

► Sampson Error for Fundamental Matrix Manifold

The epipolar algebraic error is

$$\varepsilon_i(\mathbf{F}) = \underline{\mathbf{y}}_i^\top \mathbf{F} \underline{\mathbf{x}}_i, \quad \mathbf{x}_i = (u_i^1, v_i^1), \quad \mathbf{y}_i = (u_i^2, v_i^2), \quad \varepsilon_i \in \mathbb{R}$$

Let $\mathbf{F} = [\mathbf{F}_1 \quad \mathbf{F}_2 \quad \mathbf{F}_3]$ (per columns) = $\begin{bmatrix} (\mathbf{F}^1)^\top \\ (\mathbf{F}^2)^\top \\ (\mathbf{F}^3)^\top \end{bmatrix}$ (per rows), $\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, then

Sampson

$$\mathbf{J}_i(\mathbf{F}) = \left[\frac{\partial \varepsilon_i(\mathbf{F})}{\partial u_i^1}, \frac{\partial \varepsilon_i(\mathbf{F})}{\partial v_i^1}, \frac{\partial \varepsilon_i(\mathbf{F})}{\partial u_i^2}, \frac{\partial \varepsilon_i(\mathbf{F})}{\partial v_i^2} \right] \quad \mathbf{J}_i \in \mathbb{R}^{1,4} \quad \text{derivatives over point coords.}$$
$$= [(\mathbf{F}_1)^\top \underline{\mathbf{y}}_i, (\mathbf{F}_2)^\top \underline{\mathbf{y}}_i, (\mathbf{F}^1)^\top \underline{\mathbf{x}}_i, (\mathbf{F}^2)^\top \underline{\mathbf{x}}_i]$$

$$\mathbf{e}_i(\mathbf{F}) = -\frac{\mathbf{J}_i(\mathbf{F}) \varepsilon_i(\mathbf{F})}{\|\mathbf{J}_i(\mathbf{F})\|^2} \quad \mathbf{e}_i(\mathbf{F}) \in \mathbb{R}^4 \quad \text{Sampson error vector}$$

$$e_i(\mathbf{F}) = \|\mathbf{e}_i(\mathbf{F})\| = \frac{\varepsilon_i(\mathbf{F})}{\|\mathbf{J}_i(\mathbf{F})\|} = \frac{\underline{\mathbf{y}}_i^\top \mathbf{F} \underline{\mathbf{x}}_i}{\sqrt{\|\mathbf{S} \mathbf{F} \underline{\mathbf{x}}_i\|^2 + \|\mathbf{S} \mathbf{F}^\top \underline{\mathbf{y}}_i\|^2}} \quad e_i(\mathbf{F}) \in \mathbb{R} \quad \text{scalar Sampson error}$$

- Sampson error 'normalizes' the algebraic error
- automatically copes with multiplicative factors $\mathbf{F} \mapsto \lambda \mathbf{F}$
- actual optimization not yet covered →109

► Back to Triangulation: The Golden Standard Method

Given $\mathbf{P}_1, \mathbf{P}_2$ and a correspondence $x \leftrightarrow y$, look for 3D point \mathbf{X} projecting to x and y . →89

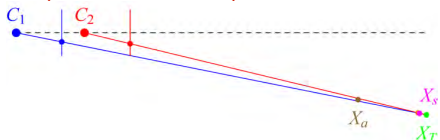
Idea:

1. if not given, compute \mathbf{F} from $\mathbf{P}_1, \mathbf{P}_2$, e.g. $\mathbf{F} = (\mathbf{Q}_1 \mathbf{Q}_2^{-1})^\top [\mathbf{q}_1 - (\mathbf{Q}_1 \mathbf{Q}_2^{-1}) \mathbf{q}_2]_\times$
2. correct measurement by the linear estimate of the correction vector →100

$$\begin{bmatrix} \hat{u}^1 \\ \hat{v}^1 \\ \hat{u}^2 \\ \hat{v}^2 \end{bmatrix} \approx \begin{bmatrix} u^1 \\ v^1 \\ u^2 \\ v^2 \end{bmatrix} - \frac{\varepsilon}{\|\mathbf{J}\|^2} \mathbf{J}^\top = \begin{bmatrix} u^1 \\ v^1 \\ u^2 \\ v^2 \end{bmatrix} - \frac{\underline{\mathbf{y}}^\top \mathbf{F} \underline{\mathbf{x}}}{\|\mathbf{S} \mathbf{F} \underline{\mathbf{x}}\|^2 + \|\mathbf{S} \mathbf{F}^\top \underline{\mathbf{y}}\|^2} \begin{bmatrix} (\mathbf{F}_1)^\top \underline{\mathbf{y}} \\ (\mathbf{F}_2)^\top \underline{\mathbf{y}} \\ (\mathbf{F}^1)^\top \underline{\mathbf{x}} \\ (\mathbf{F}^2)^\top \underline{\mathbf{x}} \end{bmatrix}$$

3. use the SVD triangulation algorithm with numerical conditioning →90; iteration possible

Ex (cont'd from →93):



- X_T – noiseless ground truth position
- – reprojection error minimizer
- X_s – Sampson-corrected algebraic error minimizer
- X_a – algebraic error minimizer
- m – measurement (m_T with noise in v^2)



► Back to Fundamental Matrix Estimation

Goal: Given a set $X = \{(x_i, y_i)\}_{i=1}^k$ of $k \gg 7$ inlier correspondences, compute a statistically efficient estimate for fundamental matrix \mathbf{F} .

What we have so far

- 7-point algorithm for \mathbf{F} (5-point algorithm for \mathbf{E}) → 84
- definition of Sampson error per correspondence $e_i(\mathbf{F} \mid x_i, y_i)$ → 104

What we need

- an optimization algorithm for

$$\mathbf{F}^* = \arg \min_{\mathbf{F}} \sum_{i=1}^k e_i^2(\mathbf{F} \mid X)$$

- the 7-point estimate is a good starting point \mathbf{F}_0

Levenberg-Marquardt (LM) Iterative Estimation in a Nutshell

Consider error function $\mathbf{e}_i(\boldsymbol{\theta}) = f(\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\theta}) \in \mathbb{R}^m$, with $\mathbf{x}_i, \mathbf{y}_i$ given, $\boldsymbol{\theta} \in \mathbb{R}^q$ unknown
 $\theta = \mathbf{F}$, $q = 9$, $m = 1$ for f.m. estimation

Our goal: $\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \sum_{i=1}^k \|\mathbf{e}_i(\boldsymbol{\theta})\|^2$

Idea 1 (Gauss-Newton approximation): proceed iteratively for $s = 0, 1, 2, \dots$

$$\boldsymbol{\theta}^{s+1} := \boldsymbol{\theta}^s + \mathbf{d}_s, \quad \text{where } \mathbf{d}_s = \arg \min_{\mathbf{d}} \sum_{i=1}^k \|\mathbf{e}_i(\boldsymbol{\theta}^s + \mathbf{d})\|^2 \quad (20)$$

$$\mathbf{e}_i(\boldsymbol{\theta}^s + \mathbf{d}) \approx \mathbf{e}_i(\boldsymbol{\theta}^s) + \mathbf{L}_i \mathbf{d},$$

$$(\mathbf{L}_i)_{jl} = \frac{\partial (\mathbf{e}_i(\boldsymbol{\theta}))_j}{\partial (\boldsymbol{\theta})_l}, \quad \mathbf{L}_i \in \mathbb{R}^{m,q} \quad \text{typically a long matrix}$$

