Robust estimation techniques in computer vision

Vasile Gui

July 2019

Goals of CV: evaluating and recognizing image content

Prior to obtaining semantics from images, we need to extract:

- locations;
- shapes of geometric objects in an image;
- motions in a video sequence;
- or projective transformations between images of the same scene;
- Etc.

What have in common all these applications?

- Presence of noise
- Neighbourhood processing involved

What have in common all these applications?

- Presence of noise
- Neighbourhood processing involved
- The neighbourhood may contain more objects
- Without prior segmentation it is unclear what we measure in the window.

What have in common all these applications?

- Presence of noise
- Neighbourhood processing involved
- The neighbourhood may contain more objects
- Without prior segmentation it is unclear what we measure in the window.
- RE can alleviate this chicken and egg problem.

Some CV applications using RE

Reconstruction: 3D from photo collections

Colosseum, Rome, Italy

San Marco Square, Venice, Italy

Q. Shan, R. Adams, B. Curless, Y. Furukawa, and S. Seitz, The Visual Turing Test for Scene Reconstruction, 3DV 2013

YouTube Video

From Svetlana Lazebnik

Reconstruction: 4D from photo collections

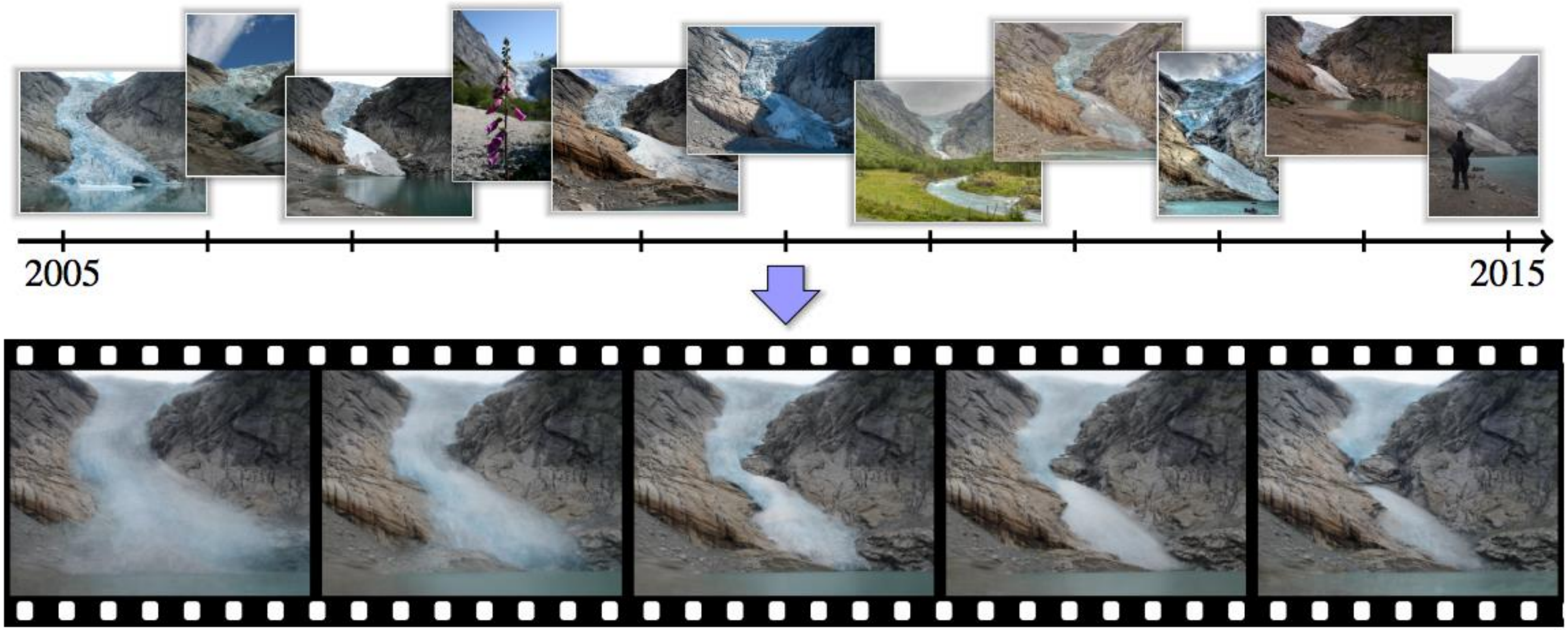


Figure 1: *We mine Internet photo collections to generate time-lapse videos of locations all over the world. Our time-lapses visualize a multitude of changes, like the retreat of the Briksdalsbreen Glacier in Norway shown above. The continuous time-lapse (bottom) is computed from hundreds of Internet photos (samples on top).* Photo credits: Aliento Más Allá, jirihnidek, mcxurxo, elka_cz, Juan Jesús Orío, Klaus Wißkirchen, Daikrieg, Free the image, dration and Nadav Tobias.

R. Martin-Brualla, D. Gallup, and S. Seitz, Time-Lapse Mining from Internet Photos, SIGGRAPH 2015

YouTube Video

From Svetlana Lazebnik

Outline

- Introducing RE from an image filtering perspective
- M estimators
- Maximum likelihood estimators (MLE)
- Kernel density estimators (KDE)
- The RANSAC family
- Some examples and conclusions
- Not a survey of RE in CV
- Raising awareness about RE

Robust estimation

A detail preserving image smoothing perspective

Image smoothing filter goal:

Generate a smoothed image from a noisy image

Image smoothing filter goal:

Generate a smoothed image from a noisy image

Usual assumptions:

– Noise is changing randomly - unorganized

– Useful image part: piecewise smooth

Smoothing filter approach:

For each pixel:

– Define a neighbourhood (window)

– Estimate central pixel's "true" value using all pixels in the window

– Assumption: the estimate should be "similar" to pixels in the window

– Filters differ in similarity definition

What is the problem?

- The processing window may contain more objects or distinctive parts of an object.

What is the problem?

- The processing window may contain more objects or distinctive parts of an object.
- This violates the assumption of similarity with central pixel.

What is the problem?

- The processing window may contain more objects or distinctive parts of an object.
- This violates the assumption of similarity with central pixel.
- If we average pixels, we reduce the effect of random noise…

What is the problem?

- The processing window may contain more objects or distinctive parts of an object.
- This violates the assumption of similarity with central pixel.
- If we average pixels, we reduce the effect of random noise…
- but we blur the image and lose some meaningful details.

Some filter comparisons

Original noisy

mean 5x5

binomial 5x5

median 5x5

Why did the median filter a better job?

- Preserving edges
- Cleaning “salt and pepper” noise

Why did the median filter a better job?

- Preserving edges
- Cleaning “salt and pepper” noise

- Robust estimation perspective of the question

M estimator for filter design

Huber, P. J. (2009). Robust Statistics. John Wiley & Sons Inc.

Pixels: color vectors in a window: $\mathbf{f}_i$

Estimated color: $\hat{\mathbf{f}}$

Residuals: $r_i = \|\mathbf{f}_i - \hat{\mathbf{f}}\|$

Loss function: $\rho(u)$

Minimize loss: $\hat{\mathbf{f}} = \mathrm{argmin} \sum_{i \epsilon W} \rho(r_i)$

M estimator for filter design

Huber, P. J. (2009). Robust Statistics. John Wiley & Sons Inc.

Pixels: color vectors in a window: $\mathbf{f}_i$

Estimated color: $\hat{\mathbf{f}}$

Residuals: $r_i = \|\mathbf{f}_i - \hat{\mathbf{f}}\|$

Loss function: $\rho(u)$

Minimize loss: $\hat{\mathbf{f}} = \operatorname{argmin} \sum_{i \epsilon W} \rho(r_i)$

Least squares (LS) loss: $\rho(u) = u^2$

M estimator for filter design

Huber, P. J. (2009). Robust Statistics. John Wiley & Sons Inc.

