Computer Vision Lecture 5

Tracking: 2D and 3D

David Murray and Victor Adrian Prisacariu

david.murray@eng.ox.ac.uk and victor@robots.ox.ac.uk
www.robots.ox.ac.uk/~dwm/Courses/Aims
(not yet up) www.robots.ox.ac.uk/~victor/Courses/Aims

2016-17
Visual tracking

Visual tracking involves the identification of some characteristic of the scene in successive images.

One immediately useful division to make is between processes for

2D tracking, where the aim is to follow and perhaps control the image position of some entity as it moves from frame to frame over time.

and those for

3D or Pose tracking, where the aim is to use image measurements (possibly involving 2D tracking) to update the 6 degrees of freedom (3 translation + 3 rotation) which define 3D pose.
5.1 2D Tracking

5.2 3D Visual Pose tracking for object pose

5.3 3D RGB-D Pose tracking for object pose

5.4 3D Visual Pose tracking for camera pose

5.5 A note on image and scene dynamics
5.1 2D Visual tracking
5.1 2D Visual tracking

Both 2D and 3D tracking require some sort of model of appearance — one sufficient to identify high cross-correlation between frames.

In **2D tracking**, the appearance model might relate to

- a single point
- a small patch, possibly deformable
- a contour, possibly deformable
- a line element
- ...?

Let’s start simply by tracking an image point ...
5.1.1 Points

... using a bright spot detector

This is fun, but hopelessly non-robust!

We need to incorporate

- Stronger appearance/measurement models; and
- Image and/or Scene dynamics
5.1.2 Patches: Lucas-Kanade template tracker

Using more pixels is good ...

We seek the image position \( \mathbf{p} = [t_x, t_y]^T \) which maximises the similarity to the template

Exhaustive search possible but undesirable

Better to do gradient ascent/descent \( \Rightarrow \) the Lucas-Kanade tracker
5.1.2 Lucas-Kanade tracker: pure translation

Formulate search as an optimisation problem using brightness constancy as our objective function.

Non-convex function over the image — so we need to start near solution.

Suppose the starting point is \([t_x, t_y]^T\) ...

LK minimises the Sum-of-Squared-Differences (SSD) \(\text{wrt } \Delta t_x, \Delta t_y:\)

\[
E(\Delta t_x, \Delta t_y) = \sum_{x, y \in T} [l(x + t_x + \Delta t_x, y + t_y + \Delta t_y) - T(x, y)]^2
\]
5.1.2 Lucas-Kanade tracker: pure translation

Expanding to 1st order:

\[
E(\Delta t_x, \Delta t_y) \approx \sum_{x,y \in T} \left[ I(x + t_x, y + t_y) + \nabla I^\top \left[ \begin{array}{c} \Delta t_x \\ \Delta t_y \end{array} \right] - T(x, y) \right]^2
\]

Setting \( \partial E / \partial \Delta t_x \) and \( \partial E / \partial \Delta t_y \) to zero gives the vector equation

\[
2 \sum_{x,y \in T} \nabla I \left[ I(x + t_x, y + t_y) + \nabla I^\top \left[ \begin{array}{c} \Delta t_x \\ \Delta t_y \end{array} \right] - T(x, y) \right] = \left[ \begin{array}{c} 0 \\ 0 \end{array} \right]
\]

\[
\Rightarrow \left[ \sum_{x,y \in T} \nabla I \nabla I^\top \right]_{2 \times 2} \left[ \begin{array}{c} \Delta t_x \\ \Delta t_y \end{array} \right] = - \left[ \sum_{x,y \in T} \nabla I \left[ I(x + t_x, y + t_y) - T(x, y) \right] \right]_{2 \times 1}
\]

(There is great similarity to the optic flow solution here)

Set \( t_x \leftarrow t_x + \Delta t_x \), \( t_y \leftarrow t_y + \Delta t_y \) and iterate until change negligible.
5.1.2 Lucas-Kanade tracker: generalisation

Pure translation can be overly restrictive — but the algorithm generalises nicely ...

Recall that $x$ represents a location in our template.

We now seek the values of a set of parameters $p$ such that a chosen warp function $f_p(x)$ best aligns the template with the current image $I$. 
## 5.1.2 Lucas-Kanade tracker: generalisation

The function $f$ can be quite general. Typical low order examples are:

<table>
<thead>
<tr>
<th>Translation</th>
<th>$p = [t_x, t_y]^\top$</th>
<th>$f_p(x) = \begin{bmatrix} 1 &amp; 0 &amp; t_x \ 0 &amp; 1 &amp; t_y \ 0 &amp; 0 &amp; 1 \end{bmatrix} \begin{bmatrix} x \ y \ 1 \end{bmatrix}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trans+rottn</td>
<td>$p = [\theta, t_x, t_y]^\top$</td>
<td>$f_p(x) = \begin{bmatrix} \cos \theta &amp; -\sin \theta &amp; t_x \ \sin \theta &amp; \cos \theta &amp; t_y \ 0 &amp; 0 &amp; 1 \end{bmatrix} \begin{bmatrix} x \ y \ 1 \end{bmatrix}$</td>
</tr>
<tr>
<td>Trans + scale</td>
<td>$p = [s, t_x, t_y]^\top$</td>
<td>$f_p(x) = \begin{bmatrix} s &amp; 0 &amp; t_x \ 0 &amp; s &amp; t_y \ 0 &amp; 0 &amp; 1 \end{bmatrix} \begin{bmatrix} x \ y \ 1 \end{bmatrix}$</td>
</tr>
<tr>
<td>General affinity</td>
<td>$p = [p_{11} \ldots p_{23}]$</td>
<td>$f_p(x) = \begin{bmatrix} p_{11} &amp; p_{12} &amp; p_{13} \ p_{21} &amp; p_{22} &amp; p_{23} \ 0 &amp; 0 &amp; 1 \end{bmatrix} \begin{bmatrix} x \ y \ 1 \end{bmatrix}$</td>
</tr>
</tbody>
</table>
5.1.2 Lucas-Kanade tracker: generalisation

Returning to the brightness constraint equation, we need to minimize with respect to $\Delta \mathbf{p}$:

$$E = \sum_{\mathbf{x} \in T} \left[ I(\mathbf{f}_p + \Delta \mathbf{p}(\mathbf{x}), t) - T(\mathbf{x}) \right]^2$$

Expanding to first order we obtain:

$$E \approx \sum_{\mathbf{x} \in T} \left[ I(\mathbf{f}_p(\mathbf{x}), t) + \nabla I^\top \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \Delta \mathbf{p} - T(\mathbf{x}) \right]^2$$

Taking partials with respect to $\Delta \mathbf{p}$ yields:

$$2 \sum_{\mathbf{x} \in T} \left[ \nabla I^\top \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \right]^\top \left[ \nabla I^\top \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \Delta \mathbf{p} + I(\mathbf{f}_p(\mathbf{x}), t) - T(\mathbf{x}) \right] = 0 \#\text{params} \times 1$$