Then the solution to Problem (20) is a set of normal eqs

$$-\underbrace{\sum_{i=1}^k \mathbf{L}_i^\top \mathbf{e}_i(\boldsymbol{\theta}^s)}_{\mathbf{e} \in \mathbb{R}^{q,1}} = \underbrace{\left(\sum_{i=1}^k \mathbf{L}_i^\top \mathbf{L}_i \right)}_{\mathbf{L} \in \mathbb{R}^{q,q}} \mathbf{d}_s, \quad (21)$$

- \mathbf{d}_s can be solved for by Gaussian elimination using Choleski decomposition of \mathbf{L}
 \mathbf{L} symmetric \Rightarrow use Choleski, almost $2\times$ faster than Gauss-Seidel, see bundle adjustment $\rightarrow 135$
- such updates do not lead to stable convergence \rightarrow ideas of Levenberg and Marquardt

Idea 2 (Levenberg): replace $\sum_i \mathbf{L}_i^\top \mathbf{L}_i$ with $\sum_i \mathbf{L}_i^\top \mathbf{L}_i + \lambda \mathbf{I}$ for some damping factor $\lambda \geq 0$

Idea 3 (Marquardt): replace $\lambda \mathbf{I}$ with $\lambda \sum_i \text{diag}(\mathbf{L}_i^\top \mathbf{L}_i)$ to adapt to local curvature:

$$-\sum_{i=1}^k \mathbf{L}_i^\top \mathbf{e}_i(\boldsymbol{\theta}^s) = \left(\sum_{i=1}^k (\mathbf{L}_i^\top \mathbf{L}_i + \lambda \text{diag}(\mathbf{L}_i^\top \mathbf{L}_i)) \right) \mathbf{d}_s$$

Idea 4 (Marquardt): adaptive λ small $\lambda \rightarrow$ Gauss-Newton, large $\lambda \rightarrow$ gradient descend

1. choose $\lambda \approx 10^{-3}$ and compute \mathbf{d}_s
2. if $\sum_i \|\mathbf{e}_i(\boldsymbol{\theta}^s + \mathbf{d}_s)\|^2 < \sum_i \|\mathbf{e}_i(\boldsymbol{\theta}^s)\|^2$ then accept \mathbf{d}_s and set $\lambda := \lambda/10$, $s := s + 1$
3. otherwise set $\lambda := 10\lambda$ and recompute \mathbf{d}_s

- sometimes different constants are needed for the 10 and 10^{-3}
- note that $\mathbf{L}_i \in \mathbb{R}^{m,q}$ (long matrix) but each contribution $\mathbf{L}_i^\top \mathbf{L}_i$ is a square singular $q \times q$ matrix (always singular for $k < q$)
- error can be made robust to outliers, see the trick $\rightarrow 112$
- we have approximated the least squares Hessian by ignoring second derivatives of the error function (Gauss-Newton approximation) See [Triggs et al. 1999, Sec. 4.3]
- λ helps avoid the consequences of gauge freedom $\rightarrow 137$

LM with Sampson Error for Fundamental Matrix Estimation

Sampson (derived by linearization over point coordinates u^1, v^1, u^2, v^2)

$$e_i(\mathbf{F}) = \frac{\varepsilon_i}{\|\mathbf{J}_i\|} = \frac{\underline{\mathbf{y}}_i^\top \mathbf{F} \underline{\mathbf{x}}_i}{\sqrt{\|\mathbf{S} \mathbf{F} \underline{\mathbf{x}}_i\|^2 + \|\mathbf{S} \mathbf{F}^\top \underline{\mathbf{y}}_i\|^2}} \quad \text{where} \quad \mathbf{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

LM (by linearization over parameters \mathbf{F})

$$\mathbf{L}_i = \frac{\partial e_i(\mathbf{F})}{\partial \mathbf{F}} = \dots = \frac{1}{2\|\mathbf{J}_i\|} \left[\left(\underline{\mathbf{y}}_i - \frac{2e_i}{\|\mathbf{J}_i\|} \mathbf{S} \mathbf{F} \underline{\mathbf{x}}_i \right) \underline{\mathbf{x}}_i^\top + \underline{\mathbf{y}}_i \left(\underline{\mathbf{x}}_i - \frac{2e_i}{\|\mathbf{J}_i\|} \mathbf{S} \mathbf{F}^\top \underline{\mathbf{y}}_i \right)^\top \right] \quad (22)$$