Pixels: color vectors in a window: $\mathbf{f}_i$

Estimated color: $\hat{\mathbf{f}}$

Residuals: $r_i = \|\mathbf{f}_i - \hat{\mathbf{f}}\|$

Loss function: $\rho(u)$

Minimize loss: $\hat{\mathbf{f}} = \operatorname{argmin} \sum_{i \epsilon W} \rho(r_i)$

Least squares (LS) loss: $\rho(u) = u^2$

Solution: $\hat{\mathbf{f}} = \sum_{i \epsilon W} \mathbf{f}_i / \sum_{i \epsilon W} 1$ i.e. the **mean**

M estimator for filter design

Weighted LS: $\rho(u_i) = w_i(u_i)^2$

Solution: $\hat{\mathbf{f}} = \sum_{i \epsilon W} w_i \mathbf{f}_i / \sum_{i \epsilon W} \mathrm{w}_i$

i.e. the **weighted mean**

M estimator for filter design

Weighted LS: $\rho(u_i) = w_i(u_i)^2$

Solution: $\hat{\mathbf{f}} = \sum_{i \epsilon W} w_i \mathbf{f}_i / \sum_{i \epsilon W} \mathrm{w}_i$

i.e. the **weighted mean**

- Can be any convolution filter, including binomial, if weights depend on distance to window center.

M estimator for filter design

Weighted LS: $\rho(u_i) = w_i(u_i)^2$

Solution: $\hat{\mathbf{f}} = \sum_{i \epsilon W} w_i \mathbf{f}_i / \sum_{i \epsilon W} \mathrm{w}_i$

i.e. the **weighted mean**

- Can be any convolution filter, including binomial, if weights depend on distance to window center.
- Weights for the *bilateral* filter depend on distance in space-value domain from central pixel.

M estimator for filter design

Absolute value loss: $\rho(u) = |u|$

Suppose gray value images, so the loss function has derivative – sign(u).

M estimator for filter design

Absolute value loss: $\rho(u) = |u|$

Suppose gray value images, so the loss function has derivative – sign(u).

Solution: $\sum_{i \epsilon W} \mathrm{I}(\hat{\mathrm{f}} > \mathrm{f}_i) = \sum_{i \epsilon W} \mathrm{I}(\hat{\mathrm{f}} < \mathrm{f}_i)$,

Equal number of lower and higher values than the estimate,

i.e. the **median**: middle of the ordered set.

Why did the median filter outperform convolution filters?

Why did the median filter outperform convolution filters?

Outlier samples in the filtering window have less influence on the median than on the weighted mean.

Why did the median filter outperform convolution filters?

Outlier samples in the filtering window have less influence on the median than on the weighted mean.

Influence function (IF) of a linear filter:

$$\psi(u) = \frac{d\rho(u)}{du}$$

$$\rho(u) = w \times u^2$$

$$\psi(u) = 2w \times u$$

Why did the median filter outperform convolution filters?

Outlier samples in the filtering window have less influence on the median than on the weighted mean.

Influence function (IF) of a linear filter:

$$\psi(u) = \frac{d\rho(u)}{du}$$

$$\rho(u) = w \times u^2$$

$$\psi(u) = 2w \times u$$

Any sample can have unbounded effect on the estimate (not robust!)

Why did the median filter outperform convolution filters?

Outlier samples in the filtering window have less influence on the median than on the weighted mean.

Influence function (IF) of a linear filter:

$$\psi(u) = \frac{d\rho(u)}{du}$$

$$\rho(u) = w \times u^2$$

$$\psi(u) = 2w \times u$$

Any sample can have unbounded effect on the estimate (not robust!)

Higher residual sample - higher influence (!!!)

Loss function and IF of the median filter

$$\rho(u) = |u|$$

$$\psi(u) = sign(u) = \begin{cases} 1, & u > 0 \\ 0, & u = 0 \\ -1, & u < 0 \end{cases}$$

Bounded (and equal) influence of all samples.

Loss function and IF of the median filter

$$\rho(u) = |u|$$

$$\psi(u) = sign(u) = \begin{cases} 1, & u > 0 \\ 0, & u = 0 \\ -1, & u < 0 \end{cases}$$

Bounded (and equal) influence of all samples.

Brake down point (BP): number of points arbitrarily deviated causing arbitrarily big estimation error.

Median BP: 50%.

Loss function and IF of the median filter

$$\rho(u) = |u|$$

$$\psi(u) = sign(u) = \begin{cases} 1, & u > 0 \\ 0, & u = 0 \\ -1, & u < 0 \end{cases}$$

Bounded (and equal) influence of all samples.

Brake down point (BP): number of points arbitrarily deviated causing arbitrarily big estimation error.

Median BP: 50%.

Linear filters: 0%: one very bad outlier is enough ☹...

Note, the vector median is a different story.

Should we always use the sample median?

Should we always use the sample median?

- When data do not contain outliers the mean has better performance.

Should we always use the sample median?

- When data do not contain outliers the mean has better performance.
- We want estimators combining the low variance of the mean at normal distributions with the robustness of the median under contamination.

Should we always use the sample median?

- When data do not contain outliers the mean has better performance.
- We want estimators combining the low variance of the mean at normal distributions with the robustness of the median under contamination.
- Let us compare the two filters also from the maximum likelihood perspective!

ML estimation of location, μ

- Suppose we know samples are independent and identically distributed (i.i.d.) with probability density function (pdf) $p()$.
- The likelihood to observe the data samples is

$$p(x_1, x_2, \ldots x_n | \mu) = \prod_{i=1}^{n} p(x_i - \mu)$$

- The maximum likelihood estimate (MLE) of μ is

$$\hat{\mu} = \arg\max p(x_1, x_2, \ldots x_n | \mu)$$

- MLE for Gaussian data in the sample mean
- MLE for Laplacian data (larger tails) is median

M estimation of location, μ

- The maximum likelihood estimate (MLE) solves

$$\hat{\mu} = \arg\min \sum_{i=1}^{n} \rho(x_i - \mu)\,, \rho() = -\log p()$$

How can we design robust loss functions?

How can we design robust loss functions?

- We need a way to cope with the presence of outlier data, while keeping the efficiency of a classical estimator for normal data.

How can we design robust loss functions?

- We need a way to cope with the presence of outlier data, while keeping the efficiency of a classical estimator for normal data.

Two types of approaches:

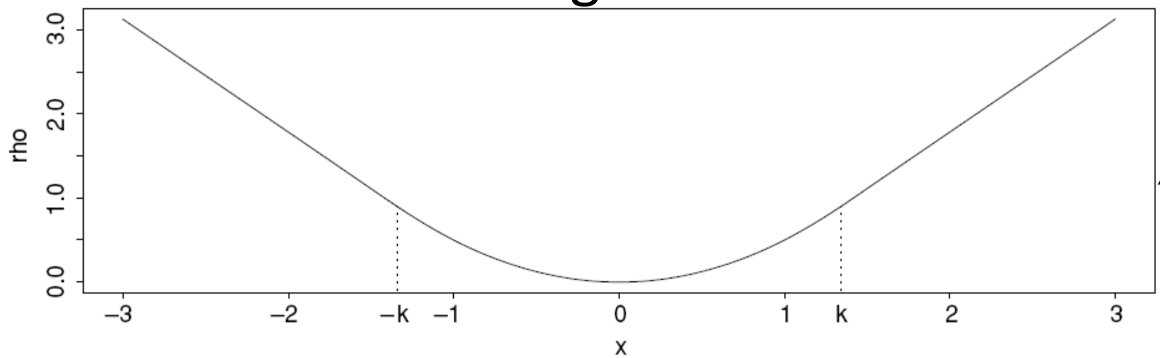
1. Shaping the $\rho(u)$ function
2. Analysis of the set of residuals

1.Shaping the $\rho(u)$ function

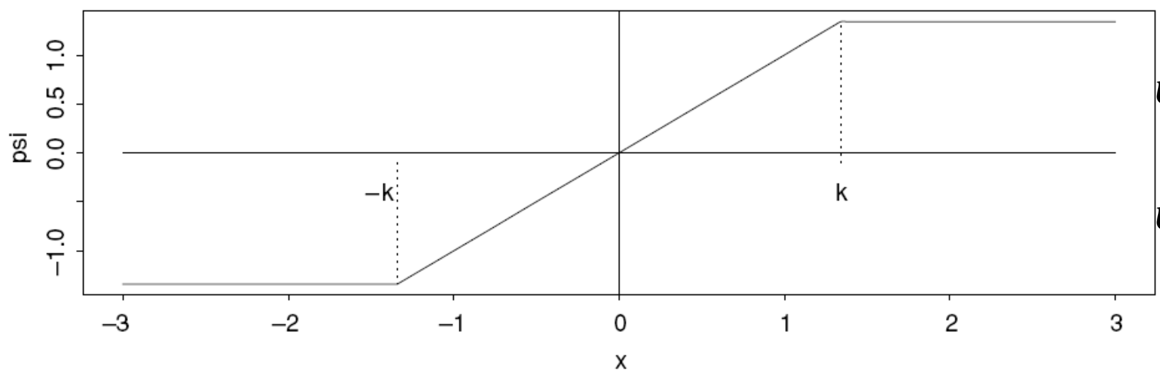
We want to give less influence to points with residuals beyond "some value".