Hence

$$\sum_{\mathbf{x} \in T} \left[ \nabla I^\top \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \right]^\top \left[ \nabla I^\top \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \right] \Delta \mathbf{p} = \sum_{\mathbf{x} \in T} \left[ \nabla I^\top \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \right]^\top \left[ T(\mathbf{x}) - I(\mathbf{f}_p(\mathbf{x}), t) \right]$$
5.1.2 Lucas-Kanade tracker: generalisation /ctd

\[ \sum_{x \in T} \left[ \nabla I^\top \frac{\partial f}{\partial p} \right]^\top \left[ \nabla I^\top \frac{\partial f}{\partial p} \right] \Delta p = \sum_{x \in T} \left[ \nabla I^\top \frac{\partial f}{\partial p} \right]^\top \left[ T(x) - I(f_p(x), t) \right] \]

\[ M \Delta p = b \]

Note that the gradients \( \nabla I \) are computed at \( f_p(x) \)
5.1.2 Lucas-Kanade tracker: Summary

Given template $T$, image $I(t)$ and current tracker parameters $p(t - 1)$:

1. $p \leftarrow p(t - 1)$
2. Repeat until $\Delta p \approx 0$:

$$M = \sum_{x \in T} \left[ \nabla I^T \frac{\partial f}{\partial p} \right]^T \left[ \nabla I^T \frac{\partial f}{\partial p} \right]$$

$$b = \sum_{x \in T} \left[ \nabla I^T \frac{\partial f}{\partial p} \right]^T \left[ T(x) - I(f_p(x), t) \right]$$

$$\Delta p = M^{-1}b$$

$$f_p() = f_{\Delta p} \circ f_p() \quad [p \leftarrow p + \Delta p]$$

3. $p(t) \leftarrow p$
5.1.2 Lucas-Kanade tracker: Observations

- The expressions look nastier than they are. For example, go back to the translational case:

\[ f_p(x) = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix} \quad \frac{\partial f}{\partial p} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \]

\[ f_{\Delta p}(x) \circ f_p(x) = \begin{bmatrix} 1 & 0 & \Delta t_x \\ 0 & 1 & \Delta t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x + \Delta t_x \\ 0 & 1 & t_y + \Delta t_y \\ 0 & 0 & 1 \end{bmatrix} \]

- We are actually doing a lot more work than we need to calculating gradients in the image at each iteration.

Instead pre-compute template gradients etc, and warp “the other way”

- Lucas-Kanade imposes a model of expected spatial transformations between template and observed image.
5.1.2 /ctd

A general difficulty with trackers relying too heavily on the spatial relationships between pixels is that they are prone to break due to partial occlusion and orientation changes in the scene.

The LK template tracker and similar try to get round the problem of appearance change by updating the template from frame to frame.

This works well enough, but only if the segmentation between tracked object and background is unequivocal. Suppose we were tracking someone’s head front on. As the head turned and the template updated it is likely that pixels from the background would be incorporated and this gradually drags the template off the head.
5.1.3 Patches: Histogram tracking by Mean Shift

One way of begin much more relaxed about predictable deformations is use a model built from the spatial template’s colour histogram \( \mathbf{q} \)

1. For each pixel in template patch
   1. Find appropriate bin in histogram
   2. Add 1 to it
2. Normalize: divide each bin by number of pixels

Compute similarity between template \( \mathbf{q} \) & image histogram \( \mathbf{p}(\mathbf{x}) \) at \( \mathbf{x} \) using the \textbf{Bhattacharyya coeff}:

\[
\rho(\mathbf{p}(\mathbf{x}), \mathbf{q}) = \int p_u(x)q_u du = \sum_u \sqrt{p_u(x)q_u} \quad 0 \leq \rho \leq 1
\]

where in the summation \( u \) denotes a distinct colour.
The template region is defined by an ellipse centred at \( o = (o_x, o_y) \) and aligned with the \( x \)– and \( y \)–axes with sizes \( h_x \) and \( h_y \).

More weight is given to the colours of pixels closer to the centre. This extra weight is

\[
k_E(r^2_i) = \frac{2}{\pi} (1 - r^2_i) \text{if } r_i < 1 \text{ else } 0
\]

\[
r^2_i = \left( \frac{(x_i - o_x)}{h_x} \right)^2 + \left( \frac{(y_i - o_y)}{h_y} \right)^2
\]

The weighting therefore is quite flat close to the centre (as shown).

For each colour \( u \), the value stored in template histogram centred at \( c \) is

\[
q_u = \frac{\sum_{i=1}^{n} k_E(r^2_i) \delta[U(x_i) - u]}{\sum_{i=1}^{n} k_E(r^2_i)}
\]

Here function \( U(x_i) \) just returns the colour bin for the pixel at \( x_i \), and

\[
\delta[U - u] = \begin{cases} 
1 & \text{if } U = u \\
0 & \text{otherwise}
\end{cases}
\]

The pdf for the target is derived identically.
5.1.3 Mean-shift tracking

Tracking requires us to move to

$$x^* = \max_x [\rho(p(x), q)]$$

We could consider

*gradient ascent* — turns out not to be so amenable to analytic approach

*naive numerical approach* — estimate $\partial \rho(x)/\partial x$, $\partial \rho(x)/\partial y$ by shifting a small amount and reevaluating

$$\frac{\partial \rho(x)}{\partial x} \approx \frac{[\rho(x + [\delta x, 0]^\top) - \rho(x)]}{\delta x}$$

$$\frac{\partial \rho(x)}{\partial y} \approx \frac{[\rho(x + [0, \delta y]^\top) - \rho(x)]}{\delta y}$$

But this is computationally costly!

Instead, clever idea is to use *mean-shift* to move in the direction of the gradient...

Real-time tracking of non-rigid objects using mean shift, Comaniciu, Ramesh and Meer, Proc Computer Vision and Pattern Recognition, 2000
5.1.3 Mean-shift tracking

What is mean-shift?

For any density function, the mean of a set of samples taken about $x$ will be biased towards a local mode (i.e. local maximum).

The mean-shift vector $\bar{x} - x$ points in the direction of the gradient.
5.1.3 Mean-shift tracking

Back to images: the density to maximise is the Bhattacharyya coefficient

\[ \rho(p(x), q) \]

Compute \( p(x) \), the colour histogram at the current location \( x \)
Each pixel \( i \) in the window around \( x \) provides a *weighted* sample. Suppose its colour is in bin \( u \) of the histogram, then the weight is given by

\[ w_i = \sqrt{q_u/p_u} \]

i.e. the square root of the “likelihood ratio”
Compute the mean shift:

\[ m(x) = \frac{\sum_i w_i x_i}{\sum_i w_i} - x \]

Update current position, and iterate

\[ x \leftarrow x + m(x) \quad \text{or just} \quad x \leftarrow \frac{\sum_i w_i x_i}{\sum_i w_i} \]
5.1.3 Mean-shift tracking — Observations

Quantity $\frac{\sum_i w_i x_i}{\sum_i w_i}$ is biased towards a mode: colours are more likely to “match” in correct direction – hence the weights will on average be greater closer to the true location.

