

Visual SLAM

Andrew Davison, Andrew Calway, Walterio Mayol

Imperial College

University of Bristol

Visual SLAM

- Simultaneous Localisation and Mapping with a single camera (Monocular SLAM).
- **Localisation** : track the 6-D pose (3-D position and orientation) of an agile (e.g. hand-held) camera;
- **Mapping** : whilst also determining the 3-D structure of the surroundings.
- **In real-time, e.g. > 30 fps**
- Using only vision, i.e. no gyros, accelerometers, etc.



Visual SLAM with a Single Camera 2003-2007

Applications

- Anything that would benefit from a low cost, portable positioning device:
 - Wearable computing (location awareness)
 - Augmented reality (real-time interfaces)
 - Robotics (industrial and domestic)
 - Virtual map building (Google earth)
- But also to facilitate wider research in vision (human and computer) → provides a tool to collect vast amounts of relative pose stamped data in real-time

The Probabilistic Way

- Several possible approaches to localisation and mapping
- In Computer Vision, SFM work has made significant progress:
 - e.g: real-time ‘visual odometry’ work of Nister, 2004
 - essentially batch processing and optimisation (RANSAC)
- Here we consider the sequential probabilistic approach, based around Kalman filtering:
 - necessarily less accurate
 - but genuine ‘online solution’, with rigorous representation of estimation uncertainty.