Huber ρ and ψ functions

Quadratic loss for small residuals
Linear loss for large residuals



$$\rho_k(x) = \begin{cases} x^2 & if\ |x| \le k \\ 2k|x| - k^2 & if\ |x| > k \end{cases}$$



$$\psi_k(x) = \begin{cases} x & if\ |x| \le k, k \ne 0 \\ sign(x)k & if\ |x| > k, k \ne 0 \end{cases}$$

$$\psi_0(x) = sign(x)$$

Ricardo A. Maronna, R. Douglas Martin and V´ıctor J. Yohai, Robust Statistics: Theory and Methods , 2006 John Wiley & Sons

type	$\rho(x)$	$\psi(x)$	$w(x)$
L_2	$x^2/2$	x	1
L_1	$\lvert x\rvert$	$\text{sgn}(x)$	$\frac{1}{\lvert x\rvert}$
$L_1 - L_2$	$2(\sqrt{1+x^2/2}-1)$	$\frac{x}{\sqrt{1+x^2/2}}$	$\frac{1}{\sqrt{1+x^2/2}}$
L_p	$\frac{\lvert x\rvert^\nu}{\nu}$	$\text{sgn}(x)\lvert x\rvert^{\nu-1}$	$\lvert x\rvert^{\nu-2}$
"Fair"	$c^2[\frac{\lvert x\rvert}{c} - \log(1+\frac{\lvert x\rvert}{c})]$	$\frac{x}{1+\lvert x\rvert/c}$	$\frac{1}{1+\lvert x\rvert/c}$
Huber $\begin{cases}\text{if } \lvert x\rvert \leq k \\ \text{if } \lvert x\rvert \geq k\end{cases}$	$\begin{cases}x^2/2 \\ k(\lvert x\rvert - k/2)\end{cases}$	$\begin{cases}x \\ k\,\text{sgn}(x)\end{cases}$	$\begin{cases}1 \\ k/\lvert x\rvert\end{cases}$
Cauchy	$\frac{c^2}{2}\log(1+(x/c)^2)$	$\frac{x}{1+(x/c)^2}$	$\frac{1}{1+(x/c)^2}$
Geman-McClure	$\frac{x^2/2}{1+x^2}$	$\frac{x}{(1+x^2)^2}$	$\frac{1}{(1+x^2)^2}$
Welsch	$\frac{c^2}{2}[1-\exp(-(x/c)^2)]$	$x\exp(-(x/c)^2)$	$\exp(-(x/c)^2))$
Tukey $\begin{cases}\text{if } \lvert x\rvert \leq c \\ \text{if } \lvert x\rvert > c\end{cases}$	$\begin{cases}\frac{c^2}{6}\left(1-[1-(x/c)^2]^3\right) \\ (c^2/6)\end{cases}$	$\begin{cases}x[1-(x/c)^2]^2 \\ 0\end{cases}$	$\begin{cases}[1-(x/c)^2]^2 \\ 0\end{cases}$

Lots of robust loss functions have been studied.

How to choose best parameters for a loss function?

2. Analysis of the set of residuals

Order statistics approaches

2. Analysis of the set of residuals

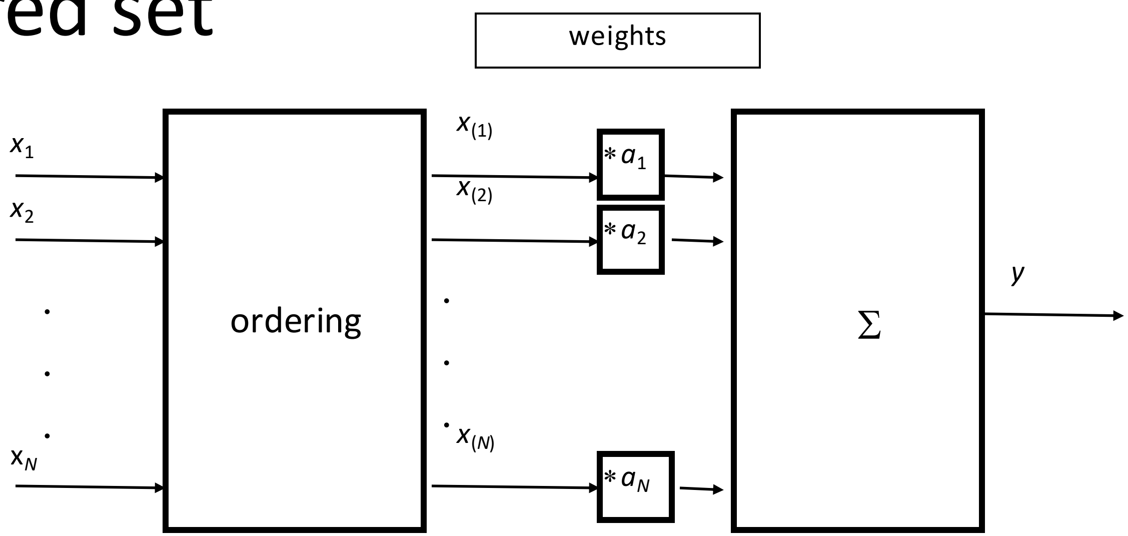
Order statistics approaches

- **L estimators**: linear combination on ordered statistics set $x_{(i)}$.
- Samples weighted according to position in ordered set

2. Analysis of the set of residuals

Order statistics approaches

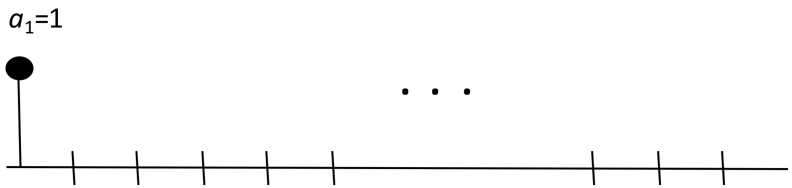
- **L estimators**: linear combination on ordered statistics set $x_{(i)}$.
- Samples weighted according to position in ordered set



$$\sum_{k=1}^{N} a_k = 1$$

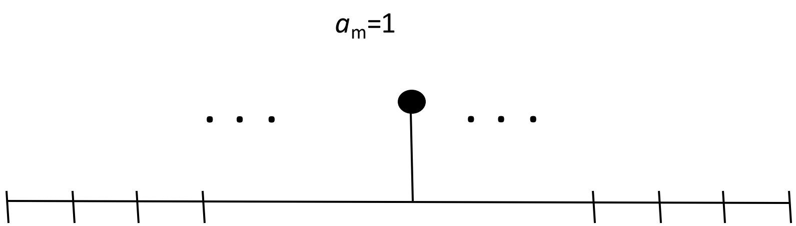
$$y = a_1 x_{(1)} + a_2 x_{(2)} + \ldots + a_N x_{(N)}$$

L estimator examples.

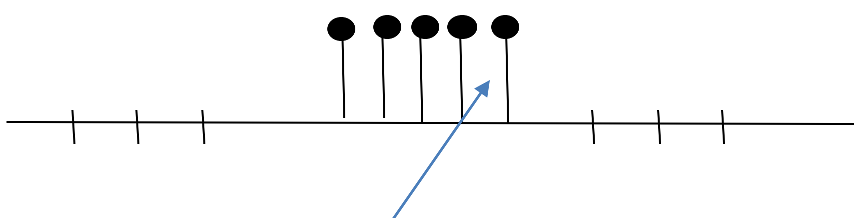


Min = **morphological erosion**

Max = **morphological dilation**



Median



Trimmed mean

$$a_i = I(m < i \leq n - m)/(n - 2m)$$

The trimmed mean

- Let $\beta \in [0,0.5)$ and $m = int[\beta(N-1)]$

- The β-trimmed mean is defined by

$$\bar{x}_\beta = \frac{1}{N-2m} \sum_{i=m+1}^{N-m} x_{(i)}$$

- $\overline{x}_\beta$ is the sample mean after the m largest and the m smallest samples have been discarded
- Half percentage of discarded samples: β

The trimmed mean – cont.

