Chapter 14. (Supplementary) Bayesian Filtering for State Estimation of Dynamic Systems

*Neural Networks and Learning Machines* (Haykin)

Lecture Notes on
Self-learning Neural Algorithms

Byoung-Tak Zhang
School of Computer Science and Engineering
Seoul National University

Version 20171115
Supplementary Material to Ch 14

- Kalman Filter .......................... 3
- Sequential Monte Carlo ............ 25
- Particle Filters ......................... 35
Overview

• The Problem – Why do we need Kalman Filters?
• What is a Kalman Filter?
• Conceptual Overview
• The Theory of Kalman Filter
• Simple Example
The Problem

- System state cannot be measured directly
- Need to estimate “optimally” from measurements
What is a Kalman Filter?

• **Recursive** data processing algorithm
• Generates **optimal** estimate of desired quantities given the set of measurements
• Optimal?
  – For linear system and white Gaussian errors, Kalman filter is “best” estimate based on all previous measurements
  – For non-linear system optimality is ‘qualified’
• Recursive?
  – Doesn’t need to store all previous measurements and reprocess all data each time step
Conceptual Overview

• Simple example to motivate the workings of the Kalman Filter
• Theoretical Justification to come later – for now just focus on the concept
• Important: Prediction and Correction
Conceptual Overview

- Lost on the 1-dimensional line
- Position – $y(t)$
- Assume Gaussian distributed measurements
Conceptual Overview

- Sextant Measurement at $t_1$: Mean = $z_1$ and Variance = $\sigma_{z_1}$
- Optimal estimate of position is: $\hat{y}(t_1) = z_1$
- Variance of error in estimate: $\sigma_x^2(t_1) = \sigma_{z_1}^2$
- Boat in same position at time $t_2$ - Predicted position is $z_1$
Conceptual Overview

- So we have the prediction $\hat{y}(t_2)$
- GPS Measurement at $t_2$: Mean = $z_2$ and Variance = $\sigma_{z2}$
- Need to **correct** the prediction due to measurement to get $\hat{y}(t_2)$
- Closer to more trusted measurement – linear interpolation?
Corrected mean is the new optimal estimate of position
New variance is smaller than either of the previous two variances
Conceptual Overview

• Lessons so far:

  Make prediction based on previous data - $\hat{y}^-, \sigma^-$

  Take measurement – $z_k, \sigma_z$

  Optimal estimate ($\hat{y}$) = Prediction + (Kalman Gain) * (Measurement - Prediction)

  Variance of estimate = Variance of prediction * (1 – Kalman Gain)
Conceptual Overview

- At time $t_3$, boat moves with velocity $\frac{dy}{dt}=u$
- Naïve approach: Shift probability to the right to predict
- This would work if we knew the velocity exactly (perfect model)
Better to assume imperfect model by adding Gaussian noise
\[
\frac{dy}{dt} = u + w
\]
Distribution for prediction moves and spreads out

- Naïve Prediction
- Prediction

\[\hat{y}(t_2)\]
\[\hat{y}(t_3)\]
• Now we take a measurement at $t_3$
• Need to once again correct the prediction
• Same as before
Conceptual Overview

- Lessons learnt from conceptual overview:
  - Initial conditions ($\hat{y}_{k-1}$ and $\sigma_{k-1}$)
  - Prediction ($\hat{y}^{-}_{k}$, $\sigma^{-}_{k}$)
    - Use initial conditions and model (e.g., constant velocity) to make prediction
  - Measurement ($z_{k}$)
    - Take measurement
  - Correction ($\hat{y}_{k}$, $\sigma_{k}$)
    - Use measurement to correct prediction by ‘blending’ prediction and residual – always a case of merging only two Gaussians
    - Optimal estimate with smaller variance
Theoretical Basis

• Process to be estimated:

\[ y_k = Ay_{k-1} + Bu_k + w_{k-1} \]  
Process Noise \((w)\) with covariance \(Q\)

\[ z_k = Hy_k + w_k \]  
Measurement Noise \((v)\) with covariance \(R\)

• Kalman Filter

Predicted: \(\hat{y}_k\) is estimate based on measurements at previous time-steps

\[ \hat{y}_k = Ay_{k-1} + Bu_k \]

\[ P_{-k} = AP_{k-1}A^T + Q \]