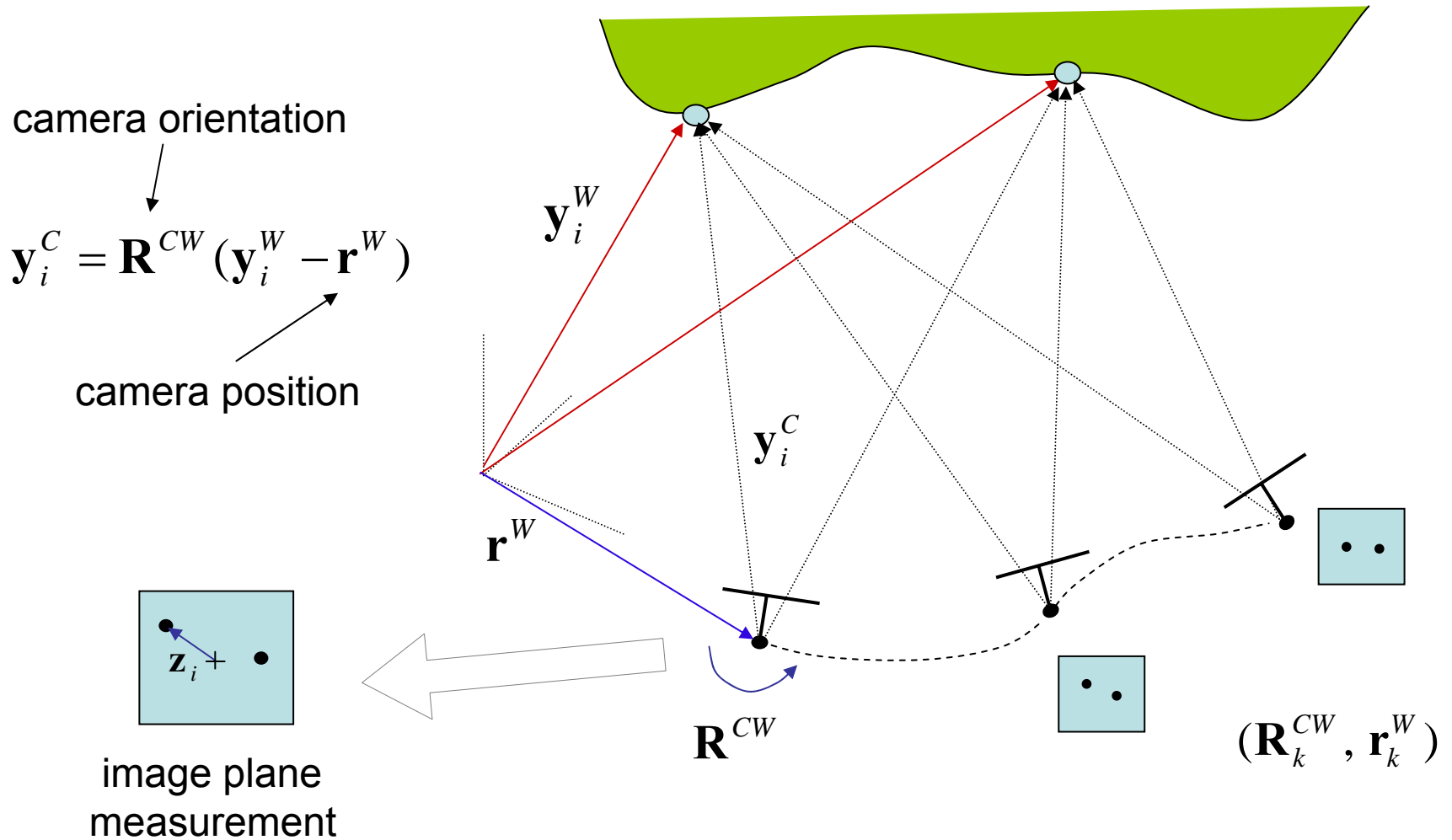
Related Work

- Probabilistic frameworks for SLAM have been studied extensively in Robotics
- Using a variety of sensors, including vision.
- For example → work of Durrant-Whyte and Thrun.
- But – primarily aimed at autonomous robots, often slow moving with additional control input, e.g. odometry
- Here we consider visual SLAM for a highly agile single camera, e.g hand-held, operating in real-time.

Tutorial Outline

- Part I : **Localisation using probabilistic filtering**
‘tracking a camera in real-time with known structure’
- Part II : **Simultaneous localisation and mapping**
‘building a point based scene map whilst tracking’
- Part III : **Advances and challenges**
‘recent progress and what needs to be done next’

