A path following algorithm for graph matching

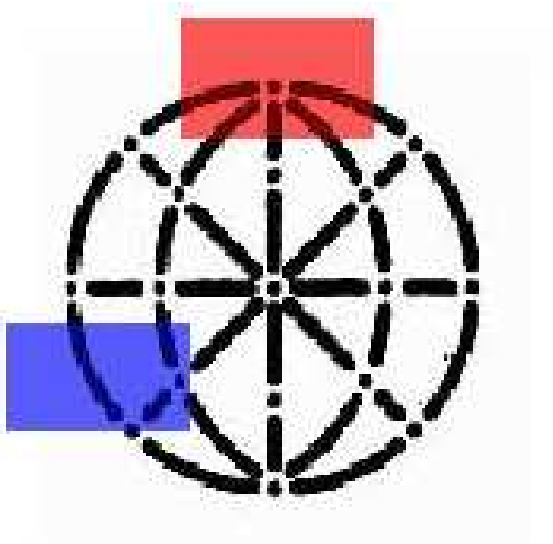
Mikhail Zaslavkiy [1] Francis Bach[2]
Jean-Philippe Vert[1]

[1]Mines ParisTech / Institut Curie / INSERM

[2]INRIA / Ecole normale superieure de Paris

National Institute of Informatics, Tokyo, Japan, June 26, 2008.

Image matching: pixel independent approach



$$C = \left[\begin{array}{c|cccc} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \\ \hline \mathbf{1} & 10 & 50 & 3 & 70 \\ \mathbf{2} & 2 & 100 & 10 & 6 \\ \mathbf{3} & 30 & 40 & 30 & 10 \\ \mathbf{4} & 50 & 15 & 43 & 120 \end{array}\right]$$

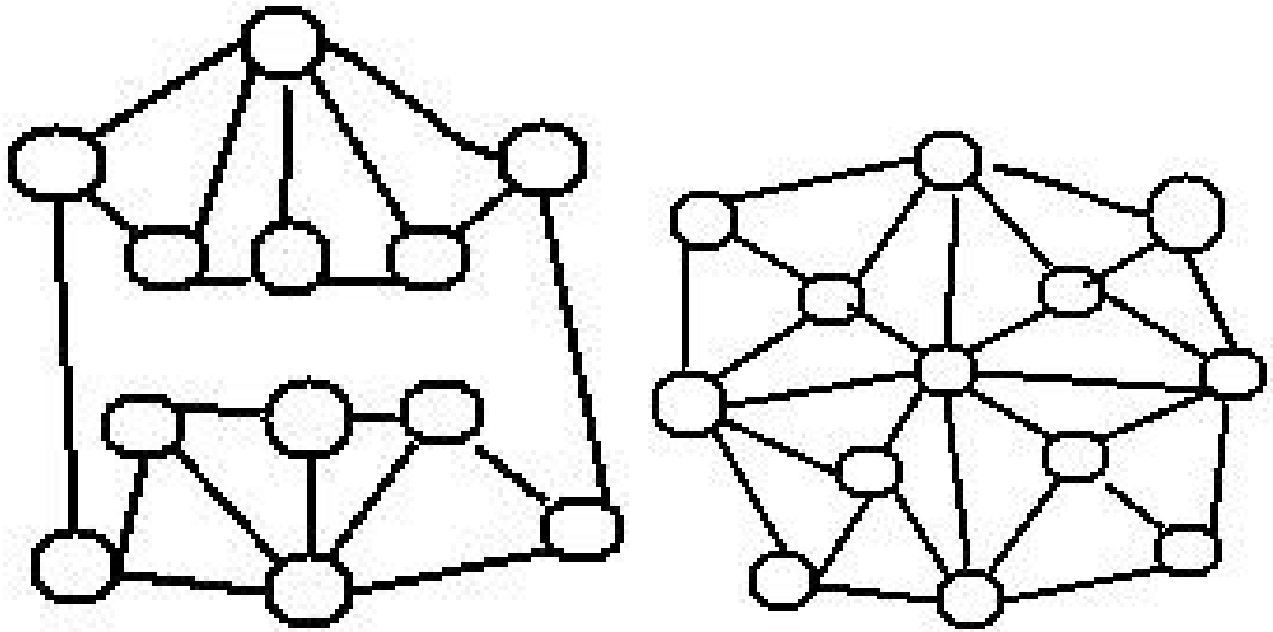
$C_{i,j}$-difference between context of i-th pixel in the first image and context of j-th pixel in the second one

