3D Computer Vision

Radim Šára Martin Matoušek

Center for Machine Perception
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

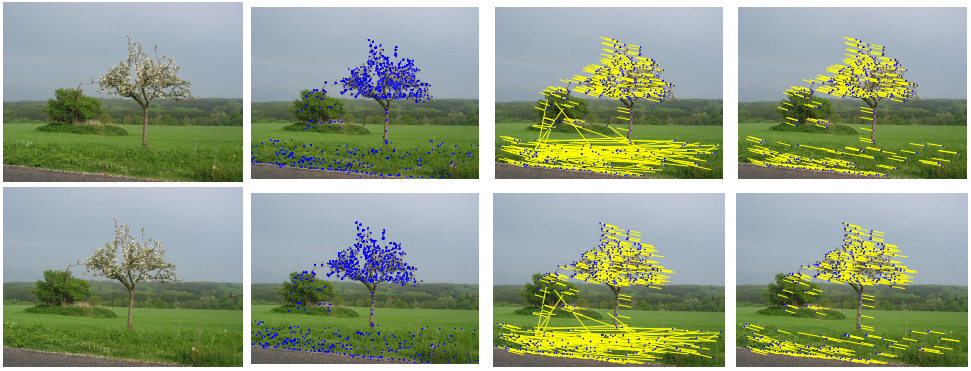
https://cw.fel.cvut.cz/wiki/courses/tdv/start

http://cmp.felk.cvut.cz
mailto:sara@cmp.felk.cvut.cz
phone ext. 7203

rev. November 8, 2022

Open Informatics Master's Course

Example Matching Results for the 7-point Algorithm with RANSAC



input images | interest points (ca. 3600) | tentative corresp. (416) | matching (340)

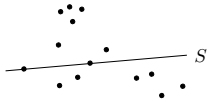
= matches

- no descriptors used
- without local optimization the minimization is over a discrete set of epipolar geometries proposable from 7-tuples
- notice some wrong matches (they have wrong depth, even negative)
- they cannot be rejected without additional constraints or scene knowledge

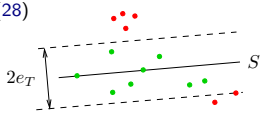
remember: hidden labels →111

▶A Preview: RANSAC with Local Optimization and Early Stopping

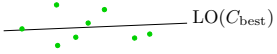
1. initialize the best configuration as empty $C_{\text{best}} := \emptyset$ and proposal index $k := 0$
2. estimate the total number of needed proposals as $N := \binom{n}{s}$ — n – No. of primitives, s – minimal config size
3. while $k \leq N$:
   a) **propose** a minimal random configuration S of size s from $q(S)$



   b) if $\pi(S) > \pi(C_{\text{best}})$ then **accept**
      i) update the best config $C_{\text{best}} := S$ — $\pi(S)$ marginalized as in (27); $\pi(S)$ includes a prior ⇒ MAP
      ii) threshold-out inliers using e_T from (28)



      iii) locally **optimize** from the inliers of C_{best} — LM optimization with robustified (→117) Sampson error possibly weighted by posterior $\pi(m_{ij})$ [Chum et al. 2003]



      iv) update C_{best}, update inliers using (28), re-estimate N from inlier counts — →126 for derivation

$$N = \frac{\log(1-P)}{\log(1-\varepsilon^s)}, \quad \varepsilon = \frac{|\,\text{inliers}(C_{\text{best}})|}{n},$$

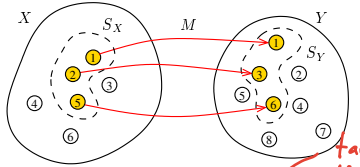
   c) $k := k + 1$
4. output C_{best}

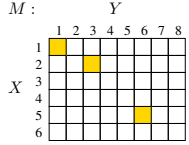
- see MPV course for RANSAC details — see also [Fischler & Bolles 1981], [25 years of RANSAC]

▶Towards $\pi(S)$: The Full Problem of Matching and Fundamental Matrix Estimation

Problem: Given image keypoint sets $X = \{x_i\}_{i=1}^{m}$ and $Y = \{y_j\}_{j=1}^{n}$ and their descriptors D, find the most probable