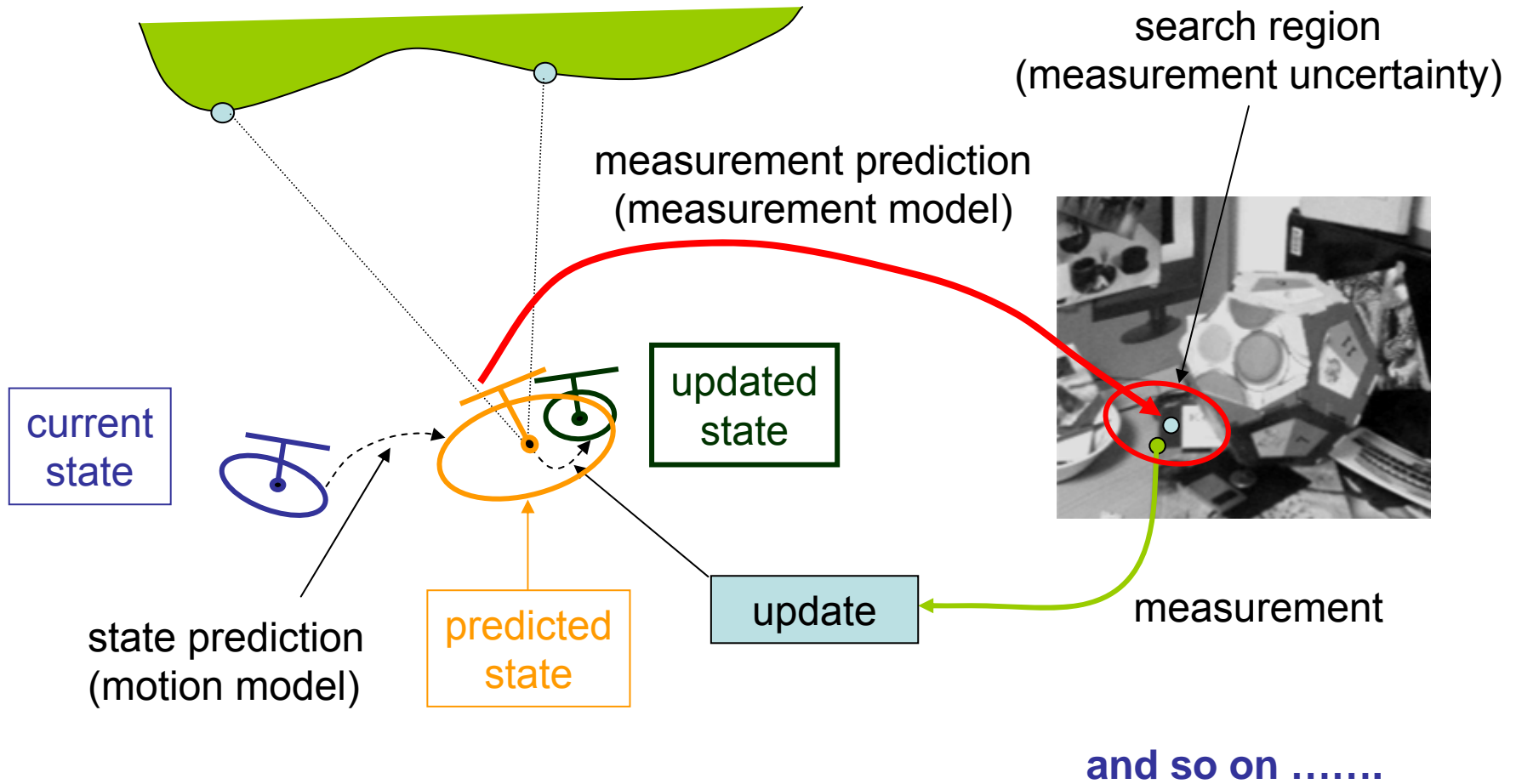
Camera Localisation



Probabilistic Filtering

- Localisation – for each frame, obtain estimates of parameters \mathbf{R}^{CW} and \mathbf{r}^W from measurements \mathbf{z}_i .
- Probabilistic filtering - parameters and measurements treated as random variables.
- Parameter estimates and uncertainties obtained from estimates of density functions:
 - either in parametric form (e.g. Kalman filtering using Gaussian means and covariances)
 - or using approximations (e.g. importance sampling, particle filtering)
- Rigorous framework for obtaining online estimates and associated uncertainties.

Predictor-Corrector



Detection vs Tracking

- The predictor-corrector framework enables active tracking of features.
- Measurement uncertainty regions derived from the filter constrain the search for matching features
 - reduces image processing operations → real-time performance
- Contrast with detection methodologies – detect all potential features and find best set of matches with previous frame (e.g. using optimisation, RANSAC, etc).
- NB: as uncertainty in camera pose increases – search regions increase → detection + matching takes over.

Localisation : The State

- For localisation, filter state at given time represents the pose of the camera w.r.t the world coordinate system.
- This is defined by its position \mathbf{r}^W and rotation \mathbf{R}^{CW}
- For rotation, use the redundant but easy to use and numerically stable quaternion representation, i.e.

$$\mathbf{q}^{CW} = (q_0^{CW}, q_1^{CW}, q_2^{CW}, q_3^{CW}) \equiv \mathbf{R}^{CW}$$

- Giving the 7-D state vector:

$$\mathbf{x} = \begin{bmatrix} \mathbf{r}^W \\ \mathbf{q}^{CW} \end{bmatrix}$$

Localisation : Motion Model

- State predictions are defined by the motion model representing the assumed dynamics of the camera.
- In general, $\mathbf{x}^{new} = \mathbf{f}(\mathbf{x}, \mathbf{n})$, where \mathbf{n} is a noise term allowing us to incorporate uncertainty.
- Example : constant position model (Gaussian)

$$\begin{bmatrix} \mathbf{r}^{new} \\ \mathbf{q}^{new} \end{bmatrix} = \begin{bmatrix} \mathbf{r} + \mathbf{n}_r \\ \mathbf{q} \otimes \mathbf{q}(\mathbf{n}_\omega) \end{bmatrix} \quad \mathbf{n} = \begin{bmatrix} \mathbf{n}_r \\ \mathbf{n}_\omega \end{bmatrix} \sim N(0, \mathbf{P}_{nn})$$

\otimes quaternion multiplication

$\mathbf{q}(\mathbf{n}_\omega) \longrightarrow$ quaternion representing rotation defined by \mathbf{n}_ω

Constant Velocity Model

- Expand state to include velocity terms and assume zero mean (Gaussian) acceleration between frames:

$$\begin{bmatrix} \mathbf{r}^{new} \\ \mathbf{q}^{new} \\ \mathbf{v}^{new} \\ \boldsymbol{\omega}^{new} \end{bmatrix} = \begin{bmatrix} \mathbf{r} + (\mathbf{v} + \mathbf{n}_r)\Delta t \\ \mathbf{q} \otimes \mathbf{q}((\boldsymbol{\omega} + \mathbf{n}_\omega)\Delta t) \\ \mathbf{v} + \mathbf{n}_r \\ \boldsymbol{\omega} + \mathbf{n}_\omega \end{bmatrix} \quad \mathbf{n} = \begin{bmatrix} \mathbf{n}_r \\ \mathbf{n}_\omega \end{bmatrix} \sim N(\mathbf{0}, \mathbf{P}_{nn})$$

- Giving a 13-D state vector.

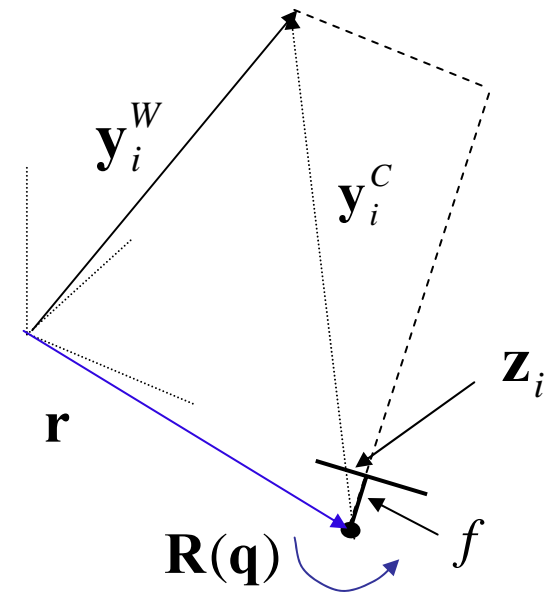
better for fast smooth
camera motion

Localisation : Measurement Model

- Measurements \mathbf{z}_i related to the state by camera parameters and perspective projection.
- In general, $\mathbf{z}_i = \mathbf{h}_i(\mathbf{x}, \mathbf{y}_i^W, \mathbf{e})$, where \mathbf{e} is a measurement noise term.
- Assuming a pinhole calibrated camera:

$$\mathbf{z}_i = \underbrace{\mathbf{R}(\mathbf{q})(\mathbf{y}_i^W - \mathbf{r})}_{\mathbf{y}_i^C} + \mathbf{e}_i$$

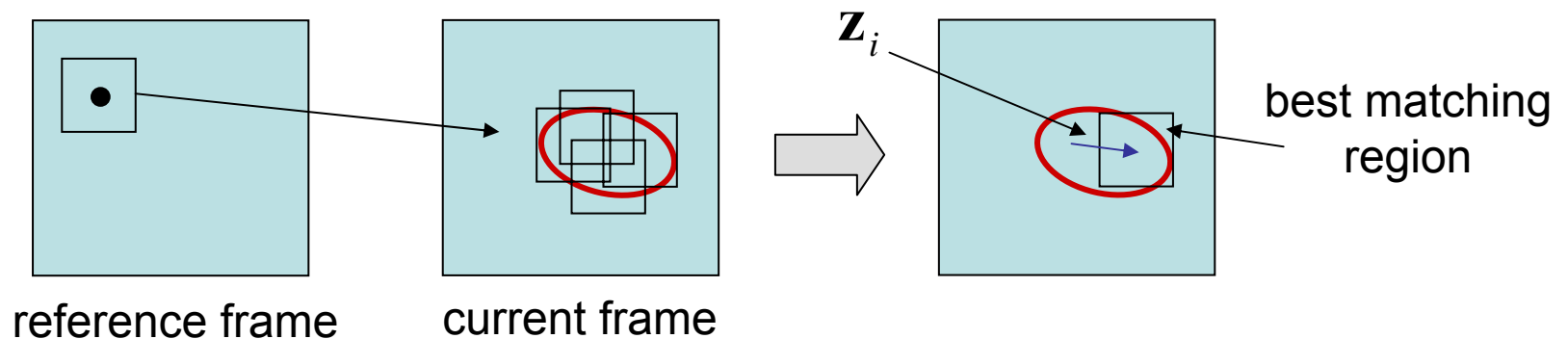
↑
pinhole projection



- Can also define $\mathbf{h}()$ to take account of radial distortion and use of wide angled lens

The Measurements

- Measurements – estimated projections of point features with known 3-D position into current frame.
- Difficult problem even with known 3-D position (perspective effects, occlusions, lighting) – more later.
- Basic solution:
 - template matching (e.g. correlation) using reference template
 - search within measurement uncertainty window (from filter)



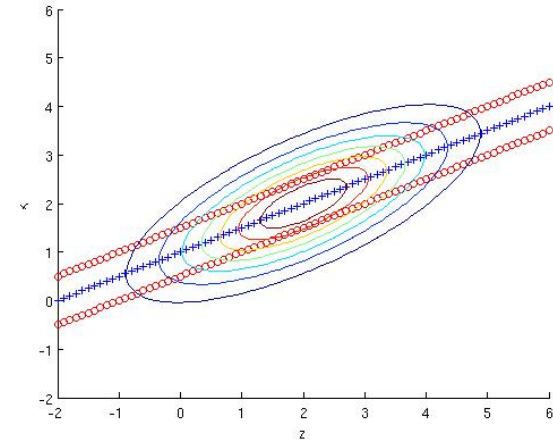
The Kalman Filter

- Given all measurements up to current time, the Kalman filter algorithm is the optimal Minimum Mean Squared Error (MMSE) estimator of the state, i.e. the best
- **Providing:**
 - initial state is Gaussian with known mean and covariance;
 - motion and measurement models are linear;
 - and noise terms are uncorrelated, white, Gaussian, zero mean and with known covariances.
- For camera localisation - motion and measurement models are non-linear → use **Extended Kalman Filter** (approximation based on linearisation).