P is a permutation matrix, $P(i,j) == 1$ if i is matched to j

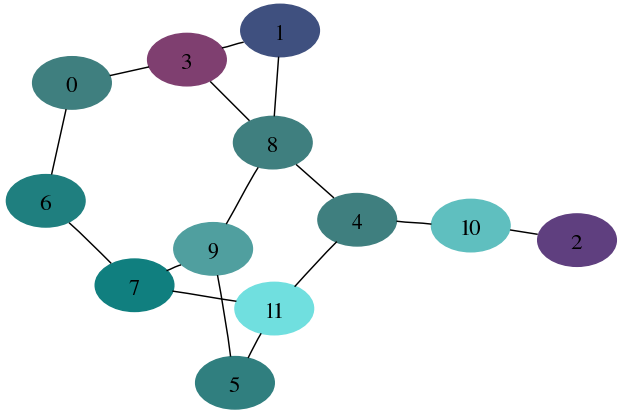
$\mathrm{tr}(C^T P) \rightarrow \min_P$ Hungarian algorithm: $\mathcal{O}(N^3)$

Motivation: Structural information

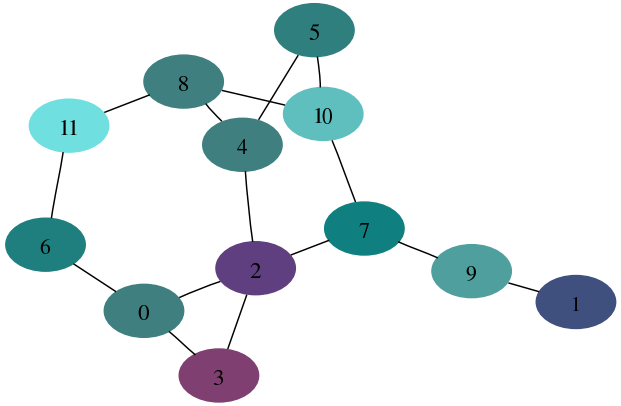
Matching may be more efficient if we take into account information about object structure.