1. inlier keypoints $S_X \subseteq X$, $S_Y \subseteq Y$
2. one-to-one perfect matching $M\colon S_X \to S_Y$ — perfect matching: 1-factor of the bipartite graph
3. fundamental matrix $\mathbf{F}$ such that $\operatorname{rank}\mathbf{F} = 2$
4. such that for each $x_i \in S_X$ and $y_j = M(x_i)$ it is probable that
   a) the image descriptor $D(x_i)$ is similar to $D(y_j)$, and
   b) the total reprojection error $E = \sum_{ij} e_{ij}^2(\mathbf{F})$ is small — note a slight change in notation: e_{ij}
5. inlier-outlier and outlier-outlier matches are improbable





target distrib.

$2^n >> n!$

$$(M^*, \mathbf{F}^*) = \arg\max_{M,\mathbf{F}} \pi(E, D, \mathbf{F}, M) = \arg\max_{M,\mathbf{F}} p(E, D, \mathbf{F} \mid M)\, P(M) \tag{22}$$

- probabilistic model: an efficient language for problem formulation — it also unifies 4.a and 4.b
- the (22) is a Bayesian probabilistic model — there is a constant number of random variables!
- binary matching table $M_{ij} \in \{0, 1\}$ of fixed size $m \times n$
  - each row/column contains at most one unity
  - zero rows/columns correspond to unmatched point x_i / y_j

Deriving A Robust Matching Model by Approximate Marginalization

For algorithmic efficiency, instead of $(M^*, \mathbf{F}^*) = \arg\max_{M,\mathbf{F}} p(E, D, \mathbf{F} \mid M)\, P(M)$ solve

$$\mathbf{F}^* = \arg\max_{\mathbf{F}} p(E, D, \mathbf{F}) \tag{23}$$

by marginalization of $p(E, D, \mathbf{F} \mid M)\, P(M)$ over the set of all matchings $\mathcal{M}$ s.t. $M \in \mathcal{M}$ this changes the problem!

drop the assumption that M is a 1:1 matching, assume correspondence-wise independence:

$$p(E, D, \mathbf{F} \mid M) P(M) = \prod_{i=1}^{m} \prod_{j=1}^{n} p_e(e_{ij}, d_{ij}, \mathbf{F} \mid m_{ij}) P(m_{ij})$$

- e_{ij} represents (reprojection) error for match $x_i \leftrightarrow y_i$: e.g. $e_{ij}(x_i, y_i, \mathbf{F})$
- d_{ij} represents descriptor similarity for match $x_i \leftrightarrow y_i$: e.g. $d_{ij} = \|\mathbf{d}(x_i) - \mathbf{d}(y_j)\|$

Approximate marginalization: take all the 2^{mn} terms in place of M

$$p(E, D, \mathbf{F}) \approx \sum_{m_{11} \in \{0,1\}} \sum_{m_{12}} \cdots \sum_{m_{mn}} p(E, D, \mathbf{F} \mid M) P(M) =$$

$$= \sum_{m_{11}} \cdots \sum_{m_{mn}} \prod_{i=1}^{m} \prod_{j=1}^{n} p_e(e_{ij}, d_{ij}, \mathbf{F} \mid m_{ij}) P(m_{ij}) \;=\; \circledast 1 \;=\; \cdots =$$

$$= \prod_{i=1}^{m} \prod_{j=1}^{n} \underbrace{\sum_{m_{ij} \in \{0,1\}} p_e(e_{ij}, d_{ij}, \mathbf{F} \mid m_{ij}) P(m_{ij})}_{\text{we will continue with this term}} \tag{24}$$