- The weights are crucial, since otherwise the samples are simply at pixel locations and therefore uniformly distributed.

- Ridiculously simple to implement, very fast and effective.

- Can be applied generally to pdfs, not just to colour histograms.
5.1.4 Active Contour tracking — snakes

Patch tracking has used descriptors based on “interior” properties.

But one can also track using *change* in such properties.

Leads to the idea of edge, line and contour tracking.

We’ll mention line tracking for 3D tracking later.

In 2D we now consider **active contours** or **snakes**.

They exploit the fact that object boundaries tend to be continuous, smooth and move with the object.

One can constrain them with the physics of shape in the specific context, and then let the constraints and the image data compete ...
5.1.4 Contour tracking: snakes

A snake is a deformable contour $r(s)$ parameterised by arc-length $s$ (and possibly by time, $r(s, t)$) that minimises an energy equation

$$\mathcal{E}(r) = \mathcal{E}_{\text{int}}(r) + \mathcal{E}_{\text{ext}}(r)$$

where competing are

- the internal energy which controls the “material” properties of the contour — terms for membrane (stretch) behaviour and thin-plate (curvature) behaviour
- and the external energy derived from the image data (and possible from user input)

For example, if $\mathcal{E}_{\text{ext}}$ involves $-|\nabla I(x, y)|$ the snake will be attracted to edges in the image.

Do it for successive frames and you have a contour tracker...
5.1.4 Contour tracking: snakes

Notes:

- used splines and control points to define the contour
- added terms for inertia $\rho \frac{\partial^2 r}{\partial t^2}$, or damping $\gamma \frac{\partial r}{\partial t}$ to control temporal behaviour
5.1.4 Contour Tracking: Examples

Benedicte as herself

Benedicte as a cat

5.1.5 Regions — Level Sets

Objects change appearance when they move so their outline might break-up or merge.

Parametrisable contours are horrible at modeling this.

**Solution**: model contours implicitly
... as zero levels of embedding functions
... implemented using the signed distance transform / chamfer distance.
5.1.5 Regions — Level Sets

For any point $x$ in an image, the signed distance function / chamfer distance $\Phi$ is the shortest (signed) distance from $x$ to the outline of the object in the image.
5.1.5 Regions — Level Sets

This is used in region-based energy functions for segmentation and tracking:

$$E = \int_{\omega_f} r_f(x, C) d\Omega + \int_{\omega_b} r_b(x, C) d\Omega$$
5.1.5 Regions — Level Sets

This is used in region-based energy functions for segmentation and tracking:

Explicit contour: \( E = \int_{\omega_f} r_f(x, C) d\Omega + \int_{\omega_b} r_b(x, C) d\Omega \)

SDF: \( E = \int_{\omega} r_f(x) H_e(\Phi) + (1 - H_e(\Phi)) r_b(x) \)
5.1.5 Regions — Level Sets

This is used in region-based energy functions for segmentation and tracking:

\[ E = \int_\omega r_f(x)H_e(\Phi) + (1 - H_e(\Phi))r_b(x) \]

Segmentation \( \frac{\partial \Phi}{\partial t} \) and tracking \( \frac{\partial \Phi}{\partial \text{pose}} \) + any nonlinear optimisation.
5.1.5 Regions — Level Sets

This is used in region-based energy functions for segmentation and tracking:

SDF: \( E = \int_{\omega} r_f(x) H_e(\Phi) + (1 - H_e(\Phi)) r_b(x) \)

Segmentation \( \frac{\partial \Phi}{\partial t} \) and tracking \( \frac{\partial \Phi}{\partial \text{pose}} \) + any nonlinear optimisation.
5.1.5 Regions — Pixel Wise Posteriors

If $r_f$ and $r_b$ pixel-wise posteriors:

Bibby and Reid

*Robust Real-Time Visual Tracking using Pixel-Wise Posteriors*

Proc ECCV 2008, Marseille
5.1.6 Tracking by Detection

Paradigm integrates detectors (sometime trained live) into the tracking pipeline.

Example: Tracking-Learning-Detection (TLD):

- the target object is defined by a bounding box in a single frame.
- a template/patch-based tracker follows the object frame to frame.
- a random forest-based detector localizes the object and corrects the tracker if needed.
- learning is used to improve the detector.

Kalal at al Tracking-Learning-Detection T-PAMI 2010
5.1.6 Tracking-Learning-Detection
5.1.6 Tracking by detection

Breitenstein, Reichlin, Leibe, Koller-Meier & van Gool *Robust Tracking-by-Detection using a Detector Confidence Particle Filter* Proc 12th ICCV, 2009, 1515-1522
5.1.6 Tracking by detection

Benfold and Reid *Stable Multi-Target Tracking in Real-Time Surveillance Video* Proc CVPR, 2011
Today: ROAM

Miksik et al “ROAM: a Rich Object Appearance Model with Application to Rotoscopying”
ROAM

How does it work? ...

- reincarnation of classic snakes, but with strong and "local" per-edge appearance model (= each edge represented by foreground and background GMM exemplar model).
- global appearance model (= inner part of the object) is integrated through Green theorem (region integral => contour integral).
- pictorial structure (landmarks) - handles "large displacement" + control non-rigid deformability + reduces run-time ('cause it allows you to shrink the state space).
- two alternating minimization steps - both solved exactly with dynamic programming => block coordinate descent (in practice it was enough to run 1 or 2 iterations).

Primary use is rotoscoping (accurate video segmentation for high-end VFX). Code available soon.
5.1.8 Optic flow

Regard the image as a sampling of a continuous irradiance function \( I = I(x, y, t) \).

Follow the pixels from an object as they move through the image. 
\( x = x(t) \) and \( y = y(t) \), and total \( dI/dt \) must exist.

\[
\Rightarrow \frac{dI}{dt} = \frac{\partial I}{\partial t} + \frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt}
\]

But \( I \approx \) constant in the patch

\[
\Rightarrow \frac{dI}{dt} = 0 = \frac{\partial I}{\partial t} + \frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt}
\]

Now write

\[
\nabla I = \left[ \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right]^T \quad \mu = \left[ \frac{dx}{dt}, \frac{dy}{dt} \right]^T
\]

We arrive at the **motion constraint equation**

\[
\nabla I \cdot \mu = -\frac{\partial I}{\partial t}
\]
5.1.8 Aperture problem ... in brief

\[ \nabla I \cdot \mu = -\partial I / \partial t \] — Unfortunately, you only learn about the component of optical flow \( \mu \) in the direction of the image gradient.

You might combine m.c.e with a smoothness term to “estimate” \( \mu \)

\[
\min \iint_{R} \left[ \left( \nabla I \cdot \mu + \frac{\partial I}{\partial t} \right)^2 + \beta \left( |\nabla \mu_x|^2 + |\nabla \mu_y|^2 \right) \right] dxdy.
\]

But you can also fit particular fields to particular patches.