Static MMSE Estimation

- Jointly normal state \mathbf{x} and measurement \mathbf{z}

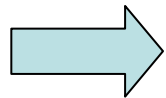
$$\mathbf{y} = \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} \quad \bar{\mathbf{y}} = \begin{bmatrix} \bar{\mathbf{x}} \\ \bar{\mathbf{z}} \end{bmatrix} \quad \mathbf{P}_{yy} = \begin{bmatrix} \mathbf{P}_{xx} & \mathbf{P}_{xz} \\ \mathbf{P}_{zx} & \mathbf{P}_{zz} \end{bmatrix}$$



- MMSE estimate : $\hat{\mathbf{x}} = E[\mathbf{x} | \mathbf{z}]$

$$p(\mathbf{x} | \mathbf{z}) = p(\mathbf{x}, \mathbf{z}) / p(\mathbf{z})$$

$$\propto \exp[-(\mathbf{y} - \bar{\mathbf{y}})' \mathbf{P}_{yy}^{-1} (\mathbf{y} - \bar{\mathbf{y}}) + (\mathbf{z} - \bar{\mathbf{z}})' \mathbf{P}_{zz}^{-1} (\mathbf{z} - \bar{\mathbf{z}})]$$



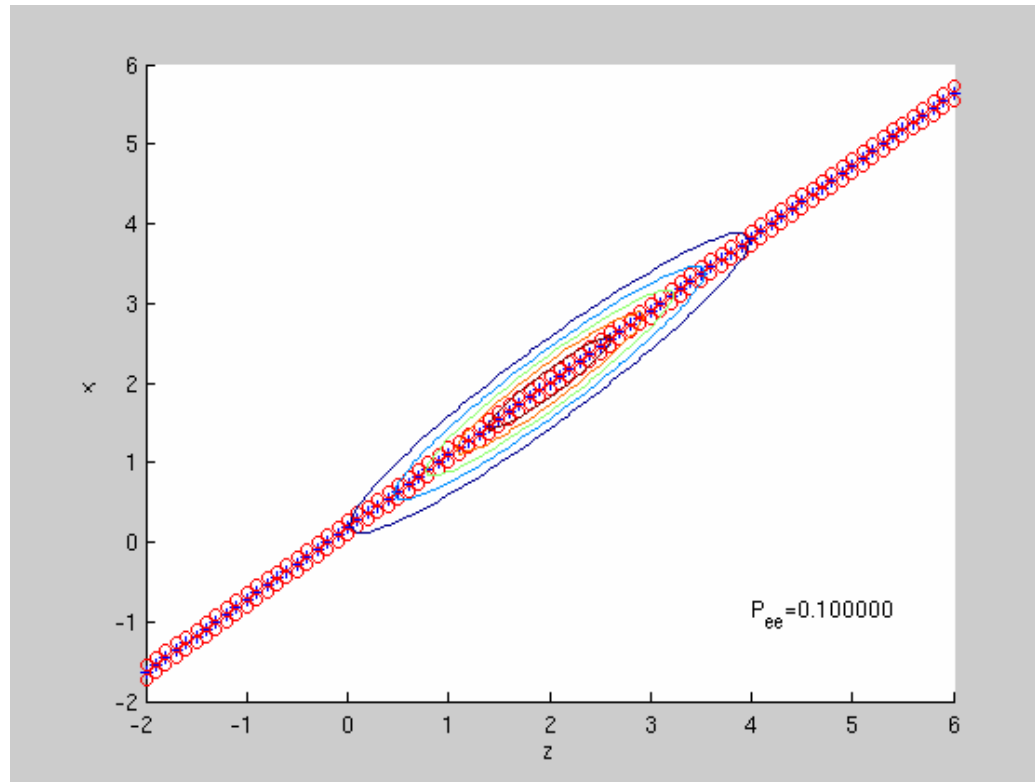
$$\hat{\mathbf{x}} = E(\mathbf{x} | \mathbf{z}) = \bar{\mathbf{x}} + \mathbf{P}_{xz} \mathbf{P}_{zz}^{-1} (\mathbf{z} - \bar{\mathbf{z}}) \quad \text{Linear in } \mathbf{z}$$

$$\mathbf{P}_{xx|z} = \text{cov}(\mathbf{x} | \mathbf{z}) = \mathbf{P}_{xx} - \mathbf{P}_{xz} \mathbf{P}_{zz}^{-1} \mathbf{P}_{zx} \quad \text{Independent of } \mathbf{z}$$

MMSE : Simple Example

$$z = x + e \quad e \sim N(0, P_{ee}) \quad P_{yy} = \begin{bmatrix} P_{xx} & P_{xx} \\ P_{xx} & P_{xx} + P_{ee} \end{bmatrix}$$

$$P_{xx} = 1$$



Linear Time-Varying Process

Models : $\mathbf{x}^{new} = \mathbf{F}\mathbf{x} + \mathbf{n}$ $\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{e}$

$\mathbf{u} = \mathbf{A}\mathbf{v} \Rightarrow \mathbf{P}_{uu} = \mathbf{A}\mathbf{P}_{vv}\mathbf{A}'$