Robust Matching Model (cont'd)

$$\sum_{m_{ij}\in\{0,1\}} p_e(e_{ij}, d_{ij}, \mathbf{F} \mid m_{ij}) P(m_{ij}) =$$

$$= \underbrace{p_e(e_{ij}, d_{ij}, \mathbf{F} \mid m_{ij}=1)}_{p_1(e_{ij},d_{ij},\mathbf{F})} \underbrace{P(m_{ij}=1)}_{1-P_0} + \underbrace{p_e(e_{ij}, d_{ij}, \mathbf{F} \mid m_{ij}=0)}_{p_0(e_{ij},d_{ij},\mathbf{F})} \underbrace{P(m_{ij}=0)}_{P_0} =$$

$$= (1-P_0)\, p_1(e_{ij}, d_{ij}, \mathbf{F}) + P_0\, p_0(e_{ij}, d_{ij}, \mathbf{F}) \qquad (25)$$

- the $p_0(e_{ij}, d_{ij}, \mathbf{F})$ is a penalty for 'missing a correspondence' but it should be a p.d.f. (cannot be a constant) (→117 for a simplification)

$$\text{choose} \quad P_0 \to 1, \quad p_0(\cdot) \to 0 \quad \text{so that} \quad \frac{P_0}{1-P_0}\, p_0(\cdot) \approx \text{const}$$

- the $p_1(e_{ij}, d_{ij}, \mathbf{F})$ is typically an easy-to-design term: assuming independence of reprojection error and descriptor similarity:

$$p_1(e_{ij}, d_{ij}, \mathbf{F}) = p_1(e_{ij} \mid \mathbf{F})\, p_F(\mathbf{F})\, p_1(d_{ij})$$

- we choose, e.g.

$$p_1(e_{ij} \mid \mathbf{F}) = \frac{1}{T_e(\sigma_1)}\, e^{-\frac{e_{ij}^2(\mathbf{F})}{2\sigma_1^2}}, \quad p_1(d_{ij}) = \frac{1}{T_d(\sigma_d, \dim \mathbf{d})}\, e^{-\frac{\|\mathbf{d}(x_i)-\mathbf{d}(y_j)\|^2}{2\sigma_d^2}} \qquad (26)$$

- $\mathbf{F}$ is a random variable and σ_1, σ_d, P_0 are parameters
- the form of $T_e(\sigma_1)$ depends on the error definition, it may depend on x_i, y_j but not on $\mathbf{F}$
- we will continue with the result from (25)

▶Simplified Robust Energy (Error) Function

- assuming the choice of p_1 as in (26), we are simplifying (24) to

$$p(E, D, \mathbf{F}) = p(E, D \mid \mathbf{F})\, p_F(\mathbf{F}) = p_F(\mathbf{F}) \prod_{i=1}^{m} \prod_{j=1}^{n} \Big[(1 - P_0)\, p_1(e_{ij}, d_{ij} \mid \mathbf{F}) + P_0\, p_0(e_{ij}, d_{ij} \mid \mathbf{F}) \Big]$$

- we choose $\sigma_0 \gg \sigma_1$ and omit d_{ij} for simplicity; then the square-bracket term is

$$\frac{1 - P_0}{T_e(\sigma_1)}\, e^{-\frac{e_{ij}^2(\mathbf{F})}{2\sigma_1^2}} + \frac{P_0}{T_e(\sigma_0)}\, e^{-\frac{e_{ij}^2(\mathbf{F})}{2\sigma_0^2}} = \frac{1 - P_0}{T_e(\sigma_1)} \left(e^{-\frac{e_{ij}^2(\mathbf{F})}{2\sigma_1^2}} + \frac{T_e(\sigma_1)}{1 - P_0} \frac{P_0}{T_e(\sigma_0)}\, e^{-\frac{e_{ij}^2(\mathbf{F})}{2\sigma_0^2}} \right)$$

- we define the 'potential function' as: $V(x) = -\log p(x)$, then we maximize

$$V(E, D \mid \mathbf{F}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left[\underbrace{-\log \frac{1 - P_0}{T_e(\sigma_1)}}_{\Delta\, =\, \text{const}} - \log\Big(e^{-\frac{e_{ij}^2(\mathbf{F})}{2\sigma_1^2}} + \underbrace{\frac{P_0}{1 - P_0} \frac{T_e(\sigma_1)}{T_e(\sigma_0)}\, e^{-\frac{e_{ij}^2(\mathbf{F})}{2\sigma_0^2}}}_{t\, \approx\, \text{const}} \Big) \right] =$$