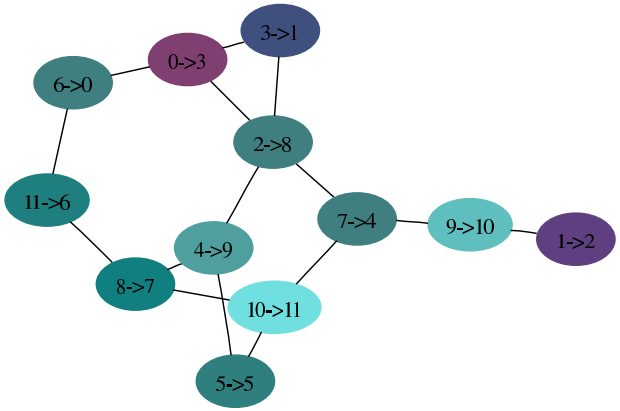
Problem description



(a) G



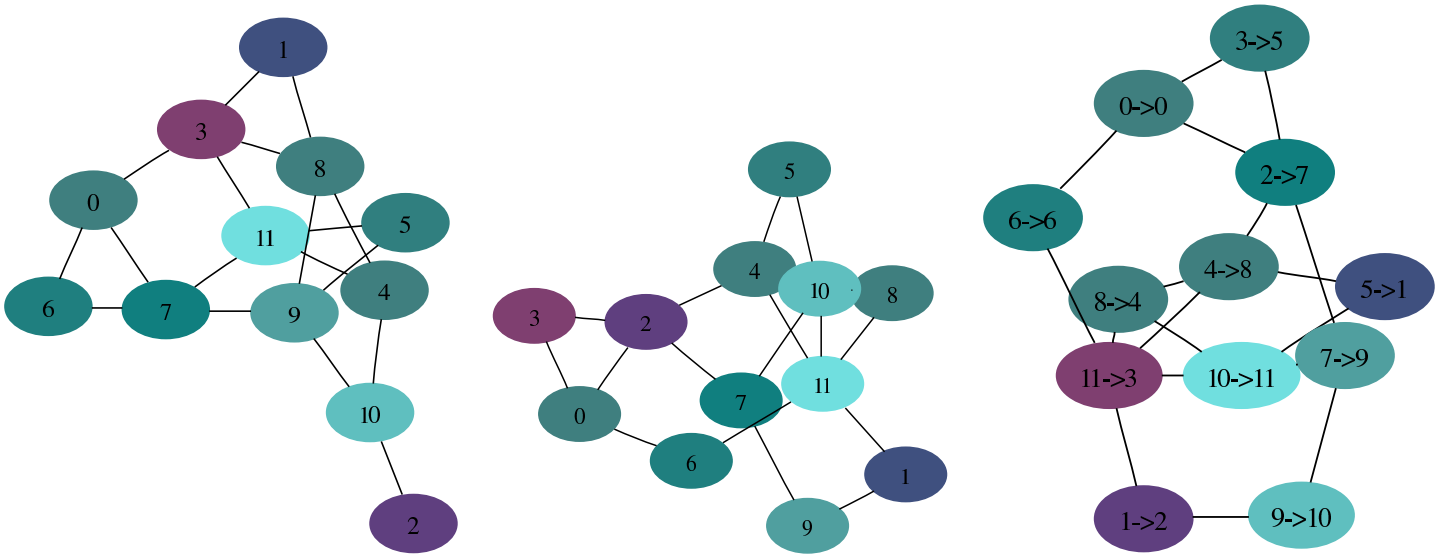
(b) H



(c) P(H)

Figure: Graph matching illustration. Isomorphic graphs.

Problem description



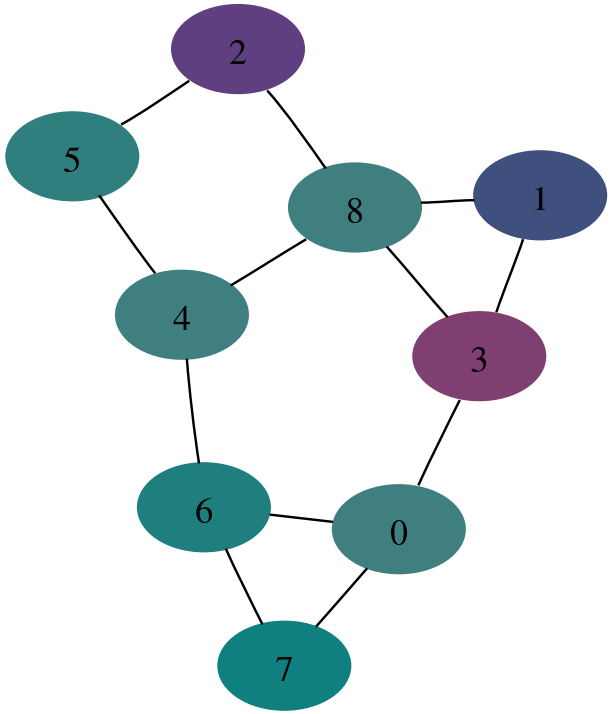
(a) G (b) H (c) P(H)

Figure: Graph matching illustration.Non-isomorphic graphs.

Mathematical formulation

$G = (V, E)$ where V is a set of vertices and $E \in V \times V$ is a set of edges.

