Efficient implementation of entropy estimation for registration

Mobility project report
5.12.2005 - 26.3.2006

Project information

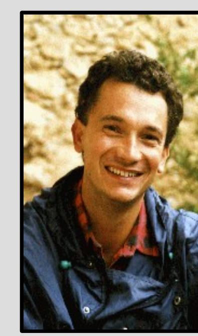
- Creative vision in practice
- Partners:
  - Neovision

NEOVISION
Industrial Vision Systems

  - Center for Machine Perception, Czech Technical University
    - part of the Miracle Center of Excellence

Education and Culture
Leonardo da Vinci

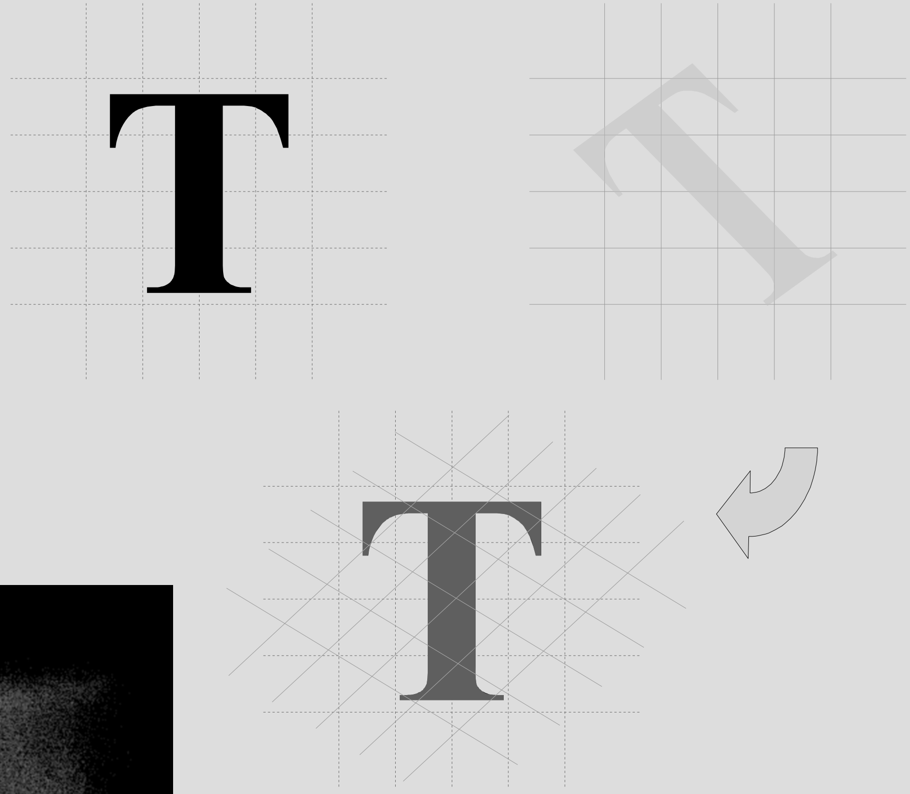
m p

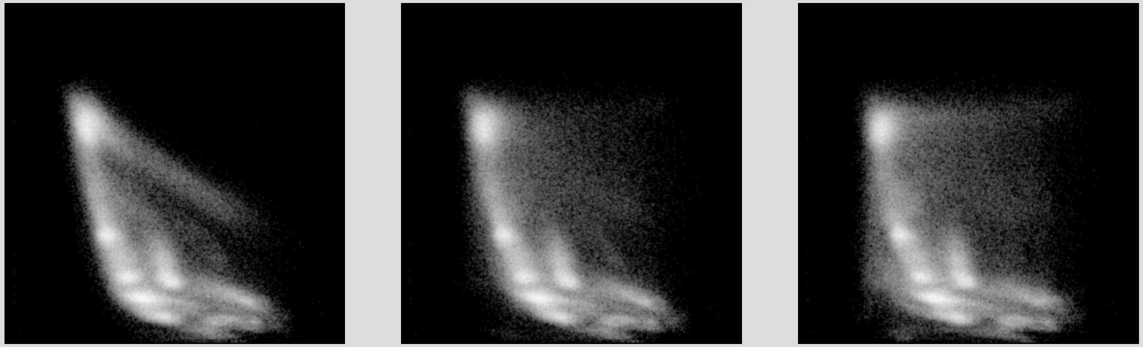


Leonardo Da Vinci mobility, 5.12.2005–26.3.2006
13 weeks

Registration using MI

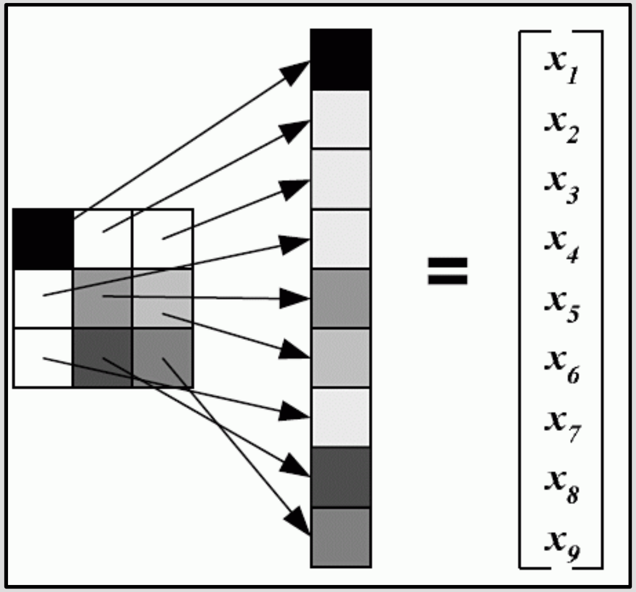