Corrected: \(\hat{y}_k\) has additional information – the measurement at time \(k\)

\[ \hat{y}_k = \hat{y}_{-k} + K(z_k - H \hat{y}_{-k}) \]

\[ K = P_{-k}H^T(HP_{-k}H^T + R)^{-1} \]

\[ P_k = (I - KH)P_{-k} \]
Blending Factor

• If we are sure about measurements:
  – Measurement error covariance (R) decreases to zero
  – K decreases and weights residual more heavily than prediction

• If we are sure about prediction
  – Prediction error covariance $P_{-k}$ decreases to zero
  – K increases and weights prediction more heavily than residual
Theoretical Basis

<table>
<thead>
<tr>
<th>Prediction (Time Update)</th>
<th>Correction (Measurement Update)</th>
</tr>
</thead>
</table>
| (1) Project the state ahead
\[ \hat{y}_k = Ay_{k-1} + Bu_k \]
| (1) Compute the Kalman Gain
\[ K = P_{k-1}^T(HP_{k-1}^TH^T + R)^{-1} \]
| (2) Project the error covariance ahead
\[ P_{k-1} = AP_{k-1}A^T + Q \]
| (2) Update estimate with measurement \( z_k \)
\[ \hat{y}_k = \hat{y}_{k-1} + K(z_k - H \hat{y}_{k-1}) \]
| (3) Update Error Covariance
\[ P_k = (I - KH)P_{k-1} \]
Quick Example – Constant Model

System

External Controls

System Error Sources

System State

Measuring Devices

Measurement Error Sources

Black Box

Observed Measurements

Estimator

Optimal Estimate of System State
Quick Example – Constant Model

Prediction

\[ \hat{y}_k^- = y_{k-1} \]
\[ P_k^- = P_{k-1} \]

Correction

\[ K = P_k^-(P_k^- + R)^{-1} \]
\[ \hat{y}_k = \hat{y}_k^- + K(z_k - H \hat{y}_k^-) \]
\[ P_k = (I - K)P_k^- \]
Quick Example – Constant Model
Quick Example – Constant Model

Convergence of Error Covariance - $P_k$
Quick Example – Constant Model

Larger value of R – the measurement error covariance (indicates poorer quality of measurements)

Filter slower to ‘believe’ measurements – slower convergence
References


Sequential Monte Carlo
Monte Carlo (MC) Approximation

\[ E_p[f(x)] = \int p(x)f(x)dx \]

\[ \approx \frac{1}{N} \sum_{i=1}^{N} f(x^{(i)}), \quad x^{(i)} \sim p(x) = N(0, \sigma^2) \]

- Monte Carlo approach
  1. Simulate \( N \) random variables from \( p(x) \), e.g. Normal distribution

\[ x^{(i)} \sim p(x) = N(0, \sigma^2) \]

2. Compute the average

\[ E_p[f(x)] = \frac{1}{N} \sum_{i=1}^{N} f(x^{(i)}) \]
MC with Importance Sampling

\[ E_p[f(x)] = \int p(x)f(x)dx \]

\[ = \int \frac{p(x)}{q(x)} q(x)f(x)dx \]

\[ \approx \sum_{i=1}^{N} w_i f(x^{(i)}) \]

\[ x^{(i)} \sim q(x) \] \hspace{1cm} q(x): \text{proposal distribution} \]

\[ w_i = \frac{p(x^{(i)})}{q(x^{(i)})} \] \hspace{1cm} w_i: \text{importance weight} \]

Note: q(x) is easier to sample from than p(x).
Importance Sampling (IS)

\[ E[f(x_{0:t})] = \int f(x_{0:t}) p(x_{0:t} \mid y_{1:t}) dx_{0:t} \]

\[ \approx \sum_{i=1}^{N} w_i f(x^{(i)}_{0:t}) \]

\[ x^{(i)}_{0:t} \sim q(x_{0:t} \mid y_{1:t}) \quad q(x): \text{proposal distribution} \]

\[ w_i = \frac{p(x^{(i)}_{0:t} \mid y_{1:t})}{q(x^{(i)}_{0:t} \mid y_{1:t})} \quad w_i: \text{importance weight} \]
Importance Sampling: Procedure