- Limit cases
  - $\beta=0$ → the sample mean
  - $\beta \to 0.5$ → the sample median

The trimmed mean – cont.

- Limit cases
  - β=0 → the sample mean
  - β →0.5 → the sample median

– **Adaptive trimmed mean**:

- Variance ↑ ⇨ β ↑

The trimmed mean – cont.

- Limit cases
  - β=0 → the sample mean
  - β →0.5 → the sample median
- **Adaptive trimmed mean**:
  - Variance ↑ ⇨ β ↑
- RE of variance: median of absolute deviations:

$$MAD = \text{median}(r_i)$$

- BP of β % trimmed mean = β %

M estimation and the mean shift filter

Weighted LS loss with weights depending on closeness to estimate (not just position in the window):

$$w(\mathbf{y}) = w(\hat{\mathbf{x}} - \mathbf{y})$$

Pixel: $\mathbf{x} = \begin{bmatrix} \mathbf{x}_s \\ \mathbf{f}(\mathrm{x}_s) \end{bmatrix}$

$$\hat{\mathbf{x}} = \frac{\sum_{\mathbf{y}} w(\hat{\mathbf{x}} - \mathbf{y})\mathbf{y}}{\sum_{\mathbf{y}} w(\hat{\mathbf{x}} - \mathbf{y})}$$

M estimation and the mean shift filter

Weighted LS loss with weights depending on closeness to estimate (not just position in the window):

$$w(\mathbf{y}) = w(\hat{\mathbf{x}} - \mathbf{y})$$

Pixel: $\mathbf{x} = \begin{bmatrix} \mathbf{x}_s \\ \mathbf{f}(\mathrm{x}_s) \end{bmatrix}$

$$\hat{\mathbf{x}} = \frac{\sum_{\mathbf{y}} w(\hat{\mathbf{x}} - \mathbf{y})\mathbf{y}}{\sum_{\mathbf{y}} w(\hat{\mathbf{x}} - \mathbf{y})}$$

Needs iterations to be solved because the weights depend on the unknown estimate!

Mean shift iterations are similar with M estimators

Minimize loss <–> maximize probability density

Mean shift iterations are similar with M estimators

Minimize loss <–> maximize probability density

Algorithm: gradient ascent to find maxima of the kernel probability density estimate (KDE)

Mean shift iterations are similar with M estimators

Minimize loss <–> maximize probability density

Algorithm: gradient ascent to find maxima of the kernel probability density estimate (KDE)

Fukunaga 1975, Comaniciu & Meer 2002

Mean shift used for filtering, segmentation, tracking…

Mean shift iterations are similar with M estimators

Minimize loss <–> maximize probability density

Algorithm: gradient ascent to find maxima of the kernel probability density estimate (KDE)

Fukunaga 1975, Comaniciu & Meer 2002

Mean shift used for filtering, segmentation, tracking…

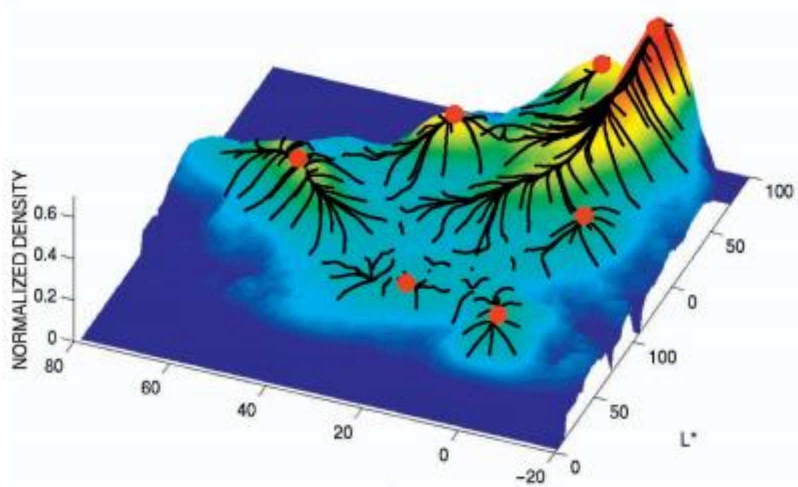
$$w(\hat{\mathbf{x}} - \boldsymbol{y}) = g(||\hat{\mathbf{x}} - \boldsymbol{y}||/h)$$

g: derivative of the density interpolation kernel
h: scale (degree of smoothing)
Radially symmetric distance metric here

Mean shift iterations are similar with M estimators

Original

$(h_s, h_r) = (32, 8)$



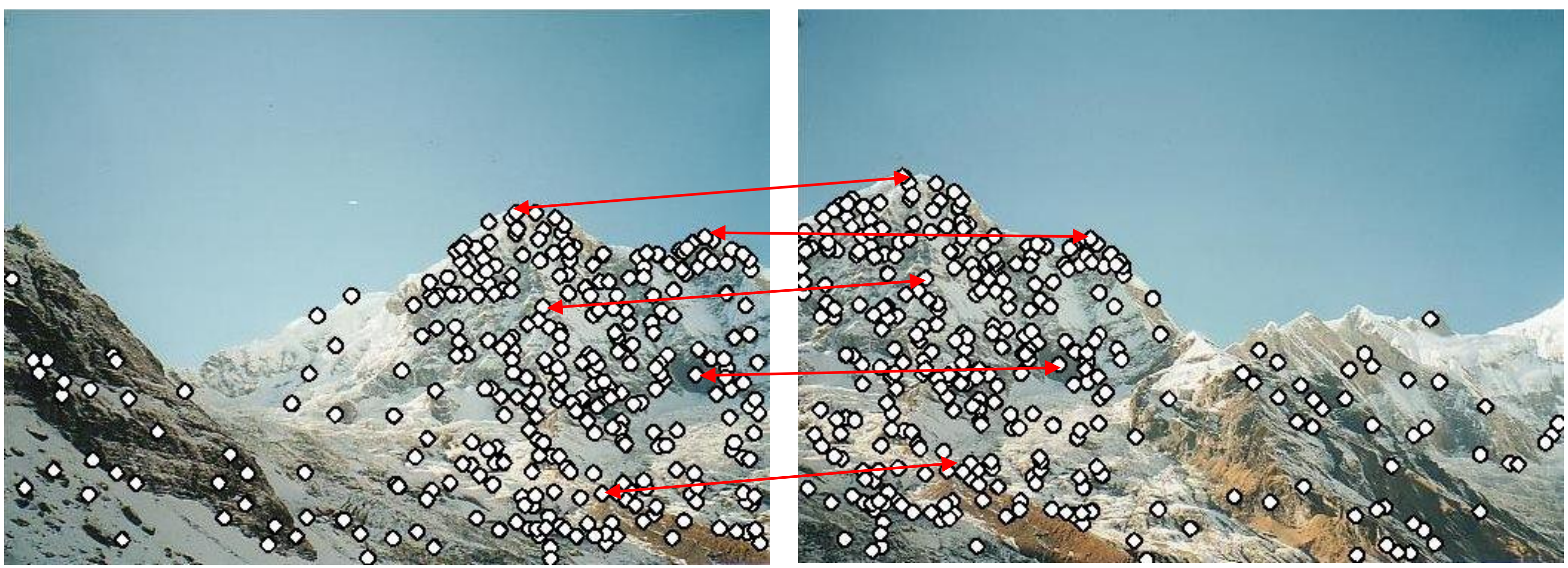
(c)

What if we have more than just 50% outliers?

This situation occurs often in key point based image registration.

How do we build a panorama?

- Detect feature points in both images
- Find corresponding pairs



Matching with Features

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images

Can we beat the 50% BP of the median?

Can we beat the 50% BP of the median?

- If outliers do not conspire, it should be possible ☺

Can we beat the 50% BP of the median?

- If outliers do not conspire, it should be possible ☺
- Probability density mode definition does not imply majority!

Can we beat the 50% BP of the median?