- Spatial alignment of images
- For transformation evaluation
  => similarity measure
  - Geometrical features
  - Corresponding voxel values
    - SSD, correlation
    - Dispersion of joint histogram
      => joint entropy, MI





Problem introduction

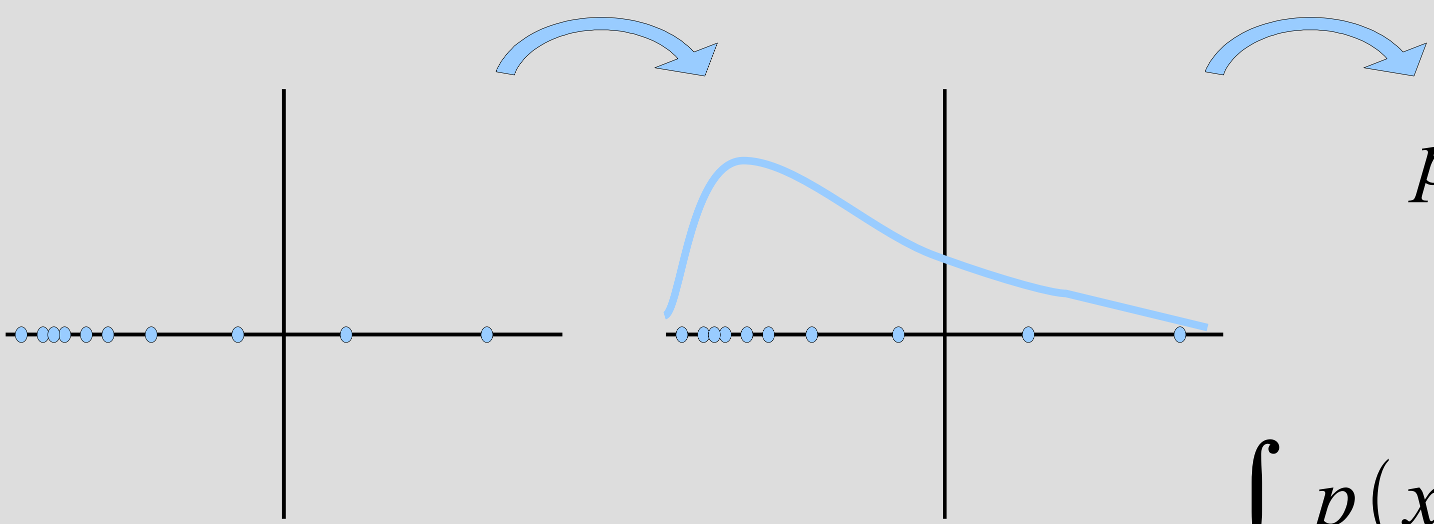
- Mutual information criterion for image registration => estimation from finite sample
  - 10^5 - 10^7 data points => fast, if we want to use the whole sample => not Parzen windows
  - High dimensional features (color, spatial neighborhood) to use more information => reliable for high dimensional distributions => no histogram based estimation