5.1.8 Optic Flow Example: Sintel

Butler et al “A naturalistic open source movie for optical flow evaluation” ECCV 2012
5.2 3D Visual Pose tracking — for Objects
5.2 3D Pose Tracking — Objects

- 1 Simple Polyhedra using control points and RAPiD
- 2 Human Bodies as articulated generalized cylinders
- 3 Articulations using RAPiD
- 4 Using depth cameras, with reconstruction too
5.2.1 3D Tracking using points on edges

Tracking rigid models using RAPiD

Assumes calibrated camera and rigid 3D polyhedral model consisting of
- 3D edges
- control points on model edges

Harris and Stennett *RAPiD – A Video Rate Object Tracker* Proc BMVC 1990, 73-77
5.2.1 RAPiD tracker

Coordinates of a control point given (in camera frame) by

\[ X = T + P \]

After a small motion \( V \delta t, \Omega \delta t \)

\[ X' = T + V \delta t + P + \Omega \delta t \times P \]

Hence the original image position \( x \) moves to

\[
\begin{align*}
x' &= \begin{bmatrix} X'/Z' \\ Y'/Z' \end{bmatrix} \\
&\approx \begin{bmatrix} x + \frac{\delta t}{(T_z+P_z)} \left( V_x + \Omega_y P_z - \Omega_z P_y - x(V_z + \Omega_x P_y - \Omega_y P_x) \right) \\ y + \frac{\delta t}{(T_z+P_z)} \left( V_y + \Omega_z P_x - \Omega_x P_z - y(V_z + \Omega_x P_y - \Omega_y P_x) \right) \end{bmatrix} \\
&= x + \frac{\delta t}{T_z + P_z} \begin{bmatrix} -xP_y & xP_x + P_z & -P_y & 1 & 0 & -x \\ -yP_y - P_z & yP_x & P_x & 0 & 1 & -y \end{bmatrix} \begin{bmatrix} V \\ \Omega \end{bmatrix}
\end{align*}
\]

\[ \Rightarrow x' - x = \delta t G \begin{bmatrix} V \\ \Omega \end{bmatrix} \]
5.2.1 RAPiD tracker

\[ x' - x = \delta t G \begin{bmatrix} V \\ \Omega \end{bmatrix} \] looks solvable, BUT we suffer the aperture problem ...

For each control point, measure distance to edge in image

\[ \ell = \hat{n} \cdot (x' - x) = \hat{n}^\top \delta t G \begin{bmatrix} V \\ \Omega \end{bmatrix} \]

With \( m \) control points, solve with known vector \( \ell \) and matrix \( M \)

\[ \ell_{m \times 1} = M_{m \times 6} \begin{bmatrix} V \\ \Omega \end{bmatrix}_{6 \times 1} \]

Wrap this up in a Kalman Filter — e.g. assuming \( V, \Omega \) are constant.
5.2.1 RAPiD tracker: SUMMARY

- 3D polyhedral model
- 1D search at various points normal to projected model edges
- Filter state (6 dof) using constant velocity KF
5.2.1 Multi-camera RAPiD provides visual feedback


5.2.2 Using Edge and Region information

Model-based human motion capture

29 DOF kinematic model fleshed out with generalized cylinders
The representation goes back to Marr & Nishihara (1977), and Hogg’s “Walker” (1983)
Dynamics handled with particle filter (later)

Deutscher and Reid *Articulated body motion capture by stochastic search* Int J Computer Vision 61 (2005)
5.2.2 Model-based human motion capture

**Measurement model:** Information from both edges and foreground region (after segmentation)

\[ p(z|x) \sim \exp \left( -\sum_{i=1}^{N} \left( \sum_{j} E(z, x) + \sum_{k} R(z, x) \right) \right) \]
5.2.2 Model-based human motion capture

Agile motion ...
5.2.2 Model-based human motion capture

Re-animation ...
5.2.2 Using edges alone was hard ...

**ART — an Articulated Rapid Tracker**

5.2.3 Lines: example tracking using particle filter

Klein and Murray, Proc BMVC 2006

Particle filter recovering from a misaligned pose.

Views showing max complexity handled at 30 Hz (2006)
5.2.4 Regions and Level Sets

Remember the region-based energy function for segmentation and tracking:

\[ E = \int_\omega r_f(x) H_e(\Phi) + (1 - H_e(\Phi)) r_b(x) \]

Segmentation \( \frac{\partial \Phi}{\partial t} \) and tracking \( \frac{\partial \Phi}{\partial \text{pose}} \) + any nonlinear optimisation.
5.2.4 Regions and Level Sets

\[ E = \int_{\omega} r_f(x) H_e(\Phi) + (1 - H_e(\Phi)) r_b(x) \]

For 3D the pose becomes 3D and we differentiate through the projection pipeline.

\[ \frac{\partial E}{\partial \text{pose}} = (\text{term wrt } r_f - r_b) \times \frac{\partial \Phi}{\partial \text{(2D position)}} \times \frac{\partial \text{(2D position)}}{\partial \text{pose}} \]

Can be implemented very quickly with a few hacks and approximations – runs real time on a phone.

Can be augmented with readings from inertial sensor.
5.2.4 Regions and Level Sets

Prisacariu and Reid “PWP3D: Real-time segmentation and tracking of 3D objects” IJCV 2012
5.2.4 Regions and Level Sets

Prisacariu et al. “Simultaneous 3D Tracking and Reconstruction on a Mobile Phone” ISMAR 2013, TVCG 2014, SIGGRAPH Asia 2014
5.2.4 Regions and Level Sets - Can be linked with reconstruction

Prisacariu et al. “Simultaneous 3D Tracking and Reconstruction on a Mobile Phone” ISMAR 2013, TVCG 2014, SIGGRAPH Asia 2014
5.2.4 Regions and Level Sets - Can be linked with reconstruction

Prisacariu et al. “Simultaneous 3D Tracking and Reconstruction on a Mobile Phone” ISMAR 2013, TVCG 2014, SIGGRAPH Asia 2014
5.2.4 With learnt shape models

Shape-prior represented using GP-LVM and optimised over the pose and a low-dimensional latent shape space.

Convergence

Exploring the space

Tracking

Prisacariu, 2014
5.3 3D Colour-Depth Pose tracking — for Objects
Reconstruction & tracking for rehab

The idea behind the European-Union project REWIRE is to allow patients to continue intensive and monitored rehabilitation in their homes.

Need to recover pose of objects and body — limbs, feet, hands
5.3.1 3D Pose Tracking with RGB-D Cameras

Eg Kinect $640 \times 480$ pixel color (RGB) camera
IR emitter and $640 \times 480$ pixel IR camera yielding a depth image (structured-light)

Depth and colour images at a frame rate of 30 fps
Epipolar geometry for the Kinect depth camera

The IR emitter generates the left “camera’s” ray
Reconstruction & tracking for rehab

Kinect is a highly practicable device, but it presents challenges:

- the need for better calibration
- the need for better extremity tracking particularly in the lower limb, feet
5.3.1 Camera-scene geometry

Kinect’s depth camera can be modelled via a central projection ...