1. Draw $N$ samples $x^{(i)}_{0:t}$ from proposal distribution $q(.)$.

   $x^{(i)}_{0:t} \sim q(x_{0:t} \mid y_{1:t})$

2. Compute importance weight

   $w(x^{(i)}_{0:t}) = \frac{p(x^{(i)}_{0:t} \mid y_{1:t})}{q(x^{(i)}_{0:t} \mid y_{1:t})}$

3. Estimate an arbitrary function $f(.)$:

   $E[f(x_{0:t} \mid y_{1:t})] \approx \sum_{i=1}^{N} f(x^{(i)}_{0:t})\tilde{w}^{(i)}_{t}$, \quad $\tilde{w}^{(i)}_{t} = \frac{w(x^{(i)}_{0:t})}{\sum_{j=1}^{N} w(x^{(j)}_{0:t})}$
Sequential Importance Sampling (SIS): Recursive Estimation

Augmenting the samples

\[
q(x_{0:t} \mid y_{1:t}) = q(x_{0:t-1} \mid y_{1:t-1})q(x_t \mid x_{0:t-1}, y_{1:t}) \\
= q(x_{0:t-1} \mid y_{1:t-1})q(x_t \mid x_{t-1}, y_t)
\]

\[x_t^{(i)} \sim q(x_t \mid x_{t-1}, y_t)\]

(cf. non-sequential IS: \(x_t^{(i)} \sim q(x_{0:t} \mid y_{1:t})\))

Weight update

\[
w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(y_t \mid x_t^{(i)})p(x_t^{(i)} \mid x_{t-1}^{(i)})}{q(x_t^{(i)} \mid x_{t-1}, y_t)}
\]
Sequential Importance Sampling: Idea

- Update filtering density using Bayesian filtering
- Compute integrals using importance sampling

- The filtering density $p(x_t | y_{1:t})$ is represented using particles and their weights
  \[
  \{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N
  \]
- Compute weights using:
  \[
  w_t^{(i)} = \frac{p(x_t^{(i)}, y_{1:t})}{q(x_t^{(i)}, y_{1:t})}
  \]
Sequential Importance Sampling: Procedure

1. Particle generation \( x_t^{(i)} \sim q(x_t \mid x_{t-1}^{(i)}, y_t) = p(x_t \mid x_{t-1}^{(i)}) \)

2. Weight computation \( w_t^{(i)} = w_{t-1}^{(i)} p(y_t \mid x_t^{(i)}) \)

   Weight normalization \( \tilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}} \)

3. Estimation computation \( E[f(x_t \mid y_{1:t})] = \sum_{i=1}^{N} f(x_t^{(i)}) \tilde{w}_t^{(i)} \)

Note: Step 1 above assumes the proposal density to be the prior. This does not use the information from observations. Alternatively, the proposal density could be

\( x_t^{(i)} \sim q(x_t \mid x_{t-1}^{(i)}, y_t) = p(x_t \mid x_{t-1}^{(i)}, y_t) \)

that minimizes the variance of \( w_t \) (Doucet et al., 1999).
Resampling

• SIS suffers from degeneracy problems, i.e. a small number of particles have big weights and the rest have extremely small values.

• Remedy: SIR introduces a selection (resampling) step to eliminate samples with low importance ratios (weights) and multiply samples with high importance ratios.

• Resampling maps the weighted random measure on to the equally weighted random measure by sampling uniformly with replacement from \( \{ x_{0:t}^{(i)} \}_{i=1}^{N} \) with probabilities \( \{ w_{t}^{(i)} \}_{i=1}^{N} \):

\[
\{ \tilde{x}_{0:t}^{(i)}, N^{-1} \}_{i=1}^{N} \sim \{ x_{0:t}^{(i)}, w_{t}^{(i)}(x_{0:t}^{(i)}) \}_{i=1}^{N}
\]
Sampling Importance Resampling (SIR) = Sequential Monte Carlo = Particle Filter