- If outliers do not conspire, it should be possible ☺
- Probability density mode definition does not imply majority!
- Needing a good *search strategy* (and good *parameter setting*). Better than an (inspired) initial guess.

Can we beat the 50% BP of the median?

- If outliers do not conspire, it should be possible ☺
- Probability density mode definition does not imply majority!
- Needing a good *search strategy* (and good *parameter setting*). Better than an (inspired) initial guess.
- Random sample consensus (RANSAC) approach.
- Especially designed in the CVIP community.

Along with Hough, MINPRAN etc.

M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

RANSAC is also a voting approach like mode detection.

Using a sampling strategy for optimization.

RANSAC is also a voting approach like mode detection.

Using a sampling strategy for optimization.

Randomly select a minimum size subset of points to generate a solution:

-Generate many potential solutions.

-Select the solution with the best consensus.

-Best consensus means maximum inliers (within defined limits), i.e. maximum density in the solution space.

RANSAC is also a voting approach like mode detection.

Using a sampling strategy for optimization.

Randomly select a minimum size subset of points to generate a solution:

-Generate many potential solutions.

-Select the solution with the best consensus.

-Best consensus means maximum inliers (within defined limits), i.e. maximum density in the solution space.

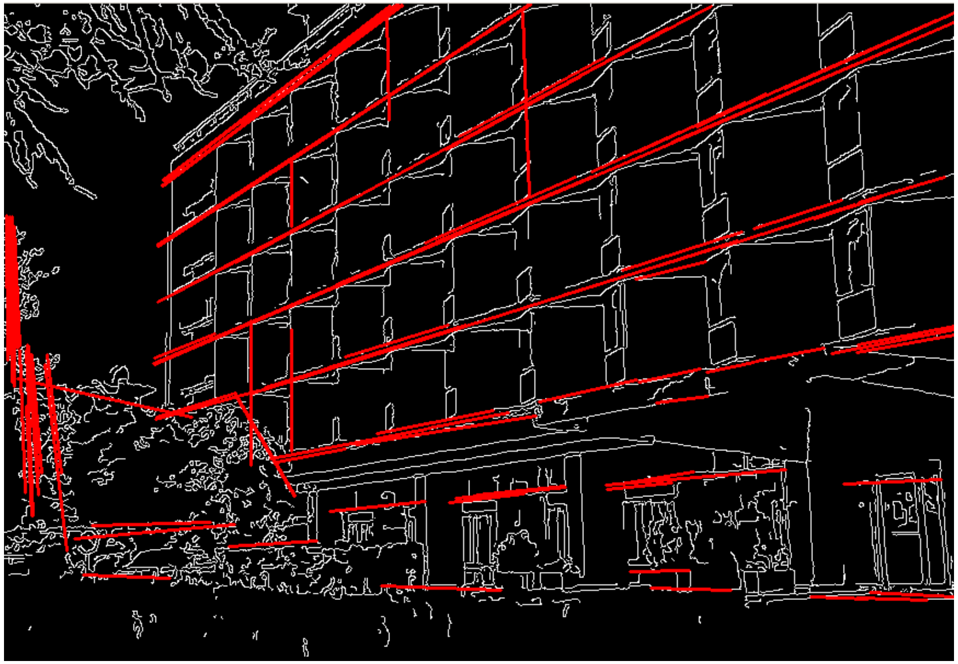
Developments: MLESAC, NAPSAC, PROSAC…

We used KDE with RANSAC (softer thresholds)

RANSAC

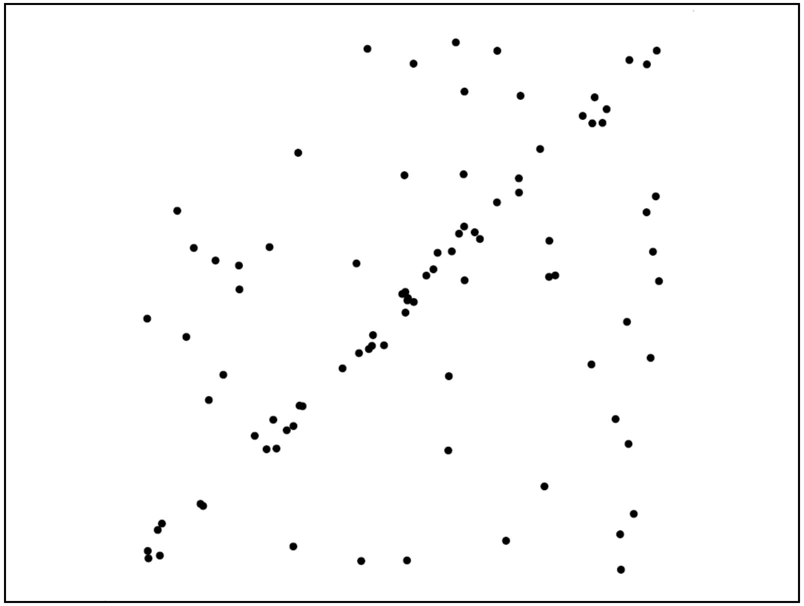
- Repeat ***N*** times:
- Draw ***s*** points uniformly at random
- Fit line to these ***s*** points
- Find *inliers* to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- If there are ***d*** or more inliers, accept the line and refit using all inliers