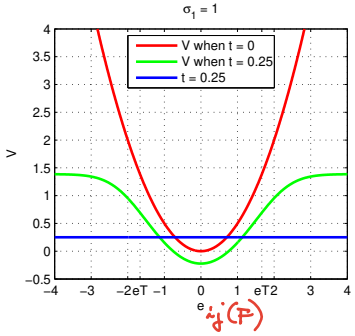
$$= m\, n\, \Delta + \sum_{i=1}^{m} \sum_{j=1}^{n} \underbrace{-\log\Big(e^{-\frac{e_{ij}^2(\mathbf{F})}{2\sigma_1^2}} + t \Big)}_{\hat{V}(e_{ij})} \qquad (27)$$

- the terms in (27) are: (constant) + (total robust error for all pairs in M)
- note we are summing over all $m\, n$ matches (m, n are constant!)
- when $t = 0$ we have quadratic inlier error function $\hat{V}(e_{ij}) = e_{ij}^2(\mathbf{F})/(2\sigma_1^2)$

expensive but explicit matching is avoided

▶The Action of the Robust Matching Model on Data

Example for $\hat{V}(e_{ij})$ from (27):



red – the (non-robust) quadratic error $\hat{V}(e_{ij})$ when $t = 0$

blue – the rejected match penalty t

green – robust $\hat{V}(e_{ij})$ from (27)

- if the error of a correspondence exceeds a limit, it is ignored
- then $\hat{V}(e_{ij}) = \text{const}$ and we just count outliers in (27)
- t controls the 'turn-off' point
- the inlier/outlier threshold is e_T – the error for which $(1 - P_0)\, p_1(e_T) = P_0\, p_0(e_T)$: note that $t \approx 0$

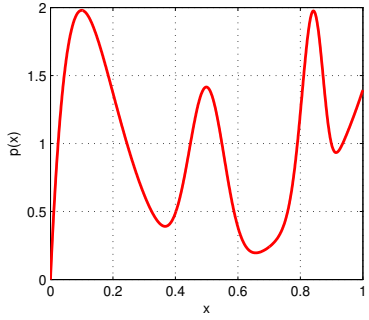
$$e_T = \sigma_1 \sqrt{-\log t^2},\ t = e^{-\frac{1}{2}\left(\frac{e_T}{\sigma_1}\right)^2} \text{ e.g. } e_T = 4\sigma_1 \rightarrow t \approx 3.4 \cdot 10^{-4} \quad (28)$$

The full optimization problem (23) uses (27):

$$\mathbf{F}^* = \arg\max_{\mathbf{F}} \frac{\overbrace{p(E, D \mid \mathbf{F})}^{\text{data model}} \cdot \overbrace{p(\mathbf{F})}^{\text{prior}}}{\underbrace{p(E, D)}_{\text{evidence}}} \approx \arg\min_{\mathbf{F}} \left[V(\mathbf{F}) + \sum_{i=1}^{m} \sum_{j=1}^{n} \log\left(e^{-\frac{e_{ij}^2(\mathbf{F})}{2\sigma_1^2}} + t \right) \right]$$

- typically we take $V(\mathbf{F}) = -\log p(\mathbf{F}) = 0$ unless we need to stabilize a computation, e.g. when video camera moves smoothly (on a high-mass vehicle) and we have a prediction for $\mathbf{F}$
- the evidence is not needed unless we want to compare different models (e.g. homography vs. epipolar geometry)

How To Find the Global Maxima (Modes) of a PDF?