1. Initialize $t \leftarrow 0$
   - For $i = 1, \ldots, N$: sample $x_t^{(i)} \sim p(x_0)$, \quad $t \leftarrow 1$.

2. Importance sampling
   - For $i = 1, \ldots, N$: sample $x_t^{(i)} \sim q(x_t \mid x_{t-1}^{(i)}, y_t) = p(x_t \mid x_{t-1}^{(i)})$
     Let $x_{0:t}^{(i)} \triangleq (x_{0:t-1}^{(i)}, x_t^{(i)})$
   - For $i = 1, \ldots, N$: compute weights $w_t^{(i)} = p(y_t \mid x_t^{(i)})$
   - Normalize the weights: $\tilde{w}_t^{(i)} = w_t^{(i)}/\sum_{j=1}^{N} w_t^{(j)}$

3. Resampling
   - Resample with replacement $N$ particles $x_{0:t}^{(i)}$ according to the importance weights $w_t^{(i)}$, resulting in $\{\tilde{x}_{0:t}^{(i)}, N^{-1}\}_{i=1}^{N}$.
   - New particle population $\{x_{0:t}^{(i)}\}_{i=1}^{N} \leftarrow \{\tilde{x}_{0:t}^{(i)}\}_{i=1}^{N}$.
   - Set $t \leftarrow t + 1$ and go to step 2.
Particle Filters
Motivating Applications

- Robotics - SLAM and localization with a stereo camera: [http://www.youtube.com/watch?v=m3L8OfbTXH0&feature=related](http://www.youtube.com/watch?v=m3L8OfbTXH0&feature=related)
- Kalman filter result on real aircraft: [http://www.youtube.com/watch?v=0GSIKwfkFCA&feature=related](http://www.youtube.com/watch?v=0GSIKwfkFCA&feature=related)
Problem Statement

- Tracking the **state of a system** as it evolves over time
- We have: Sequentially arriving (noisy or ambiguous) observations
- We want to know: Best possible **estimate of the hidden variables**
Bayesian Filtering / Tracking Problem

- Unknown state vector \( x_{0:t} = (x_0, \ldots, x_t) \)
- Observation vector \( z_{1:t} \)
- Find PDF \( p(x_{0:t} | z_{1:t}) \) \( \ldots \) posterior distribution
- or \( p(x_t | z_{1:t}) \) \( \ldots \) filtering distribution

- Prior information given:
  - \( p(x_0) \) \( \ldots \) prior on state distribution
  - \( p(z_t | x_t) \) \( \ldots \) sensor model
  - \( p(x_t | x_{t-1}) \) \( \ldots \) Markovian state-space model
Sequential Update

- Storing all incoming measurements is inconvenient
- Recursive filtering:
  - Predict next state pdf from current estimate
  - Update the prediction using sequentially arriving new measurements
- Optimal Bayesian solution: recursively calculating exact posterior density
Bayesian Update and Prediction

- **Prediction**

\[
p(x_t \mid z_{1:t-1}) = \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid z_{1:t-1}) \, dx_{t-1}
\]

- **Update**

\[
p(x_t \mid z_{1:t}) = \frac{p(z_t \mid x_t) p(x_t \mid z_{1:t-1})}{p(z_t \mid z_{1:t-1})}
\]

\[
p(z_t \mid z_{1:t-1}) = \int p(z_t \mid x_t) p(x_t \mid z_{1:t-1}) \, dx_t
\]
Kalman Filter

- Optimal solution for linear-Gaussian case
- Assumptions:
  - State model is known linear function of last state and Gaussian noise signal
  - Sensory model is known linear function of state and Gaussian noise signal
  - Posterior density is Gaussian
Kalman Filter: Update Equations

\[ x_t = F_t x_{t-1} + v_{t-1} \quad v_{t-1} \sim N(0,Q_{t-1}) \]
\[ z_t = H_t x_t + n_t \quad n_t \sim N(0,R_t) \]

\( F_t, H_t \): known matrices

\[
\begin{align*}
p(x_{t-1} \mid z_{1:t-1}) &= N(x_{t-1} \mid m_{t-1|t-1}, P_{t-1|t-1}) \\
p(x_t \mid z_{1:t-1}) &= N(x_t \mid m_{t|t-1}, P_{t|t-1}) \\
p(x_t \mid z_{1:t}) &= N(x_t \mid m_{t|t}, P_{t|t})
\end{align*}
\]

\[
\begin{align*}
m_{t|t-1} &= F_t m_{t-1|t-1} \\
P_{t|t-1} &= Q_{t-1} + F_t P_{t-1|t-1} F_t^T \\
m_{t|t} &= m_{t|t-1} + K_t (z_t - H_t m_{t|t-1}) \\
P_{t|t} &= P_{t|t-1} - K_t H_t P_{t|t-1} \\
S_t &= H_t P_{t|t-1} H_t^T + R_t \\
K_t &= P_{t|t-1} H_t^T S_t^{-1}
\end{align*}
\]
Limitations of Kalman Filtering