Predictions for next time step:

$$\begin{aligned} \bar{\mathbf{x}}^{pred} &= \mathbf{F}\bar{\mathbf{x}} & \mathbf{P}_{xx}^{pred} &= \mathbf{F}\mathbf{P}_{xx}\mathbf{F}' + \mathbf{P}_{nn} & \mathbf{P}_{zx}^{pred} &= \mathbf{H}\mathbf{P}_{xx}^{pred} \\ \bar{\mathbf{z}}^{pred} &= \mathbf{H}\bar{\mathbf{x}}^{pred} & \mathbf{P}_{zz}^{pred} &= \mathbf{H}\mathbf{P}_{xx}^{pred}\mathbf{H}' + \mathbf{P}_{ee} & \mathbf{P}_{xz}^{pred} &= \mathbf{P}_{xx}^{pred}\mathbf{H}' \end{aligned}$$

MMSE : $\hat{\mathbf{x}} = \bar{\mathbf{x}} + \mathbf{P}_{xz}\mathbf{P}_{zz}^{-1}(\mathbf{z} - \bar{\mathbf{z}})$ $\mathbf{P}_{xx|z} = \mathbf{P}_{xx} - \mathbf{P}_{xz}\mathbf{P}_{zz}^{-1}\mathbf{P}_{zx}$

Updates:

$$\bar{\mathbf{x}}^{update} = \bar{\mathbf{x}}^{pred} + \mathbf{P}_{xz}^{pred}\mathbf{P}_{zz}^{pred^{-1}}(\mathbf{z} - \bar{\mathbf{z}}^{pred})$$

Kalman Gain

$$\mathbf{P}_{xx}^{update} = \mathbf{P}_{xx}^{pred} - \mathbf{P}_{xz}^{pred}\mathbf{P}_{zz}^{pred^{-1}}\mathbf{P}_{zx}^{pred}$$

Kalman Filter

Extended Kalman Filter

General models : $\mathbf{x}^{new} = \mathbf{f}(\mathbf{x}, \mathbf{n}) \quad \mathbf{z} = \mathbf{h}(\mathbf{x}, \mathbf{e})$

Jacobian

Predictions: $\bar{\mathbf{x}}^{pred} = \mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{n}}) \quad \bar{\mathbf{z}}^{pred} = \mathbf{h}(\bar{\mathbf{x}}^{pred}, \bar{\mathbf{e}})$

$$\mathbf{P}_{xx}^{pred} = \nabla \mathbf{f}_x \mathbf{P}_{xx} \nabla \mathbf{f}_x' + \nabla \mathbf{f}_n \mathbf{P}_{nn} \nabla \mathbf{f}_n' \quad \mathbf{P}_{zx}^{pred} = \nabla \mathbf{h}_x \mathbf{P}_{xx}^{pred}$$

$$\mathbf{P}_{zz}^{pred} = \nabla \mathbf{h}_x \mathbf{P}_{xx}^{pred} \nabla \mathbf{h}_x' + \nabla \mathbf{h}_e \mathbf{P}_{ee} \nabla \mathbf{h}_e' \quad \mathbf{P}_{xz}^{pred} = \mathbf{P}_{xx}^{pred} \nabla \mathbf{h}_x'$$

Updates:

$$\bar{\mathbf{x}}^{update} = \bar{\mathbf{x}}^{pred} + \mathbf{K}(\mathbf{z} - \bar{\mathbf{z}}^{pred}) \quad \mathbf{P}_{xx}^{update} = \mathbf{P}_{xx}^{pred} - \mathbf{K} \nabla \mathbf{h}_x \mathbf{P}_{xx}^{pred}$$

$$\mathbf{K} = \mathbf{P}_{xx}^{pred} \nabla \mathbf{h}_x' (\nabla \mathbf{h}_x \mathbf{P}_{xx}^{pred} \nabla \mathbf{h}_x' + \nabla \mathbf{h}_e \mathbf{P}_{ee} \nabla \mathbf{h}_e')^{-1}$$

EKF for Localisation

Constant position model:

$$\begin{bmatrix} \mathbf{r}^{new} \\ \mathbf{q}^{new} \end{bmatrix} = \begin{bmatrix} \mathbf{r} + \mathbf{n}_r \\ \mathbf{q} \otimes \mathbf{q}(\mathbf{n}_\omega) \end{bmatrix} \equiv \mathbf{f}(\mathbf{x}, \mathbf{n})$$

Measurement model:

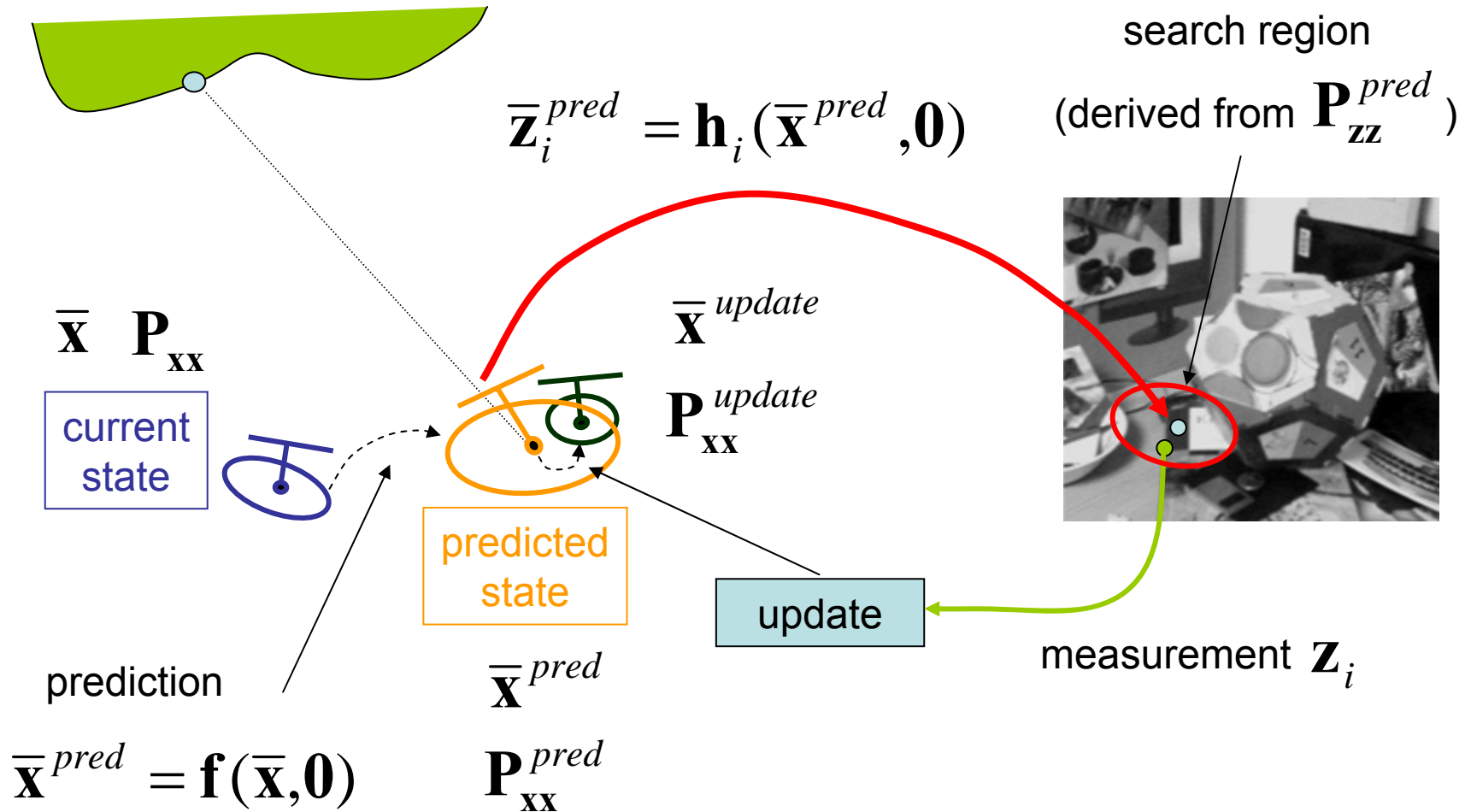
$$\mathbf{z}_i = \mathbf{h}_i(\mathbf{x}, \mathbf{e}) = \mathbf{R}(\mathbf{q})(\mathbf{y}_i^W - \mathbf{r}) + \mathbf{e}_i$$

Example Jacobian:

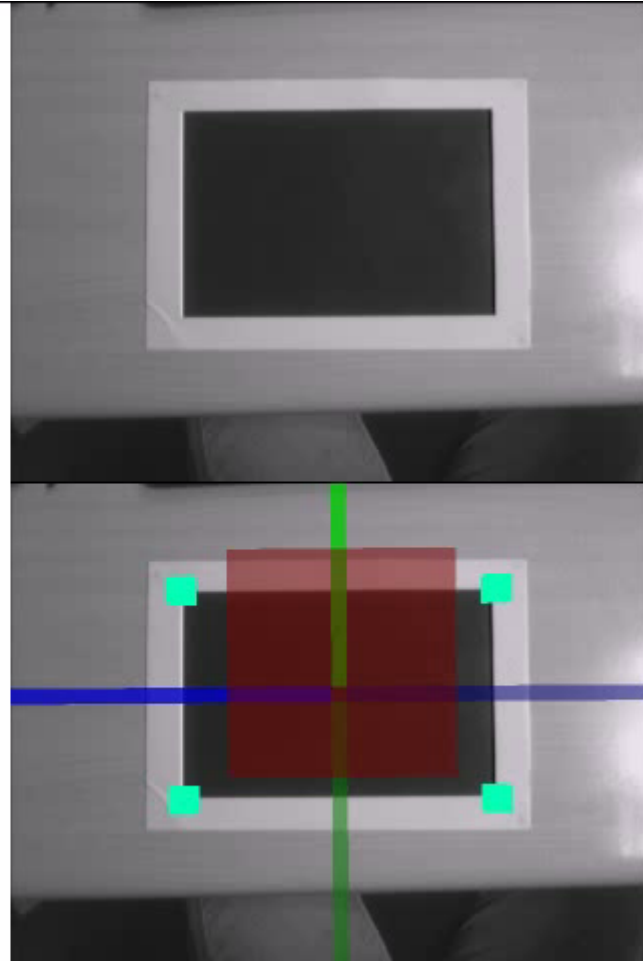
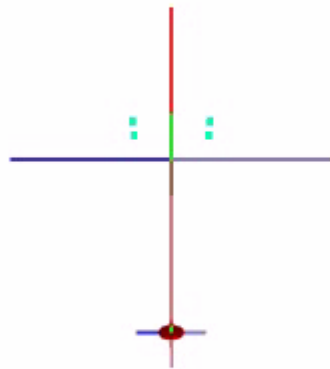
$$\nabla \mathbf{f}_x = \begin{bmatrix} \partial(\mathbf{r} + \mathbf{n}_r) / \partial \mathbf{r} & \mathbf{0}_{3 \times 4} \\ \mathbf{0}_{4 \times 3} & \partial(\mathbf{q} \otimes \mathbf{q}(\mathbf{n}_\omega)) / \partial \mathbf{q} \end{bmatrix}$$

NB: $\nabla \mathbf{f}_n \neq \mathbf{I}$ and $\nabla \mathbf{h}_e = \mathbf{I}$

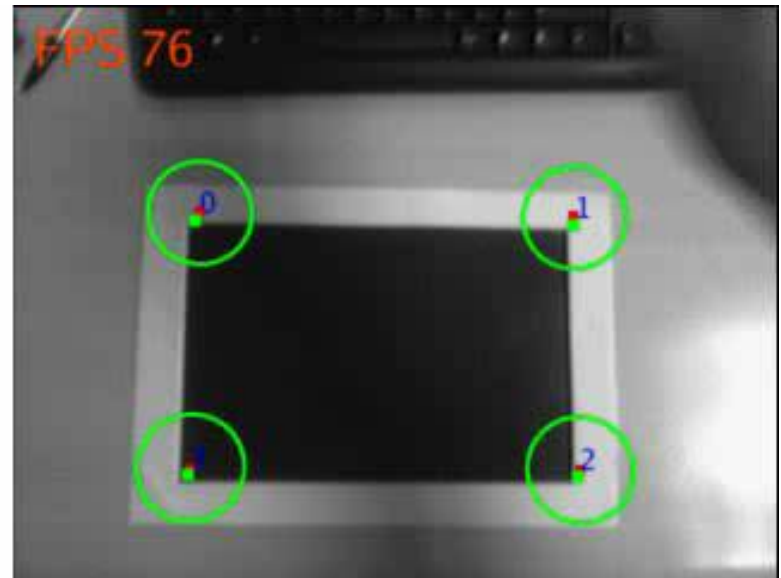
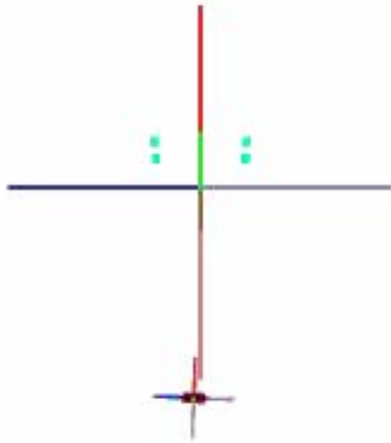
EKF Localisation Algorithm



Example : EKF Localisation

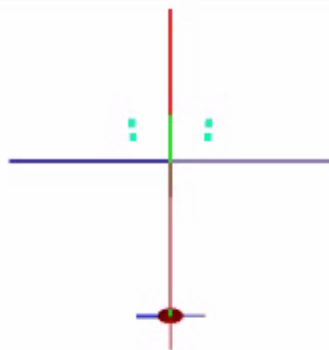


Adaptive Search Regions



But feature matching is important...

matching
with
descriptors



template
matching