- given a toy probability distribution $p(x)$ at left — p.d.f. on $[0, 1]$, mode at 0.1

consider several methods:

1. exhaustive search

```
step = 1/(iterations-1);
for x = 0:step:1
 if p(x) > bestp
  bestx = x; bestp = p(x);
 end
end
```

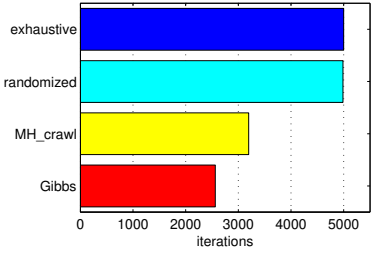
- slow algorithm (definite quantization)
- fast to implement

2. randomized search with uniform sampling

```
while t < iterations
 x = rand(1);
 if p(x) > bestp
  bestx = x; bestp = p(x);
 end
 t = t+1;  % time
end
```

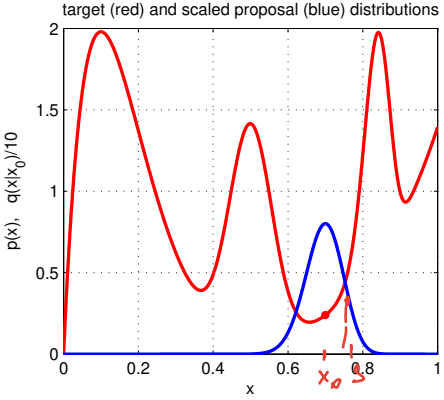
- equally slow algorithm
- fast to implement

3. random sampling from $p(x)$ (Gibbs sampler)
   - faster algorithm • fast to implement but often infeasible (e.g. when $p(x)$ is data dependent (our case in correspondence prob.))
4. Metropolis-Hastings sampling
   - almost as fast (with care) • not so fast to implement
   - rarely infeasible • RANSAC belongs here



- averaged over 10^4 trials
- simpler (unimodal) distributions result in faster convergence

How To Generate Random Samples from a Complex Distribution?



- red: probability density function $\pi(x)$ of the toy distribution on the unit interval — target distribution

$$\pi(x) = \sum_{i=1}^{4} \gamma_i \, \mathrm{Be}(x; \alpha_i, \beta_i), \quad \sum_{i=1}^{4} \gamma_i = 1, \ \gamma_i \geq 0$$

$$\mathrm{Be}(x; \alpha, \beta) = \frac{1}{\mathrm{B}(\alpha, \beta)} \cdot x^{\alpha-1}(1-x)^{\beta-1}, \quad \alpha, \beta \geq 0$$

- alg. for generating samples from $\mathrm{Be}(x; \alpha, \beta)$ is known
- $\Rightarrow$ we can generate samples from $\pi(x)$ — how?

- suppose we cannot sample from $\pi(x)$ but we can sample from some 'simple' proposal distribution $q(x \mid x_0)$, given the previous sample x_0 (blue)

$$q(x \mid x_0) = \begin{cases} \mathrm{U}_{0,1}(x) & \text{(independent) uniform sampling } = \mathrm{Be}(x, 1, 1) \\ \mathrm{Be}(x; \frac{x_0}{T} + 1, \frac{1-x_0}{T} + 1) & \text{'beta' diffusion (crawler)} \quad T - \text{temperature} \\ \pi(x) & \text{(independent) Gibbs sampler} \end{cases}$$

- note we have unified all the random sampling methods from the previous slide
- how to redistribute proposal samples $q(x \mid x_0)$ to target distribution $\pi(x)$ samples?

▶Metropolis-Hastings (MH) Sampling

C, S – configurations (of all variable values) e.g. $C = x$ and $\pi(C) = \pi(x)$ from →120

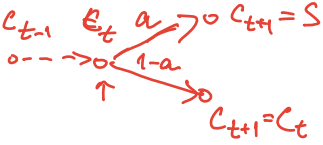
Goal: Generate a sequence of random samples $\{C_t\}$ from target distribution $\pi(C)$

- setup a Markov chain with a suitable transition probability to generate the sequence

Sampling procedure

1. given current configuration C_t, propose (draw a random) configuration sample S from $q(S \mid C_t)$
   q may use some information from C_t (Hastings)
2. compute acceptance probability
   the redistribution filter; note the evidence term drops out

$$a = \min\left\{1,\ \frac{\pi(S)}{\pi(C_t)} \cdot \frac{q(C_t \mid S)}{q(S \mid C_t)}\right\}$$



3. accept S with probability a
   a) draw a random number u from unit-interval uniform distribution $\mathrm{U}_{0,1}$
   b) if $u \leq a$ then $C_{t+1} := S$ else $C_{t+1} := C_t$