- Assumptions are too strong. We often find:
  - Non-linear models
  - Non-Gaussian noise or posterior
  - Multi-modal distributions
  - Skewed distributions

- Extended Kalman Filter:
  - Local linearization of non-linear models
  - Still limited to Gaussian posterior
Grid-based Methods

- Optimal for discrete and finite state space
- Keep and update an estimate of posterior pdf for every single state
- No constraints on posterior (discrete) density
Limitations of Grid-based Methods

- Computationally expensive
- Only for finite state sets
- Approximate grid-based filter
  - Divide continuous state space into finite number of cells
  - Hidden Markov model filter
  - Dimensionality increases computational costs dramatically
Many different names...

Particle Filters

- (Sequential) Monte Carlo filters
- Bootstrap filters
- Condensation
- Interacting particle approximations
- Survival of the fittest
- …
Monte Carlo characterization of pdf:
- Represent posterior density by a set of random i.i.d. samples \( \text{(particles)} \) from the pdf \( p(x_{0:t}|z_{1:t}) \)
- For larger number \( N \) of particles equivalent to functional description of pdf
- For \( N \rightarrow \infty \) approaches optimal Bayesian estimate
Sample-based PDF Representation

- Regions of high density
  - Many particles
  - Large weight of particles
- Uneven partitioning
- Discrete approximation for continuous pdf

\[
P_N(x_{0:t} \mid z_{1:t}) = \sum_{i=1}^{N} w^i_t \delta(x_{0:t} - x^i_{0:t})
\]
Importance Sampling

- Draw N samples $x_{0:t}^{(i)}$ from importance sampling distribution $\pi(x_{0:t} | z_{1:t})$

- Importance weight

$$w(x_{0:t}) = \frac{p(x_{0:t} | z_{1:t})}{\pi(x_{0:t} | z_{1:t})}$$

- Estimation of arbitrary functions $f_t$:

$$\hat{I}_N(f_t) = \sum_{i=1}^{N} f_t(x_{0:t}^{(i)}) \tilde{w}_t^{(i)} , \quad \tilde{w}_t^{(i)} = \frac{w(x_{0:t}^{(i)})}{\sum_{j=1}^{N} w(x_{0:t}^{(j)})}$$

$$\hat{I}_N(f_t) \overset{a.s.}{\to} I(f_t) = \int f_t(x_{0:t}) p(x_{0:t} | y_{1:t}) dx_{0:t}$$
Sequential Importance Sampling (SIS)

- Augmenting the samples

\[
\pi(x_{0:t} \mid z_{1:t}) = \pi(x_{0:t-1} \mid z_{1:t-1}) \pi(x_t \mid x_{0:t-1}, z_{1:t}) = \\
= \pi(x_{0:t-1} \mid z_{1:t-1}) \pi(x_t \mid x_{t-1}, z_t)
\]

\(x_t^{(i)} \sim \pi(x_t \mid x_{t-1}^{(i)}, z_t)\)

- Weight update

\[
w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(z_t \mid x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)})}{\pi(x_t^{(i)} \mid x_{t-1}^{(i)}, z_t)}
\]
Degeneracy Problem

- After a few iterations, all but one particle will have negligible weight
- Measure for degeneracy: *effective sample size*

\[
N_{\text{eff}} = \frac{N}{1 + \text{Var}(w_t^*)}
\]

- Small \( N_{\text{eff}} \) indicates severe degeneracy
- Brute force solution: Use very large \( N \)
Choosing Importance Density

- Choose $\pi$ to minimize variance of weights
- Optimal solution:
  $$\pi_{opt}(x_t \mid x_{t-1}^{(i)}, z_t) = p(x_t \mid x_{t-1}^{(i)}, z_t)$$
  $$\Rightarrow w_t^{(i)} \propto w_{t-1}^{(i)} p(z_t \mid x_{t-1}^{(i)})$$