  - Used as an optimization criterion => statistically stable

Estimating entropy

- Plug-in estimates
  - estimate of the distribution, which is pluged into the definition of entropy
    - histogram estimate, mixture of Gaussians, Parzen windows

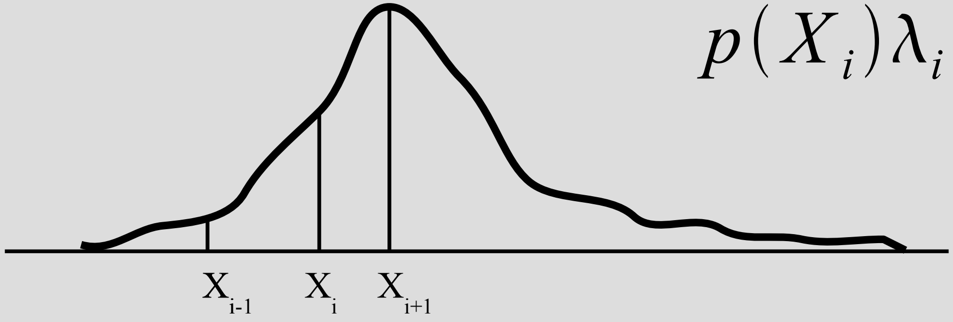


$$p(x)$$

$$\int p(x)\log(p(x))\,dx$$

Estimating entropy

- Order statistics and nearest neighbor estimates
  - estimates from the distances between neighbor values in the sample
    - for 1D order statistics
    - for arbitrary D nearest neighbor
  - In 1D case
    - Expectation of probability mass between two successive points in order statistics is constant



$$p(X_i)\lambda_i \approx \frac{1}{N} \Rightarrow$$

$$\log(p(X_i)) \approx -\log(\lambda_i) - \log(N)$$

Estimating entropy

- Compression estimates
  - entropy is the lower bound on the size of lossless compressed data
    - Lempel-Ziv
    - Burrows-Wheeler
  - compressing the data => size after is the upper bound on entropy
  - the better compression the better estimate

Implemented approach

- modified Kozachenko – Leonenko nearest neighbor estimate
- Based on an unpublished work of Jan Kybic
- kD tree used as underlying data structure for all-NN search
- for the use in optimisation procedures an update operation designed
- for the search a Best Bin First approach used

Properties of KL Estimator

- Kozachenko – Leonenko nearest neighbor estimator [KoLe]

$$KL(\{X_1,\dots,X_N\}) = -\frac{1}{N}\sum_{j=1}^{N}[\log(\lambda_j)+\log[2(N-1)]+\frac{\gamma}{\ln(2)}]$$

- Modified for L_∞ norm and dimensionality d

$$KL(\{X_1,\dots,X_N\}) = -\frac{1}{N}\sum_{j=1}^{N} d\log(\lambda_j)+\log[2^d(N-1)]+\gamma$$