A pixel \([x, y]\) has homogeneous coordinates

\[
x^{P} = [xZ, yZ, Z]^\top
\]

where the 3rd component is the depth \(Z\) of the scene point

\[
X^{P} = [X, Y, Z, 1]^\top
\]

The projection is (with actual equality)

\[
\begin{bmatrix}
xZ \\
yZ \\
Z
\end{bmatrix}
= K[I|0]
\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
\]
5.3.1 Knowing the depth $Z$ from RGB-D is not enough!

We can only reconstruct the scene point $X$ once we know $K$

$$
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} = K^{-1}Z
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
$$

$K$ is the intrinsic calibration of the camera

$$
K = \begin{bmatrix}
f_x & s(\approx 0) & c_x \\
0 & f_y & c_y \\
0 & 0 & 1
\end{bmatrix}
$$

Although the average calibration provided with the camera is a starting point, it was insufficient this work.

- We must recover the intrinsics for each camera
- We must also recover the extrinsic between depth and colour

The method is embedded in our approach to tracking ...
5.3.1 Tracking using signed distance functions

Once calibrated, Kinect delivers a dense set of depths points $X$, a subset of which lie near to the object surfaces we want to track.

Consider a 3D model-based generative approach

- Describe object as set of points $X_i^B$ in its own frame $B$.
- Hypothesis pose $p$
- Compute expected positions $X_i(p) = E(p)X_i^B$ in camera frame
- Minimize some norm the overall deviation from measured positions

$$p^* = \arg \min_p \sum_i \|X_i(p) - X_i\|_L$$

Problems

- we don’t have a neat segmentation; and even if we did
- we don’t know the object–scene point correspondence
5.3.1 Instead ...

... devise a cost function based on the 3D chamfer distance $\Phi$.

For any point $X^B_k$, the chamfer distance $\Phi_k$ is the shortest distance from $X^B_k$ to the surface of the object.

Now, for a postulated pose $p$, all the measured scene positions are all converted in the object frame, as $X^B_k = E^{-1}(p)X_k$

For a set of discretized values in voxels, $\Phi_k$ can be pre-computed.

For multiple points those values are summed.

The surface of the object is represented by the zero-level set $\Psi(X) = 0$.

The energy function is defined as $E = \sum_{X \in \Omega} \Psi(X)$

Ren, Prisacariu, Kähler, Murray and Reid *3D tracking of multiple objects with identical appearance using RGB-D input* Proc Int Conf on 3D Vision, 2014
5.3.1 Application to 3D rigid object tracking

■ Assume a known object model

■ The level-set embedding robustified using Geman-McClure
  \( (\sigma^2 \text{ determines the width of the basin of attraction}) \)

\[
\Psi = \frac{\Phi^2}{\Phi^2 + \sigma^2}
\]

■ Derive

\[
\frac{\partial E}{\partial (\Delta p)} = \sum_{\mathbf{x} \in \Omega} \left[ \frac{2\sigma^2 \Phi}{(\Phi^2 + \sigma^2)^2} \frac{\partial \Phi}{\partial \mathbf{X}} \right] \frac{\partial \mathbf{X}}{\partial (\Delta p)}
\]

where \([\partial \Psi / \partial \mathbf{X}]\) is pre-computable.

■ Use Levenberg-Marquardt to optimize pose change \(\Delta p\)

\[
\Delta \tilde{p} = - \left[ J_E^T J_E + \alpha \text{diag} [J_E^T J_E] \right]^{-1} \frac{\partial E}{\partial (\Delta p)}
\]

\[
E_t \leftarrow E(\Delta \tilde{p})E_{t-1}
\]

where \(J_E\) is the Jacobian matrix of the energy function.
5.3.1 Results

Video ... tracking is robust to heavy occlusion
5.3.1 Simultaneous tracking and calibration

When the intrinsic matrix $K$ and the pose $p$ are both unknown, the parameter vector is extended to $\lambda = \{k, \Delta p\}$, where

$$k = [f_x, f_y, c_x, c_y, s]^	op$$

The calibration object large in image — no need for a robust cost

$$\Psi = \Phi^2, \quad \frac{\partial E}{\partial \lambda} = \left[ \sum_{x \in \Omega} 2 \Phi \frac{\partial \Phi}{\partial X} \right] \frac{\partial X}{\partial \lambda}$$

Thereafter, Levenberg-Marquardt as before

Each frame provides an independent optimal calibration for the particular position of the calibration object — so combine them ...

$$\hat{k} = \arg\max_k \sum_{f=1}^F [\bar{k} - k_f]^	op \Sigma_f^{-1} [\bar{k} - k_f]$$
5.3.1 Live Intrinsic Calibration

In low noise, the method can recover the intrinsic parameter accurately from less than 200 frames (7 seconds of data).
5.3.1 Tracking quality after calibration
5.3.1 Calibrating the extrinsics

The (inhomogeneous) scene point is $\mathbf{X} = [X, Y, Z]^{\top}$

A depth and colour pixels have coordinates

$$x_d = Z[x_d, y_d, 1]^{\top} \quad x_c = [x_c, y_c, 1]^{\top}$$

The world coordinate system coincides with that of depth camera, so

$$x_d = K_d X$$
$$\lambda x_c = K_c [RX + t]$$

where $R$ and $t$ are the $3 \times 3$ rotation matrix and $3 \times 1$ translation vector between the color and the depth camera systems. $\lambda$ is an arbitrary scale factor.
5.3.1 extrinsics /ctd

Rearrange to give

\[ \lambda x_c = KC[RK_d^{-1}x_d + t] \]
\[ = A_cRA_d^{-1}x_d + A_c t \]
\[ = Hx_d + m \]

\( H \) is a 3 \times 3 matrix and \( m \) is a 3 \times 1 vector which describe the transformation from depth image to colour image.

With \( n \) pairs of corresponding points solve for \( H \) and \( m \) using MLE by minimizing the 2-norm of the reprojection error

\[ [\hat{m}, \hat{H}] = \arg_{H, m} \min \sum_{i=1}^{n} \|x_{ci} - \hat{x}_i(H, m, x_{di})\|^2 \]
5.3.1 extrinsics / ctd

A wand carrying a characteristically coloured sphere is tracked in both cameras to give $x_d \leftrightarrow x_c$ correspondences.

Result is properly combined RGBD imagery
5.3.2 Tracking and Reconstruction

We introduce: **a) Augmented shape model**

Introduce a co-representation of the shape as a set of labelled voxels \( \{ \mathbf{X}^o, V \} \), where the label takes values \( V = \text{in, on or out} \).