Adjacency matrix $A_G \in 0, 1^{|V| \times |V|}$: $A_G(i,j) = 1$ if $(i,j) \in E$

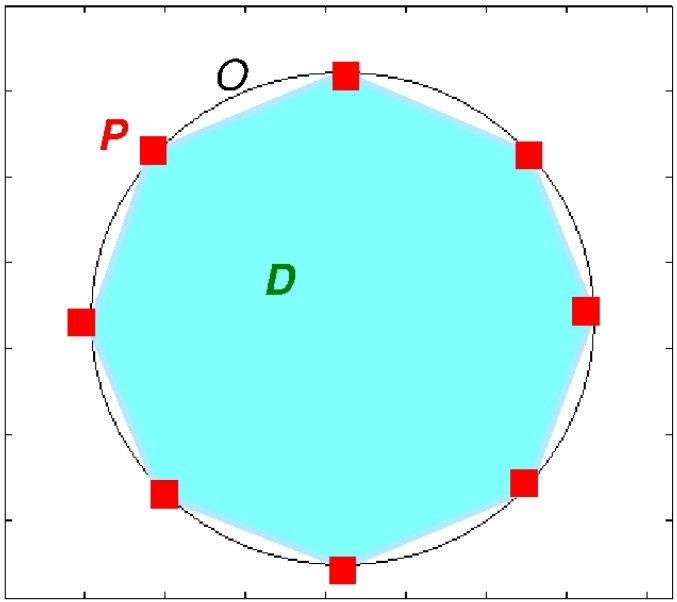


$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Matching: $F(P) = ||A_G - A_{P(H)}|| = ||A_G - PA_HP^T|| \rightarrow \min_P$ (1)

where P is a permutation matrix.