Model fitting example



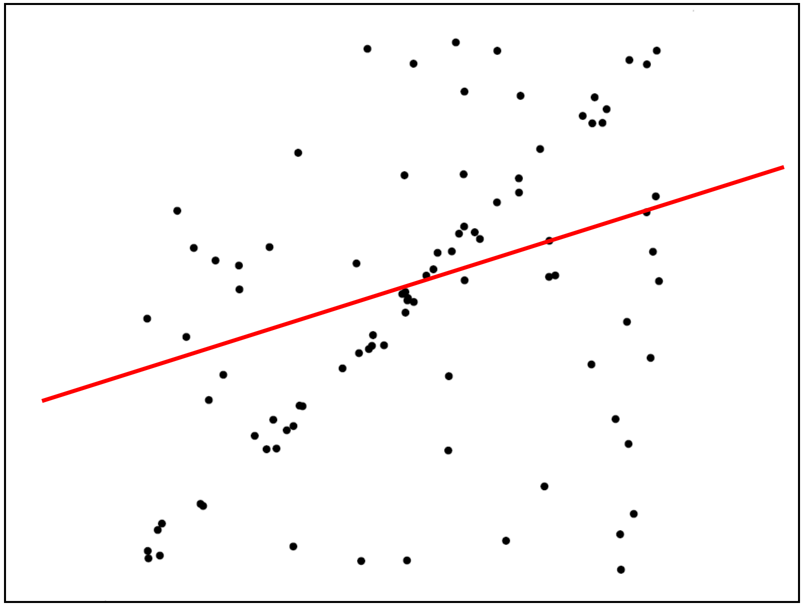
Detect lines using RANSAC…

RANSAC for line fitting example



Source: R. Raguram

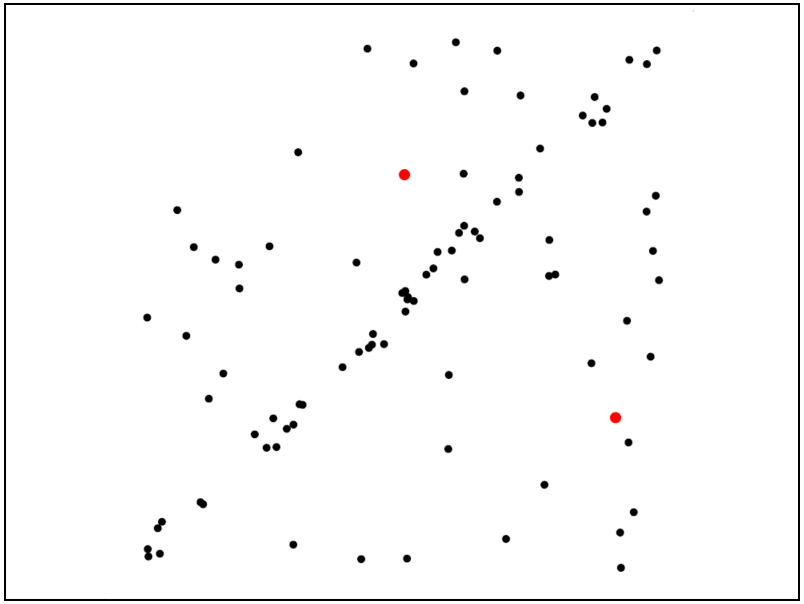
RANSAC for line fitting example



Least-squares fit

Source: R. Raguram

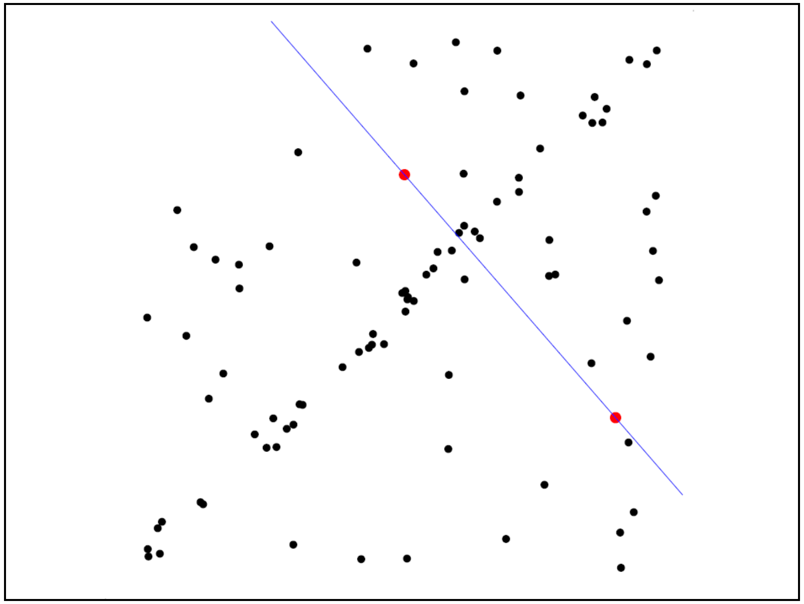
RANSAC for line fitting example



1. Randomly select minimal subset of points

Source: R. Raguram

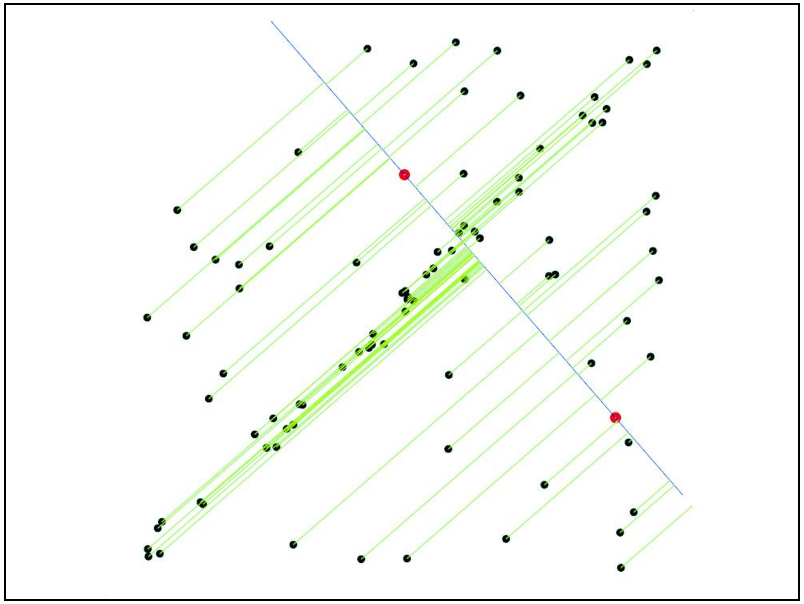
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model

Source: R. Raguram

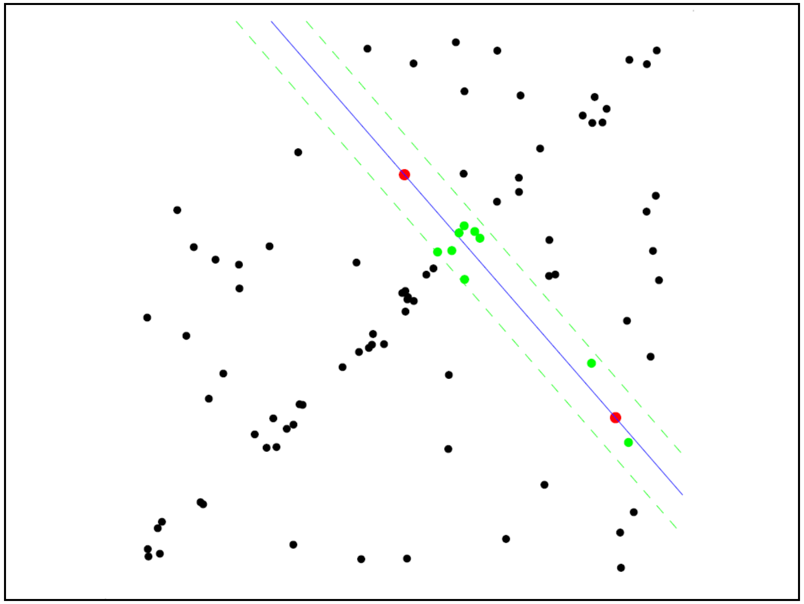
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Source: R. Raguram

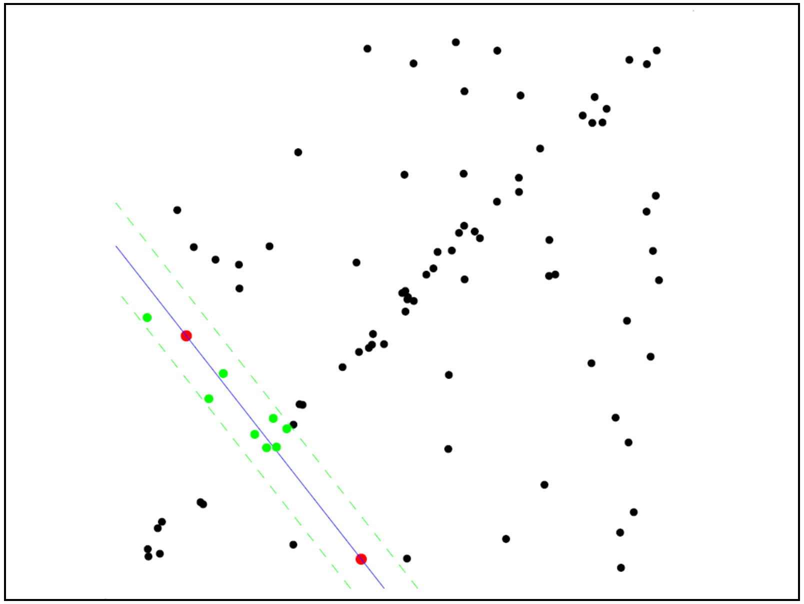
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Source: R. Raguram

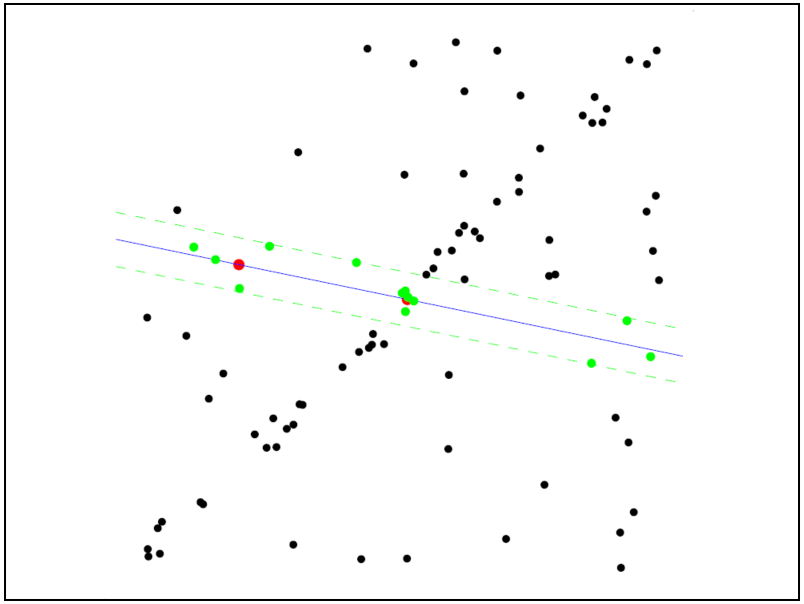
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