**b) Colour models**

Derived for:

- *foreground* \( P(c|V=\text{on}) \)
- *background* \( P(c|V=\text{out}) \)

Likelihoods initialized by:

- specialist detection module (e.g., skin), or
- user-selected bounding box in the image and are refined over time.
5.3.2 Tracking and Reconstruction

c) Generative graphical model

Indicates what depends on what ...

The locations of voxels $X^o$ are treated as draws from the shape process $\Phi$.

The voxels $X^o$ and their labels $V$, along with the pose $p$, determine the RGB-D imagery $\Omega$

The full joint distribution can therefore be re-written:

$$P(\Omega_0 \ldots \Omega_T, p_0 \ldots p_T, \Phi, \{X^o, V\}) =$$

$$P(\Phi) \prod_i P(X^o_i, V_i|\Phi) \prod_t P(\Omega_t|\{X^o, V\}, p_t)P(p_t)$$
5.3.2 Tracking and Reconstruction

We wanted the optimal poses and the shape, given the imagery

$$\max_{\Phi, p_0 \ldots p_t} P(\Phi, p_0 \ldots p_t | \Omega_0 \ldots \Omega_t) .$$

The full graphical model is computationally intractable.

Instead ...
5.3.2 Tracking and Reconstruction

Alternate periods of

**Tracking:**
Freeze the shape $\Phi$
Seek the ML estimate of the current pose

$$\max_{p_t} P(\Omega_t | \Phi, p_t)$$

**Reconstruction:**
Construct a foreground RGB-D image $\hat{\Omega}_t$ (pixels have $P_{c j}^{on} > P_{c j}^{out}$)
Freeze the poses
Seek the MAP estimate of the shape

$$\arg \max_{\Phi} \left[ P(\Phi | \hat{\Omega}_{1\ldots t}, p_{1\ldots t}) \right]$$
5.3.2 Tracking and Reconstruction

Example results: Shoe reconstruction in object space

2012carl_reconstruct_shoe_pwp
5.3.2 Tracking and Reconstruction

Example results

Color frame  Depth frame  Result

2013carl_STAR3D_x264_002
5.3.2 Tracking details

We want to find

$$\max_{p_t} [P(\Omega_t | \Phi, p_t)]$$

Treat the RGB-D image \( \Omega_t \) as a set of independent pixels \( \{x, c\}_t \):

$$P(\Omega_t | \Phi, p_t) = \prod_{x_{jt} \in \Omega_t} P(x_{jt}, c_{jt} | \Phi, p_t)$$

$$P(x_{jt}, c_{jt} | \Phi, p_t) =$$

$$P(x_{jt} | \Phi, p_t, V_j'=\text{on}) P(c_{jt} | V_j'=\text{on}) + P(x_{jt} | \Phi, p_t, V_j'=\text{out}) P(c_{jt} | V_j'=\text{out})$$

Given \( p_t \), each \( x \) back-projects to a unique \( X^o \)

$$\Rightarrow P(x_{jt} | \Phi, p_t, V_j'=\text{out}) \equiv P(X^o_j | \Phi, V_j'=\text{out})$$

Model these with logistic functions

$$P(X^o_j | \Phi, V=\text{out}) = H^{\text{out}}(\Phi(X^o_j))$$
$$P(X^o_j | \Phi, V=\text{in}) = H^{\text{out}}(\Phi(X^o_j))$$

$$P(X^o_j | \Phi, V=\text{on}) = \delta^{\text{on}}(\Phi(X^o_j))$$

$$\delta^{\text{on}}(\Phi) = 1 - H^{\text{out}}(\Phi) - H^{\text{in}}(\Phi)$$
5.3.3 Tracking multiple objects

Tracking $M$ objects involves
the recovery of $M$ sets of sequences of multiple poses

$\hat{p}_t = \{p_1 \ldots p_M\}_t$

given

- the $M$ object models $\{\Phi_1 \ldots \Phi_M\}$ and
- the depth-colour imagery $\{\Omega_1 \ldots \Omega_t\}$.

The particular challenge here is that the tracked objects may highly similar appearance

Similar appearance means that only two colour models are needed

$P(c|V=\text{on})$ for the objects, and
$P(c|V=\text{out})$ for the background
5.3.3 Tracking multiple objects

We have a modified graphical model.

Convenient to consider voxel locations in the camera frame $\Rightarrow$ their values are dependent on the pose $\mathbf{p}$.

Introduce a intermediate rep $\Phi^c$, which is the union all 3D object shapes in the camera frame.

NB! In implementation, the voxels remain in the object frame and are not duplicated in the camera frame.
5.3.3 Tracking multiple objects

The set of $M$ poses at time $t$, $\hat{p}_t$, is optimized by maximizing the posterior

$$P(\hat{p}_t|\Phi_1\ldots\Phi_M, \Omega_t)$$

$$\sim P(\Omega_t|\Phi_1\ldots\Phi_M, \hat{p}_t)P(\hat{p}_t|\Phi_1\ldots\Phi_M)$$

which depends on two terms ...

**Data term:** $P(\Omega_t|\Phi_1\ldots\Phi_M, \hat{p}_t)$, the image likelihood

**Constraint term:** $P(\hat{p}_t|\Phi_1\ldots\Phi_M)$, looks like a pose prior
5.3.3 Tracking multiple objects

**The Data term**

1. Introduce the “shape union” $\Phi^c$ (and omit subscript $t$). Then

$$P(\Omega|\Phi_1 \ldots \Phi_M, \tilde{\mathbf{p}}) = P(\Omega|\Phi^c) P(\Phi^c|\Phi_1 \ldots \Phi_M, \tilde{\mathbf{p}}).$$

2. As before assuming pixel-wise independence and marginalizing

$$P(\Omega|\Phi^c) = \prod_j P(x_j, c_j|\Phi^c)$$

$$P(x_j, c_i|\Phi^c) = P(x_j|\Phi^c, V_{j,}^{,}=\text{on}) P(c_j|V_{j,}^{,}=\text{on}) + P(x_j|\Phi^c, V_{j,}^{,}=\text{out}) P(c_j|V_{j,}^{,}=\text{out})$$

3. Again, the fore/background pixel location likelihoods are

$$P(x_j|\Phi^c, V_{j,}^{,}=\text{on}) = \delta_{\text{on}}(\Phi^c(\mathbf{X}_{j,}^{c,}))$$

$$P(x_j|\Phi^c, V_{j,}^{,}=\text{out}) = H_{\text{out}}(\Phi^c(\mathbf{X}_{j,}^{c,})).$$

That’s all we need to generate the data-fitting cost $\mathcal{E}_{\text{data}}$ to optimize.
5.3.3 Tracking multiple objects

Not quite all!

We require an expression for $\Phi^c(X^c)$ for any $X^c$.

This can be done rather neatly ...