'Programming' an MH sampler

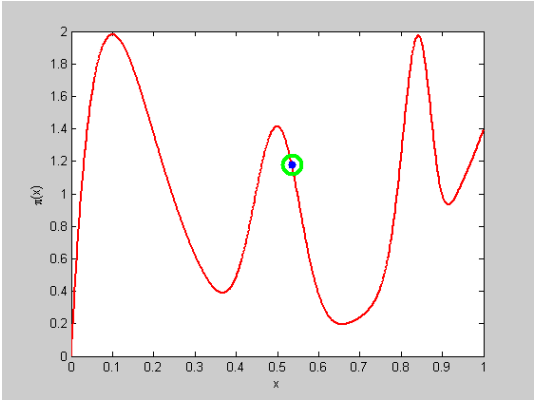
1. design a proposal distribution (mixture) q and a sampler from q
2. express functions $q(C_t \mid S)$ and $q(S \mid C_t)$ as proper distributions
   not always simple

Finding the mode

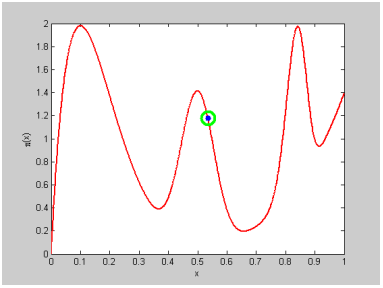
- remember the best sample
  fast implementation but must wait long to hit the mode
- use simulated annealing
  very slow
- use the sampler as an explorer and do local optimization from the accepted sample
  a trade-off between speed and accuracy

an optimal algorithm does not use just the best sample: a Stochastic EM Algorithm (e.g. SAEM)

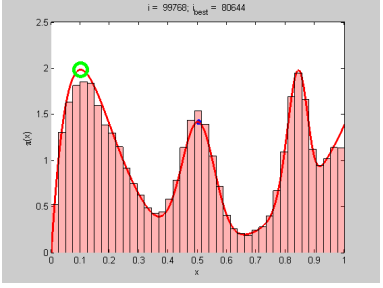
MH Sampling Demo



sampling process (100k samples; video, 7:33) click for video



initial sample



- blue point: current sample
- green circle: best sample so far quality $= \pi(x)$
- histogram: current distribution of visited states
- the vicinity of modes are the most often visited states

Demo Source Code (Matlab)

```
function x = proposal_gen(x0)
% proposal generator q(x | x0)

 T = 0.01; % temperature
 x = betarnd(x0/T+1,(1-x0)/T+1);
end


function p = proposal_q(x, x0)
% proposal distribution q(x | x0)

 T = 0.01;
 p = betapdf(x, x0/T+1, (1-x0)/T+1);
end


function p = target_p(x)
% target distribution p(x)

 % shape parameters:
 a = [2   40   100 6];
 b = [10  40   20  1];

 % mixing coefficients:
 w = [1 0.4 0.253 0.50]; w = w/sum(w);
 p = 0;
 for i = 1:length(a)
  p = p + w(i)*betapdf(x,a(i),b(i));
 end
end
```

```
%% DEMO script

k = 10000;    % number of samples
X = NaN(1,k); % list of samples

x0 = proposal_gen(0.5);
for i = 1:k
 x1 = proposal_gen(x0);
 a = target_p(x1)/target_p(x0) * ...
     proposal_q(x0,x1)/proposal_q(x1,x0);
 if rand(1) < a
  X(i) = x1;  x0 = x1;
 else
  X(i) = x0;
 end
end

figure(1)
x = 0:0.001:1;
plot(x, target_p(x), 'r', 'linewidth',2);
hold on
binw = 0.025; % histogram bin width
n = histc(X, 0:binw:1);
h = bar(0:binw:1, n/sum(n)/binw, 'histc');
set(h, 'facecolor', 'r', 'facealpha', 0.3)
xlim([0 1]); ylim([0 2.5])
xlabel 'x'
ylabel 'p(x)'
title 'MH demo'
hold off
```

Thank You