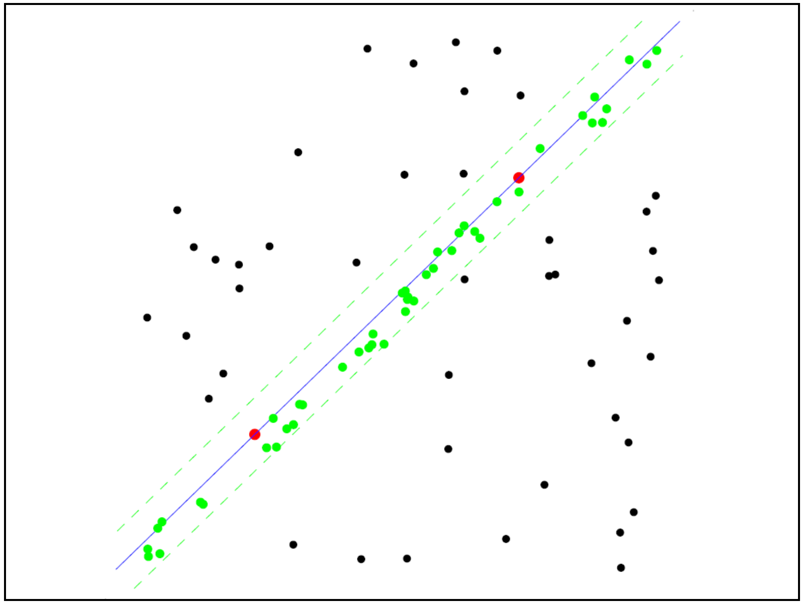


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

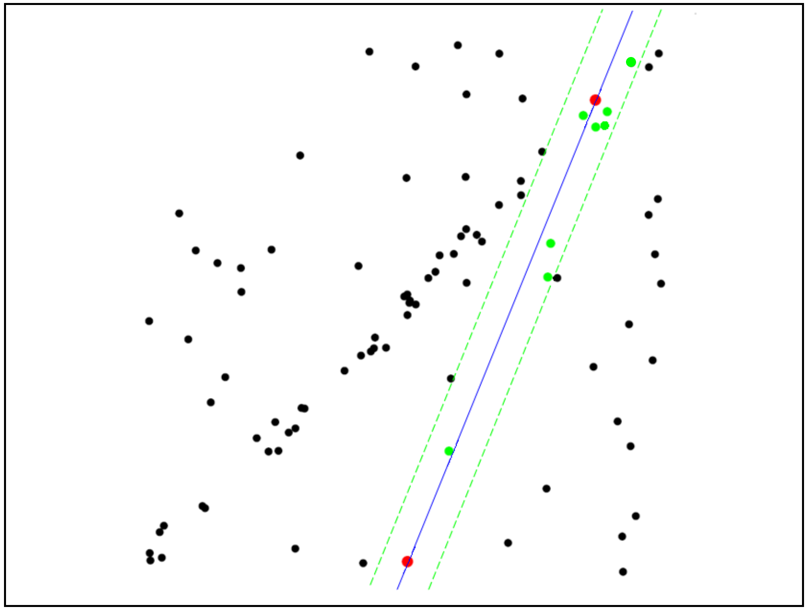
Uncontaminated sample



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

Choosing the parameters

- Initial number of points ***s***
  - Typically minimum number needed to fit the model
- Distance threshold ***t***
  - Choose ***t*** so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples ***N***
  - Choose ***N*** so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)

$$\left(1-(1-e)^s\right)^N = 1-p$$

$$N = \log(1-p)/\log\left(1-(1-e)^s\right)$$

	proportion of outliers e						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Source: M. Pollefeys

Other RE applications

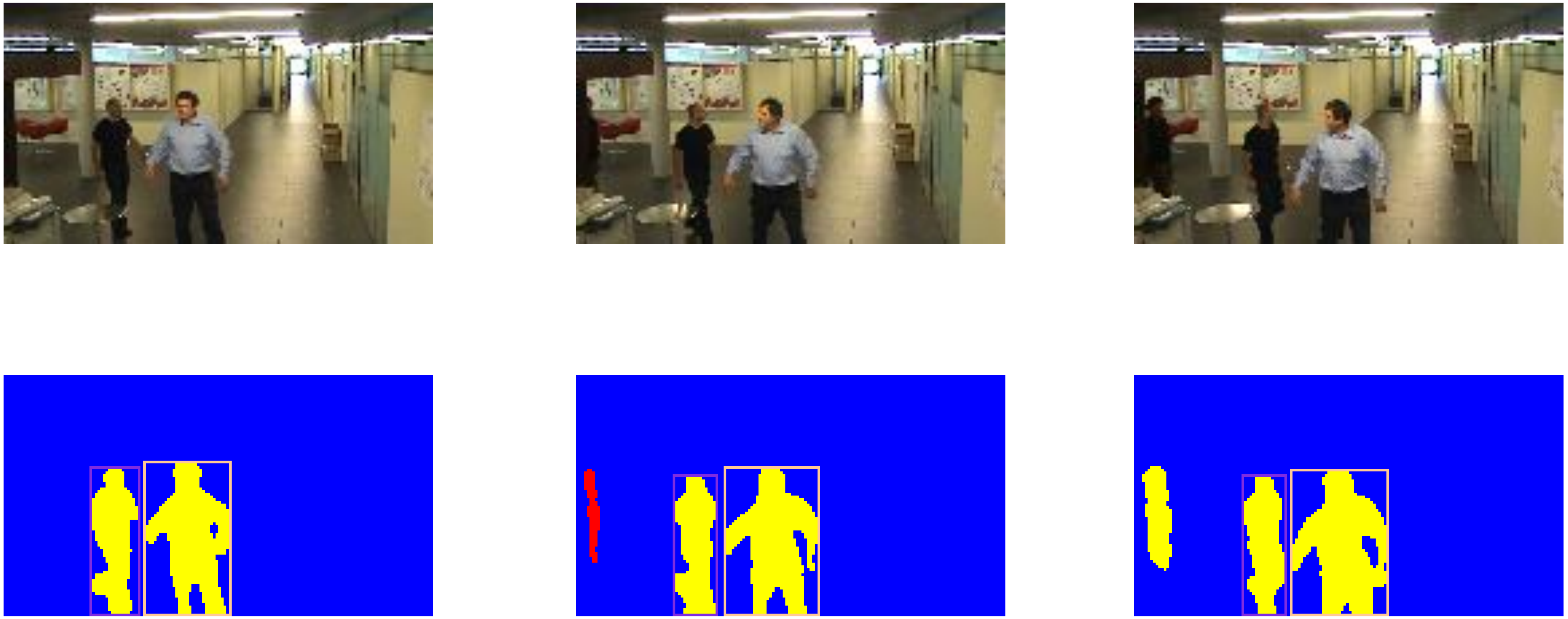
- **Detail preserving image filtering**
- **Background estimation** for video surveillance
- **Finger detection and tracking for human computer interface**
- **Posterior attenuation feature extraction for steatosis rating**