- Given a set of objects in their own frames $\{\Phi_m\}$ and poses $\{p_m\}$, each can be transformed (notionally) into the camera frame as $\{\Phi^c_m\}$.
- The value of the combined SDF $\Phi^c$ is just

$$\Phi^c(X^c) = \min (\Phi^c_1(X^c), \Phi^c_2(X^c), \ldots, \Phi^c_M(X^c))$$

- No need the copy voxels into the camera frame because $\Phi^c_m(X^c) \equiv \Phi_m(X^o)$
5.3.3 Tracking multiple objects

**Constraint term:** \( P(\tilde{p}_t|\Phi_1 \ldots \Phi_M) \)

Decompose this into a product of per-pose probabilities

\[
P(\tilde{p}|\Phi_1 \ldots \Phi_M) = [P(p_1|\Phi_1 \ldots \Phi_M) = \text{const, as no pose prior!}] \\
\times \prod_{m=2}^{M} P(p_m|[p_1 \ldots p_{m-1}], \Phi_1 \ldots \Phi_M),
\]

The remaining products can be used to enforce pose-related constraints to discourage “collision points” on the surface of one object being inside another.
5.3.3 Tracking multiple objects

2014carl_multiObjects1
5.3.3 Tracking multiple objects

2014carl_multiObjects2
5.4 3D Visual tracking for Camera Pose
5.4 Simultaneous localization and mapping

Assuming a fixed scene and using image data to recover camera pose is often described as visual odometry or camera resectioning. No interest in the scene, just how the camera has moved.

Here instead we will discuss camera pose tracking in the context of simultaneous localization and mapping.

- Track features over time through a video sequence
- Estimate scene points $\mathbf{X}$ and camera poses $\{\mathbf{R}, \mathbf{t}\}_{1...F}$
- Predict where the image points $\mathbf{x}$ should appear
- Measure the deviation from the actual measured position
- Adjust the scene and the poses to reduce the overall error
5.4 The first visual SLAM algorithm (1998)

What the camera sees

- Robot starts with no knowledge, other than it is at \((0, 0)\)
- Fixates on “unknown” points — recovers range & angle. Inserts them into state, with covariance
- Moves, re-fixates, updates state and covariance in (extended)-KF.
- Use stereo cameras, and substantial movements between views

Davison and Murray *Mobile Robot Localization using Active Vision* Proc ECCV 1998, 809-825
Davison and Murray *Sequential Localisation and Map-Building using Active Vision* IEEE Trans on PAMI 24 (2002) 865-880
5.4 First monocular SLAM (2002)

The stereo/motion version used two cameras to recover range and bearing to a 3D point.
Davison & Reid went on to implement it for one camera ... monoSLAM

- Harder — single camera, small displacements.
- Constant velocity EKF with scene points $X_i$ and camera state $c = (X_c, q, V, \Omega)$.
- For speed, initialized using 4 ($\geqslant 3$) known scene points.
- Putative new features detected using Shi-Tomasi ($\approx$Harris), and if added to map stored with $15 \times 15$ appearance template.
- Given predicted camera position, each $X_i$ projected with $3\sigma$ uncertainty region from innovation covariance $S_i$.
- Search made using normalized sum-of-squares correlation.
- Features sorted with highest $S_i$ first.
5.4 First monoSLAM (2002)

5.4 Map point initialization using particle rep

- First measurement leaves depth completely uncertain
- Incremental motion means uncertainty is only slowly tamed
- Represent pdf initially using particles
- Later put into state

Pdf evolution over time
5.4 Augmented Reality using monoSLAM

Graphical augmentation

Active Wearable

Provides fixation
5.4 Recognizing objects and places ...
5.4 PTAM — Parallel Tracking & Mapping

Two key innovations:

- The use of “keyframes”
- Two separate phases:
  - Tracking — assume the 3D map is correct, and section the camera
  - Mapping — bundle adjustment for keyframe poses and map points
5.4 Recognizing objects and places ...

Castle & Murray 2009
5.4 Other types of Sparse SLAM

Industry (to this day – e.g. Tango) uses mono SLAM-like methods.

Academia has moved forward from PTAM and produced e.g.:

- LSD-SLAM: Large Scale Semi-Dense SLAM.
- ORB-SLAM: SLAM with Orb features (and many other tricks and hacks)
- ...

SLAM can also be dense – see DTAM, MonoFusion, MobileFusion.

Dense SLAM can also used Kinect – see KinectFusion, InfiniTAM.
5.4 Monocular Dense SLAM

Dense depth estimated from multiple frames e.g. pairs of frames (PatchMatch) or total variational.

Tracking either sparse (like PTAM) or full dense aliment (L2 norm of colours).

5.4 Depth-based Dense SLAM

InfiniTAM DEMO

Dense depth estimated from Kinect-like depth sensor.

Tracking often ICP.
5.4 Iterative Closest Points (ICP)

ICP minimises the following energy function with respect to pose \((R, t)\):

\[
\sum_{\text{points}} \left( (Rp + t - V(\bar{p})) N(\bar{p}) \right)
\]

\(N(\cdot)\) denotes the normal at a point, so this specific version of ICP is called point-to-plane. There are others (point-to-point, generalised).

Minimisation can be done whichever way you want. We found LM to produce good results.
5.4 Today: Semantics

Vineet, Miksik et al. *Incremental Dense Semantic Stereo Fusion for Large-Scale Semantic Scene Reconstruction*  
Proc ICRA 2015
5.5 Image & Scene Dynamics
5.5 Image and Scene Dynamics

Why is Tracking so important in Computer Vision?

In many sensor systems obtaining the measurement is straightforward. Measurements, though perhaps noisy, are always available. In these cases the key benefits of filtering or tracking are:

- that the estimate of the state improves,
- that the error in the state reduces over time.

When processing sequences it is often the case that prediction makes the measurement available at all by disambiguating between candidate correspondences.

So having a dynamical or kinematical model is often vital ...
5.5.1 The Kalman Filter

Optimal in linear systems when the state and measurements are normally distributed ...

**Status Quo** at the end of timestep \(k\) (using data up to and incl. timestep \(k\)):

| Estimated State | \(x(k|k)\) |
|-----------------|-------------|
| Estimated Covariance | \(P(k|k)\) |

**Prediction** at the next timestep:

| State | \(x(k+1|k) = Fx(k|k)\) |
|-------|--------------------------|
| Covariance | \(P(k+1|k) = FP(k|k)F^\top + Q\) |
| Measurements | \(z|x(k+1|k) \sim \mathcal{N}(Hx(k+1|k), R)\) |

**Measure** at the next timestep:

| Measurements | \(z_{k+1}\) |

**Update** at the next timestep:

| Covariance | \(P(k+1|k+1) = (P^{-1}(k+1|k) + HR^{-1}H^\top)^{-1}\) |
| State | \(x(k+1|k+1) = P(k+1|k+1)[P^{-1}(k+1|k)x(k+1|k) + HR^{-1}z_{k+1}]\) |

(Equivalent ways of expressing this using innovation and its covariance.)
5.5.1 Where do the KF update equations come from?