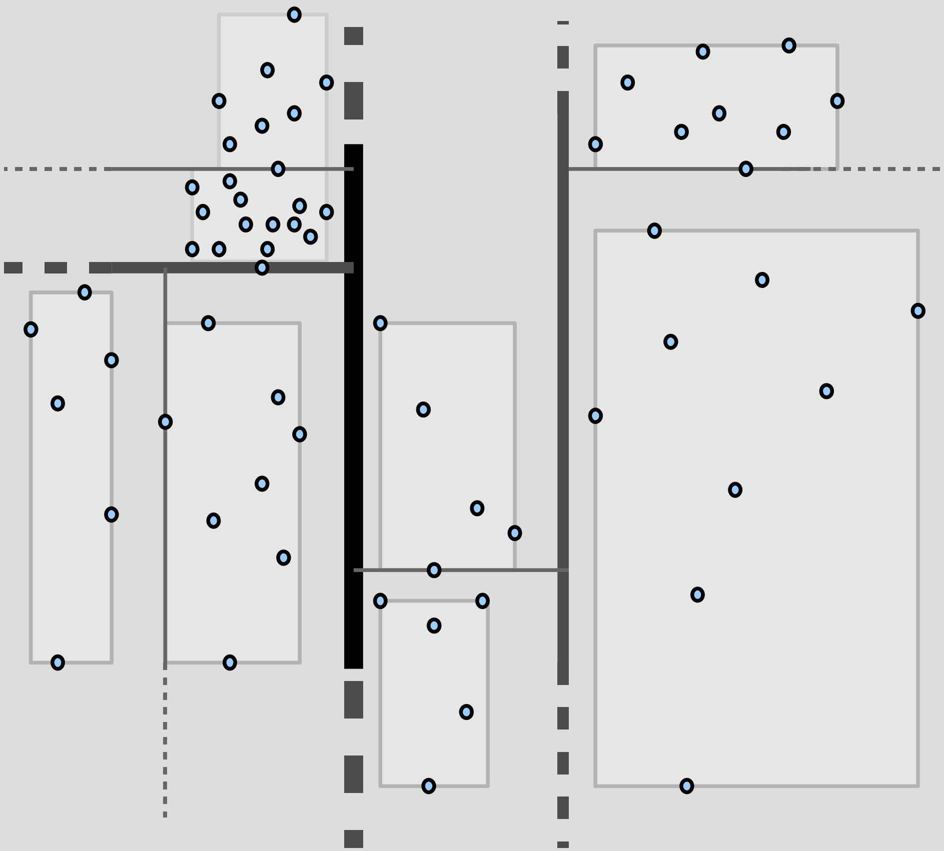
- Based directly on sample values
- Asymptotically unbiased
- Binless => no „curse of dimensionality“
- Mean square consistent => $\lim_{n\to\infty} E[(H_n - H(X))^2] = 0$

Efficient implementation issues

- All nearest neighbor search
  - $O(dN^2)$ for brute force approach
  - More efficient
    - Space partioning trees – kD trees, box-decomposition trees
    - Voronoi diagrams
  - Approximate nearest neighbor
- Quantization
  - Multiple points
  - KL estimator is based on continuous distribution assumption

kD tree

- Binary space decomposition tree
- Root node represents the whole space
- Children => cutting the parent hyperrectangle by a plane
  => loose bounding box
  for each node
- Tight bounding boxes
  maintained
- Axis aligned
  hyperrectangles
  => L_∞ norm used
- Optional parameter -
  number of points
  in leaf



Quantization problem

- KL Estimator derived with continuous impulse-free distribution assumption
- Quantization => equal values in sample =>

$$\log(\lambda_i) = \log(0) = -\infty$$

- Used solution

$$\log(\lambda_i) \;\Rightarrow\; \log\left(\frac{\epsilon^d}{k_i}\right) \quad for\ \lambda_i < \epsilon$$

Implementation details

- In ANSI C++
- Inspired by previous Ocaml implementation, but written from scratch
- Platform independent
- Succesfully compiled on Win32 platform with GCC compiler in MinGW enviroment and MS Visual C++ compiler
- Working enviroment: msys, mingw, Code::Blocks

Experiments – artifical data

- Run on Celeron 2 Ghz, 248 MB RAM
- Entropy estimation for 1 000 000 normally distributed points, for leaf size 40, max visit 1000, for d=1 100

Dimensionality	True entropy	Building time	Iterating time	Entropy error
1	1.41894	5,437	1,563	-0.000538472
3	4.25682	8,937	16,641	-0.00378593
5	7.09469	10,609	119.547	-0.0115131
10	14.1894	13.578	654.015	1.20752