Detail preserving image smoothing

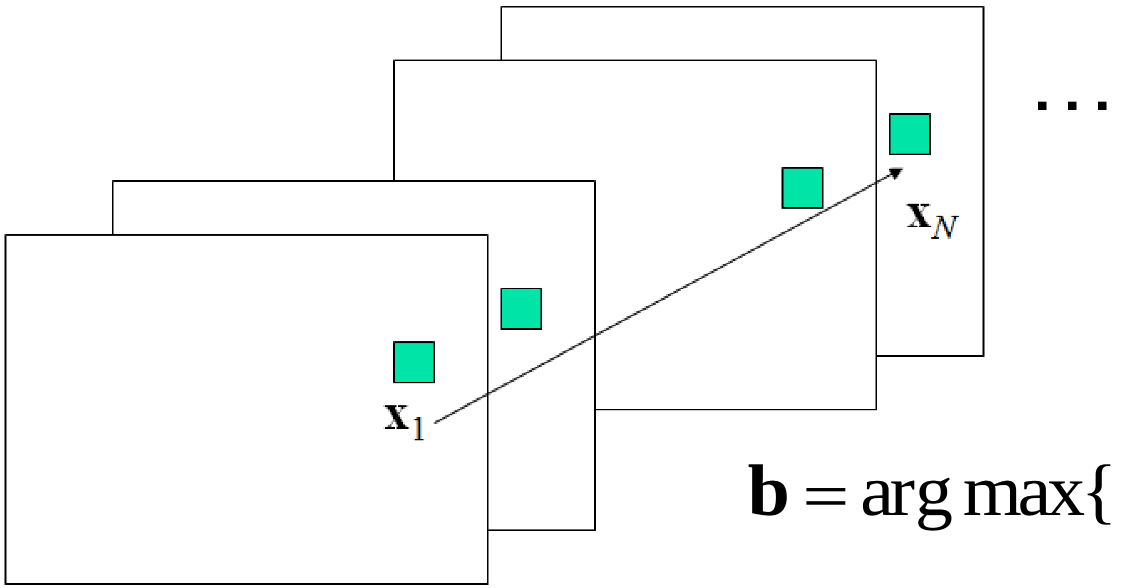
Multiscale mode filter - Improves mean shift filter

Gui - EUSIPCO 2008

Background segmentation in videos



Background segmentation in videos



$$\mathbf{b} = \arg\max\{ p(\mathbf{x}) \}$$

Assumptions:

Static camera

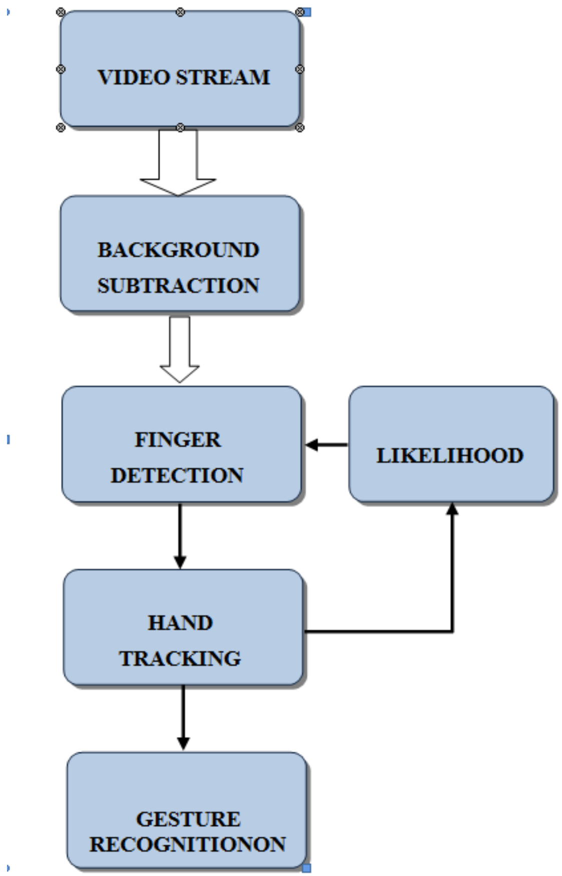
The background is what we see "most frequently" at each location

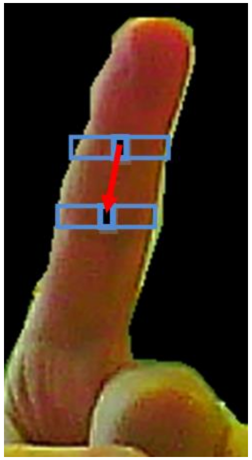
Approaches:

Parametric density estimation: MoG

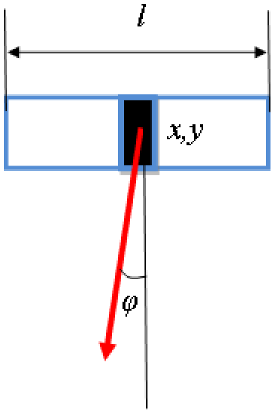
Non-parametric: KDE

HMI based on finger detection and tracking



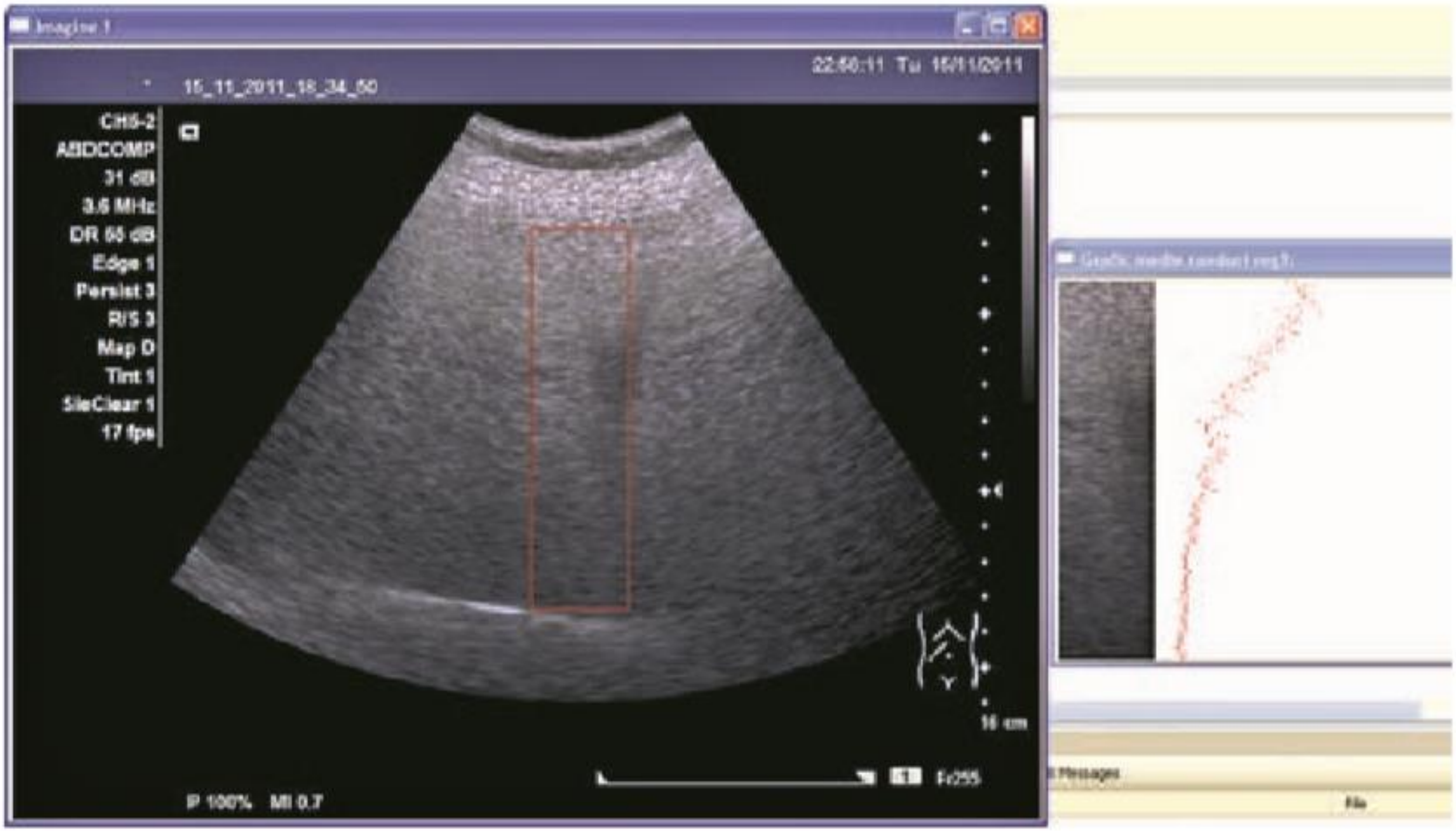


Fingerlet feature



Using histograms of *l* and *φ* to find fingerlet density modes
Fast,1D subspace search

Robust posterior attenuation feature extraction for steatosis rating



Conclusions

- M estimators more general than MLE
- KDE: similar with M but with probabilistic view
- RANSAC and related algorithms: a powerful search method
- **Principles from robust estimation worth to be kept in mind for successfully solving a large variety of CV problems.**

Thank you for your attention!
Questions?