- Bayes’ thm says that the posterior probability of the state \( x \) given the data \( z \) is the product of the probability of the data given the state and the prior probability of the state, divided by the prior of the data:

\[
p(x|z) = \frac{p(z|x)p(x)}{p(z)}
\]

The prior of the data is a constant, and of no interest.

- Assume that the prior of the state is a multivariate Gaussian centred on \( x_{(k+1|k)} \) with covariance \( P_{(k+1|k)} \), then

\[
p(x) = \exp \left[ \left( x - x_{(k+1|k)} \right)^\top P_{(k+1|k)}^{-1} \left( x - x_{(k+1|k)} \right) \right].
\]

- Similarly

\[
p(z|x) = \exp \left[ (z - Hx)^\top R_{(k|k)}^{-1} (z - Hx) \right].
\]

- But \( p(x|z) \) must be based on the updated state, so

\[
p(x|z) = \exp \left[ \left( x - x_{(k+1|k+1)} \right)^\top P_{(k+1|k+1)}^{-1} \left( x - x_{(k+1|k+1)} \right) \right].
\]
5.5.1 Where do the KF update equations come from?

- Take the log of Bayes’

\[
\ln p(x|z) = \left[ (x - x_{(k+1|k+1)})^\top P_{(k+1|k+1)}^{-1} (x - x_{(k+1|k+1)}) \right]
\]
\[
= \left[ (z - Hx)^\top R^{-1} (z - Hx) \right]
\]
\[
+ \left[ (x - x_{(k+1|k)})^\top P_{(k+1|k)}^{-1} (x - x_{(k+1|k)}) \right] + \text{Constant}
\]

where

\[
x_{(k+1|k)} = Fx_{(k|k)} \quad \text{and} \quad P_{(k+1|k)} = FP_{(k|k)}F^\top + Q
\]

- And differentiate \( \ln p(x|z) \) and set to zero. You’ll find

\[
P_{(k+1|k+1)} = \left[ P_{(k+1|k)}^{-1} + HR^{-1}H^\top \right]^{-1}
\]
\[
x_{(k+1|k+1)} = P_{(k+1|k+1)} \left[ P_{(k+1|k)}^{-1} x_{(k+1|k)} + HR^{-1}z \right]
\]
5.5.1 Graphical Interpretation
5.5.1 Example Kinematic Models in Continuous Time

The most widely used kinematical model used to describe image and/or scene motion is the **constant velocity model**.

It assumes the **acceleration** of a coordinate \( x(t) \) is given by continuous time white noise (the **process noise**):

\[
\ddot{x} = v(t) \quad \text{where} \quad E[v(t)] = 0 \quad \& \quad E[v(t)v(\tau)] = q(t)\delta(t - \tau).
\]

The state vector is

\[
x = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}
\]

and the continuous time state update equation is

\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ v(t) \end{bmatrix}
\]
5.5.1 Constant Velocity Model in Discrete Time

By integration, the discrete time update over a sampling interval $T$ is

$$x((k + 1)T) = Fx(kT) + v(kT)$$

where

$$F = \exp\left\{ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} T \right\} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}$$

and

$$v(kT) = \int_{\tau=0}^{T} \exp\left\{ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} (T - \tau) \right\} \begin{bmatrix} 0 \\ 1 \end{bmatrix} v(kT + \tau) d\tau.$$

The process noise covariance matrix is

$$Q = E[vv^T] = \int_{0}^{T} \left[ \begin{bmatrix} T - \tau \\ 1 \end{bmatrix} \begin{bmatrix} T - \tau \\ 1 \end{bmatrix}^T \right] q d\tau = \begin{bmatrix} \frac{1}{3}T^3 & \frac{1}{2}T^2 \\ \frac{1}{2}T^2 & T \end{bmatrix} q$$

This assumes $q$ is constant over the interval.
(Usual to drop the $T$ and write $x(k)$ instead of $x(kT)$.)
5.5.1 Cousins of the Kalman Filter

- The Extended Kalman Filter (EKF) linearizes a non-linear measurement and/or state update

- The Iterated EKF (IEKF) repeated linearization

- The Unscented Kalman Filter (UKF) makes a modest sampling of the state space around the current state

However the Kalman Filter and variants can (and often does) fail catastrophically if the measurement density is multimodal.
5.5.2 Particle Filtering

The Kalman Filter can (and often does) fail catastrophically if the measurement density is multimodal.

Eg: Contour tracker
Eg: RAPiD tracker

Can employ robust statistical methods to identify rogue data or “outliers”
Or represent multiple hypotheses — no longer Gaussian.
5.5.2 Representing pdfs with weighted particles

Particle filtering is a **numerical** technique for representing and propagating arbitrary densities. Unlike KF, not an analytical technique.

One is still trying to work Bayes’ theorem to obtain a posterior distribution from a prior and an observation density ...

The problem is that one does not have time to work out $p(z|x)$ for every configuration of the state.
5.5.2 Representing pdfs with weighted particles

At timestep $k$, represent an arbitrary distribution **numerically** using a set of $N$ **weighted particles**

$$\{s_k^{(i)}, \pi_k^{(i)} | i = 1 \ldots N\}$$

where $s$ is a particular instance of the state $x = s$, and where $\sum_i \pi^{(i)} = 1$.

This is a **discrete** approximation to a continuous density function, where **weight ↔ density**

Particles should be clustered where there is high density. (Would be pointless to have lots of particles with $\pi = 0$.)
5.5.2 Now use Condensation ...

Condensation algorithm:

\[ S_k^{(i)}, \pi_k^{(i)} \]

\[ S_{k+1}, \pi_{k+1} \]

Predict

Measure
5.5.2 Condensation

Condensation **algorithm:**

1. Choose $N$ samples from the existing set according to their $\pi_k$ values. (Any particle can be picked multiple or zero times.)

   
   Random number

   $\pi^{(1)}_k \quad \pi^{(i)}_k \quad \pi^{(N)}_k$

2. Allow their individual states to evolve according to the motion model including random noise. This give the new states $s_{k+1}$.

3. Derive the new weights for a state by testing the likelihood of the current observation given that state

   $$\pi^{(i)}_{k+1} = p(z_{k+1}|x_{k+1} = s^{(i)}_{k+1})$$

   Normalize so that $\sum_i \pi^{(i)}_{k+1} = 1$.

NB! that there is no “correct state value”. You are representing a pdf.
5.5.3 Condensation: multiple hypotheses
5.5.3 Condensation tracking agile motion in clutter

[Image: A diagram showing two graphs with trajectories labeled "Condensation tracker" and "Kalman filter tracker". The graphs are plotted in a 3D coordinate system with X, Y, and Z axes. The time is marked as 10 s on each graph.]

The diagram illustrates the comparison between condensation tracking and Kalman filter tracking for agile motion in a cluttered environment.
Computer Vision: Lecture 5

5.1 2D Tracking

5.2 3D Visual Pose tracking for object pose

5.3 3D RGB-D Pose tracking for object pose

5.4 3D Visual Pose tracking for camera pose

5.5 A note on image and scene dynamics