- \mathbf{L}_i from (22) is a 3×3 matrix, must be reshaped to dimension-9 vector $\text{vec}(\mathbf{L}_i)$
- $\underline{\mathbf{x}}_i$ and $\underline{\mathbf{y}}_i$ in Sampson error are normalized to unit homogeneous coordinate (22) relies on this
- reinforce rank $\mathbf{F} = 2$ after each LM update to stay in the fundamental matrix manifold and $\|\mathbf{F}\| = 1$ to avoid gauge freedom by SVD $\rightarrow 110$
- LM linearization could be done by numerical differentiation (small dimension)

► Local Optimization for Fundamental Matrix Estimation

Given a set $X = \{(x_i, y_i)\}_{i=1}^k$ of $k \gg 7$ inlier correspondences, compute a statistically efficient estimate for fundamental matrix \mathbf{F} .

Summary so far

1. Find the conditioned ($\rightarrow 92$) 7-point \mathbf{F}_0 ($\rightarrow 84$) from a suitable 7-tuple
2. Improve the \mathbf{F}_0^* using the LM optimization ($\rightarrow 107-108$) and the Sampson error ($\rightarrow 109$) on all inliers, reinforce rank-2, unit-norm \mathbf{F}_k^* after each LM iteration using SVD

We are not yet done

- if there are no wrong correspondences (mismatches, outliers), this gives a local optimum given the 7-point initial estimate
- the algorithm breaks under contamination of (inlier) correspondences by outliers
- the full problem involves finding the inliers!
- in addition, we need a mechanism for jumping out of local minima (and exploring the space of all fundamental matrices)

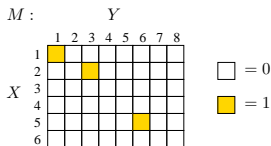
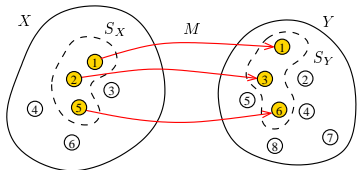
► The Full Problem of Matching and Fundamental Matrix Estimation

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

1. inliers $S_X \subseteq X$, $S_Y \subseteq Y$
2. one-to-one perfect matching $M: S_X \rightarrow S_Y$
3. fundamental matrix \mathbf{F} such that $\text{rank } \mathbf{F} = 2$
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 geometric error $E = \sum_{ij} e_{ij}^2(\mathbf{F})$ is small
5. inlier-outlier and outlier-outlier matches are improbable

perfect matching: 1-factor of the bipartite graph

note a slight change in notation: e_{ij}



$$(M^*, \mathbf{F}^*) = \arg \max_{M, \mathbf{F}} p(E, D, \mathbf{F} \mid M) P(M) \quad (23)$$

- probabilistic model: an efficient language for problem formulation it also unifies 4.a and 4.b
- the (23) 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 $(M^*, \mathbf{F}^*) = \arg \max_{M, \mathbf{F}} p(E, D, \mathbf{F} | M) P(M)$ solve

$$\mathbf{F}^* = \arg \max_{\mathbf{F}} p(E, D, \mathbf{F}) \quad (24)$$

by marginalization of $p(E, D, \mathbf{F} | M) P(M)$ over M

this changes the problem!

ignoring that M are 1:1 matchings and assuming correspondence-wise independence:

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

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

Marginalization:

$$\begin{aligned} p(E, D, \mathbf{F}) &\approx \sum_{m_{11} \in \{0,1\}} \sum_{m_{12}} \cdots \sum_{m_{mn}} p(E, D, \mathbf{F} | M) P(M) = \\ &= \sum_{m_{11}} \cdots \sum_{m_{mn}} \prod_{i=1}^m \prod_{j=1}^n p_e(e_{ij}, d_{ij}, \mathbf{F} | m_{ij}) P(m_{ij}) = \overset{*}{\dots} \overset{1}{=} \\ &= \prod_{i=1}^m \prod_{j=1}^n \underbrace{\sum_{m_{ij} \in \{0,1\}} p_e(e_{ij}, d_{ij}, \mathbf{F} | m_{ij}) P(m_{ij})}_{\text{we will continue with this term}} \end{aligned}$$

Robust Matching Model (cont'd)

$$\begin{aligned}
 & \sum_{m_{ij} \in \{0,1\}} p_e(e_{ij}, d_{ij}, \mathbf{F} \mid m_{ij}) P(m_{ij}) = \\
 & = \underbrace{p_e(e_{ij}, d_{ij}, \mathbf{F} \mid m_{ij} = 1)}_{p_1(e_{ij}, d_{ij}, \mathbf{F})} \underbrace{P(m_{ij} = 1)}_{1-P_0} + \underbrace{p_e(e_{ij}, d_{ij}, \mathbf{F} \mid m_{ij} = 0)}_{p_0(e_{ij}, d_{ij}, \mathbf{F})} \underbrace{P(m_{ij} = 0)}_{P_0} = \\
 & = (1 - P_0) p_1(e_{ij}, d_{ij}, \mathbf{F}) + P_0 p_0(e_{ij}, d_{ij}, \mathbf{F}) \quad (25)
 \end{aligned}$$

- 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) $(\rightarrow 114$ for a simplification)

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

- the $p_1(e_{ij}, d_{ij}, \mathbf{F})$ is typically an easy-to-design term: assuming independence of geometric 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, eg.

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

- \mathbf{F} is a random variable and σ_1 , σ_d , P_0 are parameters
- the form of $T(\sigma_1)$ depends on 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

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

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

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

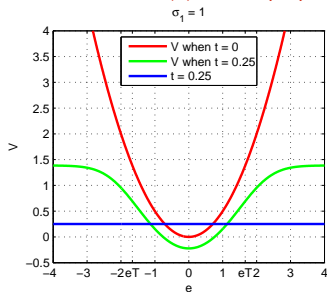
- we define the 'potential function' as: $V(x) = -\log p(x)$, then

$$\begin{aligned} V(E, D \mid \mathbf{F}) &= \sum_{i=1}^m \sum_{j=1}^n \left[\underbrace{-\log \frac{1 - P_0}{T_e(\sigma_1)}}_{\Delta = \text{const}} - \log \left(\underbrace{e^{-\frac{e_{ij}^2(\mathbf{F})}{2\sigma_1^2}} + \frac{P_0}{1 - P_0} \frac{T_e(\sigma_1)}{T_e(\sigma_0)} e^{-\frac{e_{ij}^2(\mathbf{F})}{2\sigma_0^2}}}_{t \approx \text{const}} \right) \right] = \\ &= mn \Delta + \sum_{i=1}^m \sum_{j=1}^n \underbrace{-\log \left(e^{-\frac{e_{ij}^2(\mathbf{F})}{2\sigma_1^2}} + t \right)}_{\hat{V}(e_{ij})} \quad (27) \end{aligned}$$

- note we are summing over all mn matches (m, n are constant!)

► The Action of the Robust Matching Model on Data

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



red – the usual (non-robust) error when $t = 0$
 blue – the rejected correspondence penalty t
 green – ‘robust energy’ (27)

- if the error of a correspondence exceeds a limit, it is ignored
- then $\hat{V}(e) = \text{const}$ and we essentially 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 $t \approx 0$

$$e_T = \sigma_1 \sqrt{-\log t^2} \quad (28)$$

The full optimization problem (24) uses (27):

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

- 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}
- evidence is not needed unless we want to compare different models (eg. homography vs. epipolar geometry)

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

