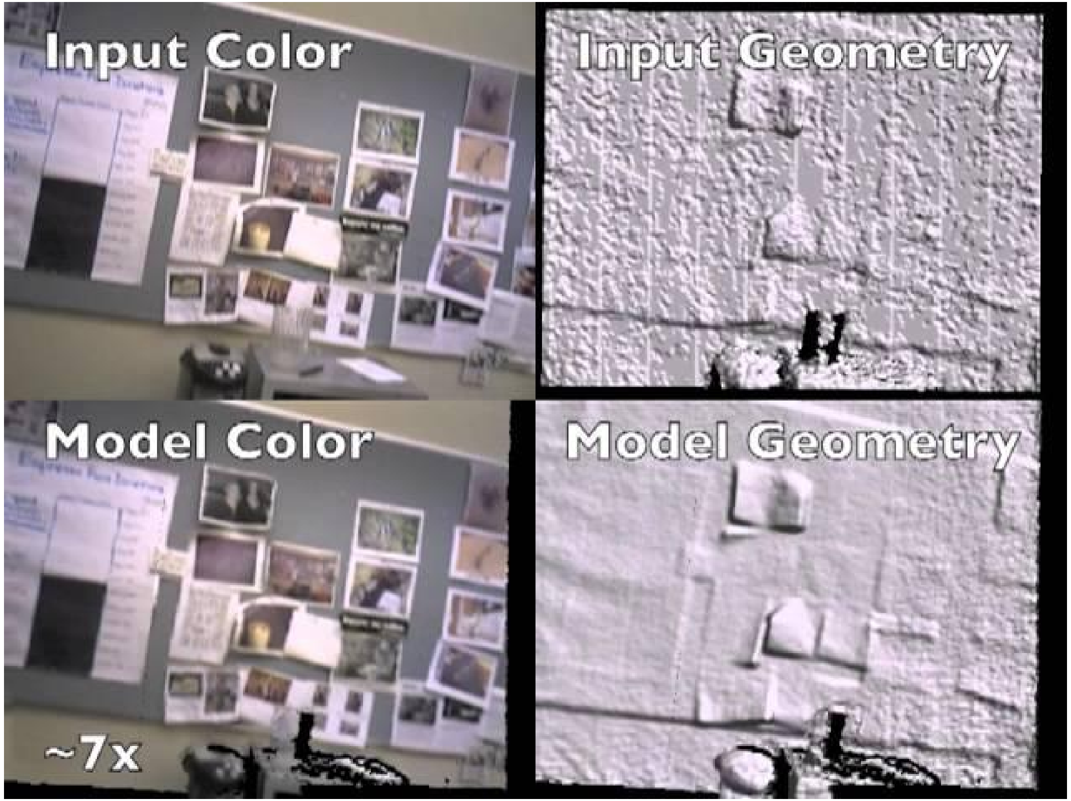
$$(x, y, z)^\top = \tilde{T}_{rl} v_l(\mathbf{w}(u, \mathbf{x}))$$

The resulting image **error gradient** vector for pixel u is:

$$J(u, \mathbf{x}) = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}^\top \begin{pmatrix} 1 & 0 & 0 & 0 & z & -y \\ 0 & 1 & 0 & -z & 0 & x \\ 0 & 0 & 1 & y & -x & 0 \end{pmatrix},$$

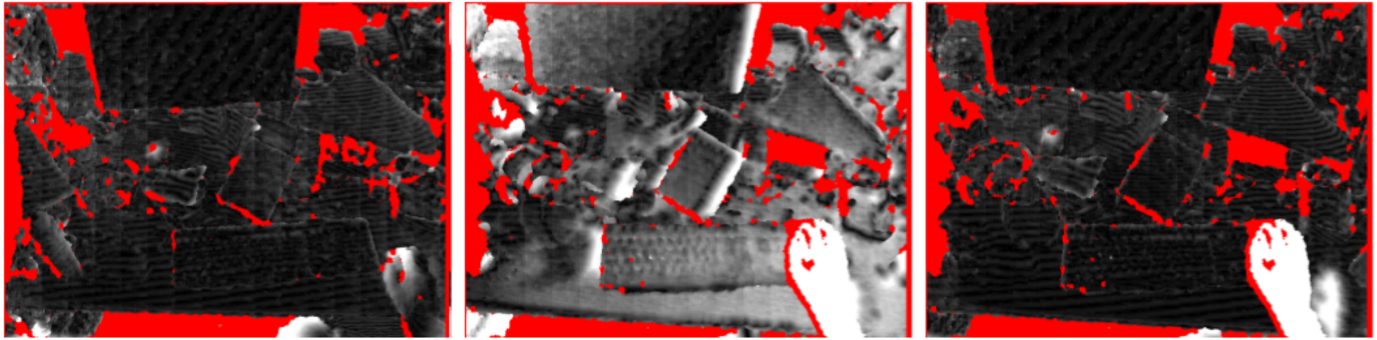
Solve normal equations and compose: $\tilde{T}_{lr} \leftarrow \exp(\hat{\mathbf{x}}) \tilde{T}_{lr}$.

Example Application: RGB-D + ICP Tracking (Henry et al, 3DV 2013)



Basic robustness to a generative models outliers

Example dense ICP errors before/after outliers are introduced:



(b) Error prior to outliers. (c) ψ as quadratic penalty. (d) ψ as Huber penalty.

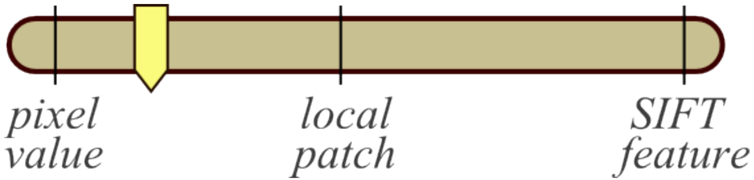
With example known **x***, we choose the penalty function ψ to closely match -log of probability distribution over pixel errors: $\mathbf{P}(e(\mathrm{u},\mathbf{x}^*))$

Robustified tracking

Per-pixel gating using photometric error between the predicted and live images

Conclusions: Dense visual tracking

Remember, we can trade off between complexity of the descriptor size and density of descriptor extraction:



- ➔ However, dense tracking formulations are **trivially parallelisable**
- ➔ We can make use of all image data to mitigate issues with where to extract and match features: can increase robustness
- ➔ As frame-rate increases, computational requirements reduce

Thanks! Questions?