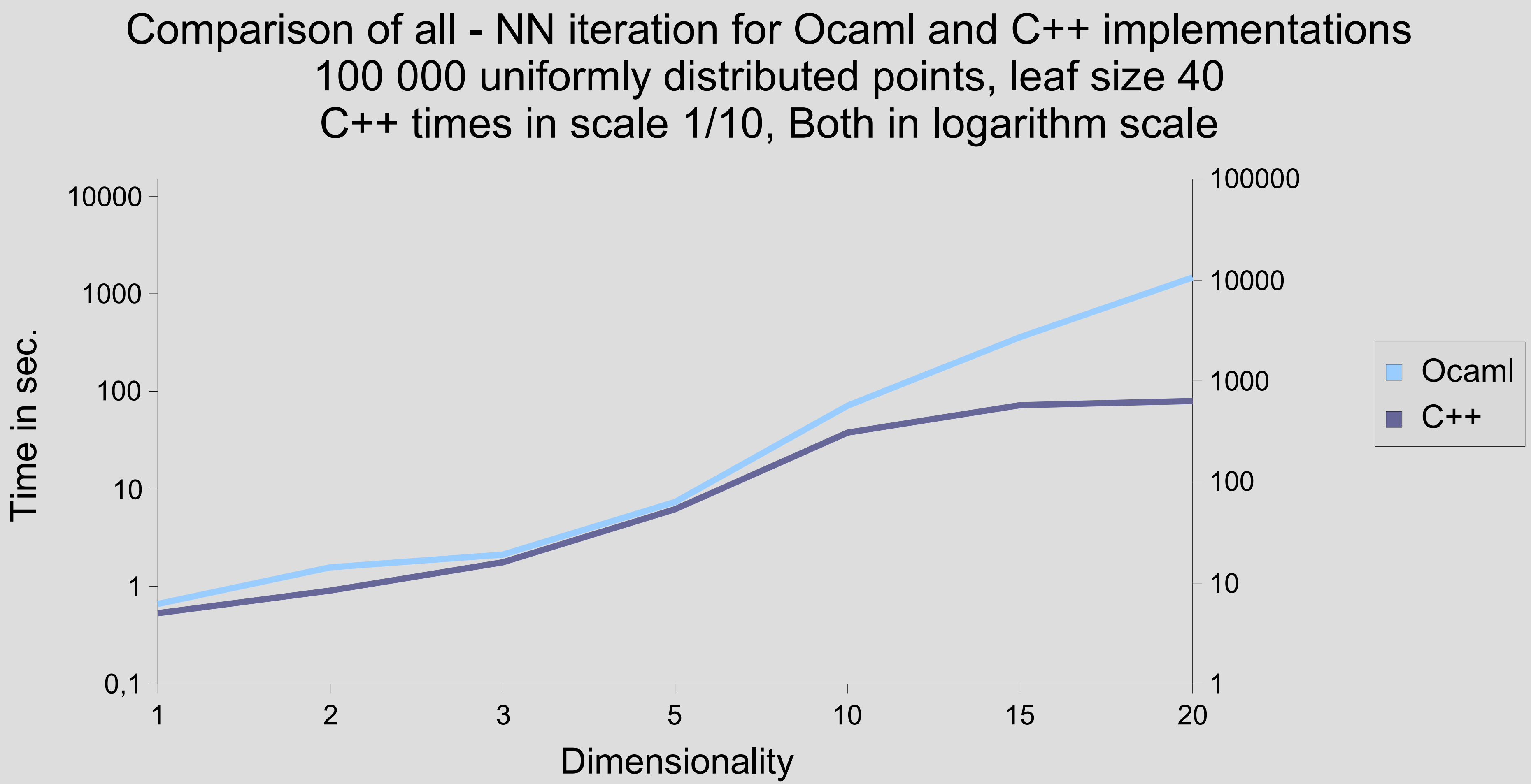
Experiments – artifical data

- Run on Celeron 2 Ghz, 248 MB RAM
- Entropy estimation for 100 000 3D normally distributed points, with quantization

Step	Building time	Iterating time	Entropy error
0	0,5500	1,3203	-0,001134730
0,00001	0,5576	1,3206	-0,000190042
0,0001	0,5612	1,3188	-0,004629920
0,001	0,5532	1,3158	-0,005963390
0,01	0,5500	1,1516	-0,036931000
0,1	0,5188	0,3140	-8,399770000
1	0,2564	0,0108	-19,362000000

Experiments – artifical data

- Run on Pentium IV 2 GHz and Celeron 2GHz



Open problems and future work

- Quantization
  - Low amplitude noise
  - Smoothing
  - Derivation of estimator with the quantization noise assumption
- Derivation estimation
- Mutual information estimation
  - $H(X)+H(Y)-H(X,Y)$
  - Direct nearest neighbor MI estimator [Kraskov] implementation in this framework
- Normalized MI experiments $\frac{H(X)+H(Y)}{H(X,Y)}$

References

[KoLe] Kozachenko L., Leonenko N., „On statistical estimation of entropy of random vector“, Problems Infor. Transmiss., 23(2):95–101, 1987

[NumRec] Numerical recepies in C, http://www.library.cornell.edu/nr/bookcpdf.html

[Kraskov] Kraskov A., Stogbauer H., Grassberger P., "Estimating Mutual Information", ArXiv cond-mat/0305641, 2003