- Practical solution
  - Importance density = prior
  $$\pi(x_t \mid x_{t-1}^{(i)}, z_t) = p(x_t \mid x_{t-1}^{(i)})$$
  $$\Rightarrow w_t^{(i)} \propto w_{t-1}^{(i)} p(z_t \mid x_{t}^{(i)})$$
Resampling

- Eliminate particles with small importance weights
- Concentrate on particles with large weights

- Sample $N$ times with replacement from the set of particles $x_t^{(i)}$ according to importance weights $w_t^{(i)}$
- "Survival of the fittest"
Sampling Importance Resample Filter: Basic Algorithm

1. INIT, t=0
   - for i=1,..., N: sample $x_0^{(i)} \sim p(x_0)$; t:=1;

2. IMPORTANCE SAMPLING
   - for i=1,..., N: sample $x_t^{(i)} \sim p(x_t|x_{t-1}^{(i)})$
     - $x_{0:t}^{(i)} := (x_{0:t-1}^{(i)}, x_t^{(i)})$
   - for i=1,..., N: evaluate importance weights $w_t^{(i)} = p(z_t|x_t^{(i)})$
   - Normalize the importance weights

3. SELECTION / RESAMPLING
   - resample with replacement N particles $x_{0:t}^{(i)}$ according to the importance weights
   - Set t:=t+1 and go to step 2
Variations

- Auxiliary Particle Filter:
  - Resample at time $t-1$ with one-step lookahead (re-evaluate with new sensory information)

- Regularisation:
  - Resample from continuous approximation of posterior $p(x_t|z_{1:t})$
Visualization of Particle Filter

unweighted measure

compute importance weights $\Rightarrow p(x_{t-1}|z_{1:t-1})$

resampling

move particles

predict $p(x_t|z_{1:t-1})$
Particle Filter Demo 1

moving Gaussian + uniform, N=100 particles
Particle Filter Demo 2

moving Gaussian + uniform, N=1000 particles
Particle Filter Demo 3

moving (sharp) Gaussian + uniform, N=100 particles
Particle Filter Demo 4

moving (sharp) Gaussian + uniform, N=1000 particles
mixture of two Gaussians, filter loses track of smaller and less pronounced peaks
Obtaining state estimates from particles

- Any estimate of a function $f(x_t)$ can be calculated by discrete PDF-approximation

  $$E[f(x_t)] = \frac{1}{N} \sum_{j=1}^{N} w_t^{(j)} f(x_t^{(j)})$$

- Mean:

  $$E[x_t] = \frac{1}{N} \sum_{j=1}^{N} w_t^{(j)} x_t^{(j)}$$

- MAP-estimate: particle with largest weight

- Robust mean: mean within window around MAP-estimate
Pros and Cons of Particle Filters

+ Estimation of full PDFs
+ Non-Gaussian distributions
  - e.g. multi-modal
+ Non-linear state and observation model
+ Parallelizable
- Degeneracy problem
- High number of particles needed
- Computationally expensive
- Linear-Gaussian assumption is often sufficient
Mobile Robot Localization

- Animation by Sebastian Thrun, Stanford
- [http://robots.stanford.edu](http://robots.stanford.edu)
Model Estimation

- Tracking with multiple motion-models
  - Discrete hidden variable indicates active model (manoeuvre)

- Recovery of signal from noisy measurements
  - Even if signal may be absent (e.g. synaptic currents)
  - Mixture model of several hypotheses

- Neural Network model selection [de Freitas]¹
  - Estimate parameters and architecture of RBF network from input-output pairs
  - On-line classification (time-varying classes)

---

¹ de Freitas, et.al.: Sequential Monte Carlo Methods for Neural Networks. in: Doucet, et.al.: Sequential Monte Carlo Methods in Practice, Springer Verlag, 2001
Other Applications

- Visual Tracking
  - e.g. human motion (body parts)
- Prediction of (financial) time series
  - e.g. mapping gold price $\rightarrow$ stock price
- Quality control in semiconductor industry
- Military applications
  - Target recognition from single or multiple images
  - Guidance of missiles