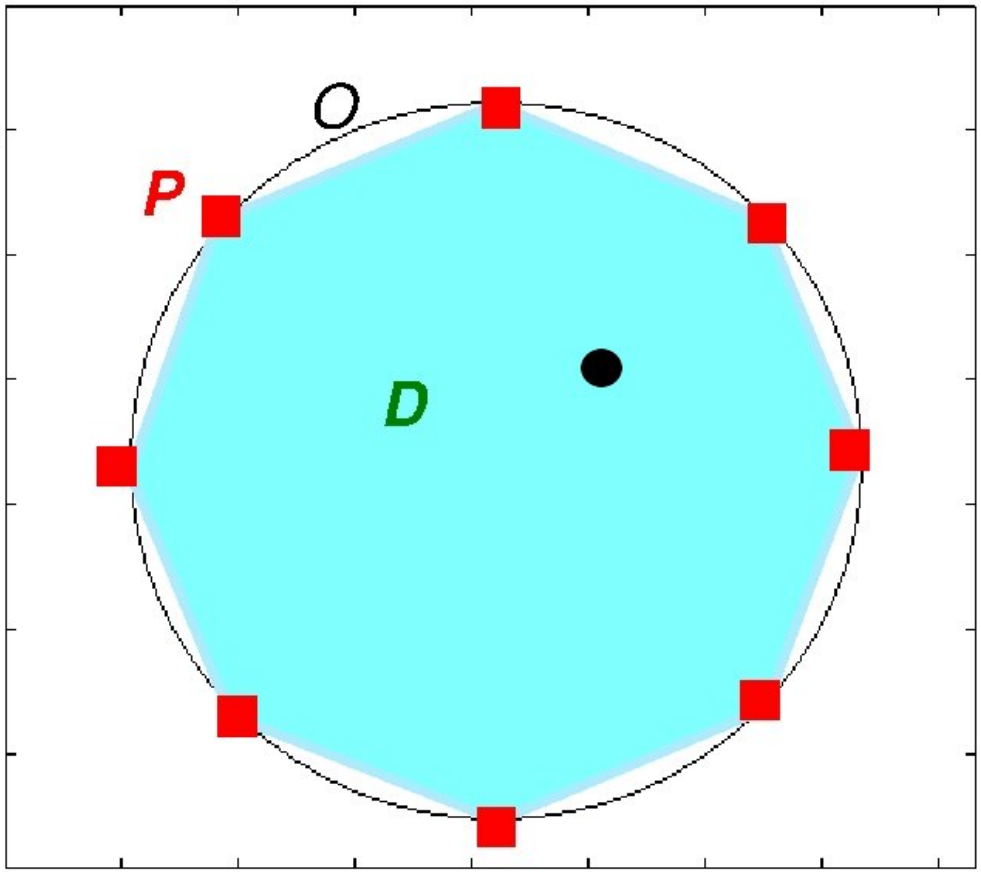
Problem structure



$$\mathbf{P}: P1_N = P^T 1_N = 1_N, \quad P \in \{0,1\}^{N \times N}$$
$$\mathbf{O}: P^T P = P P^T = 1$$
$$\mathbf{D}: P1_N = 1_N, \; P^T 1_N = 1_N$$

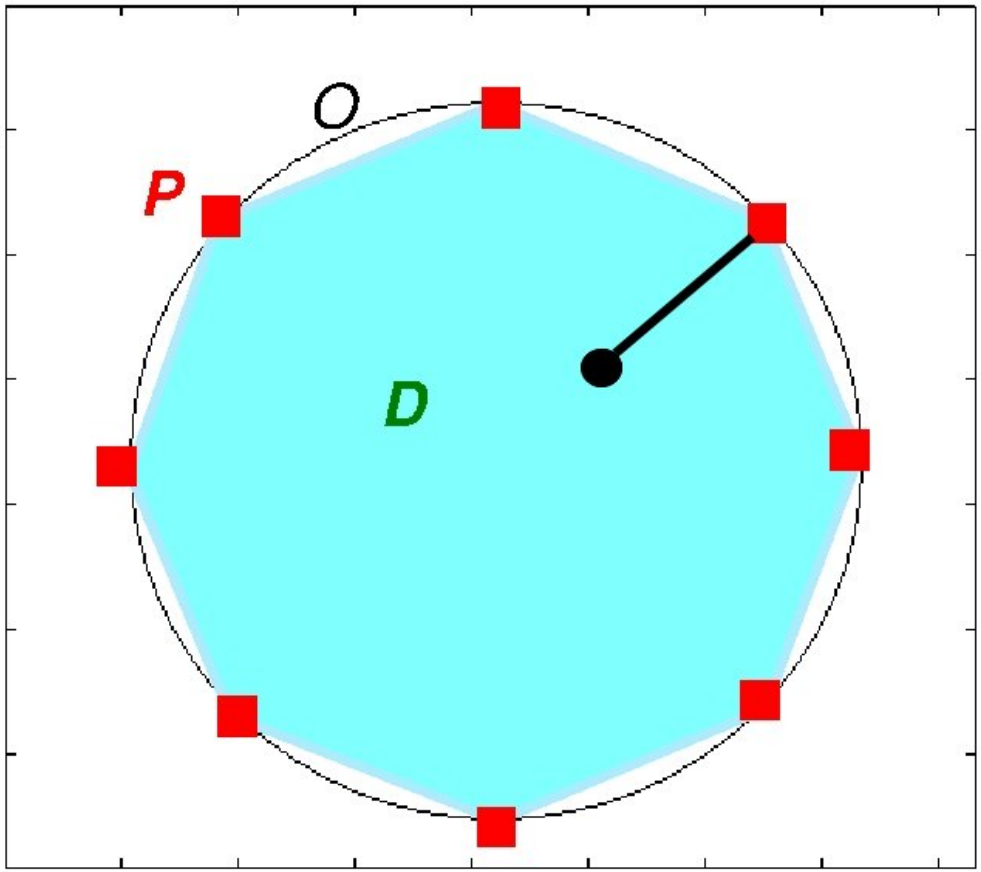
$$F(P) = ||A_G P - P A_H||_F^2 = \text{vec}(P)^T Q \text{vec}(P)$$

Problem structure: convex relaxation



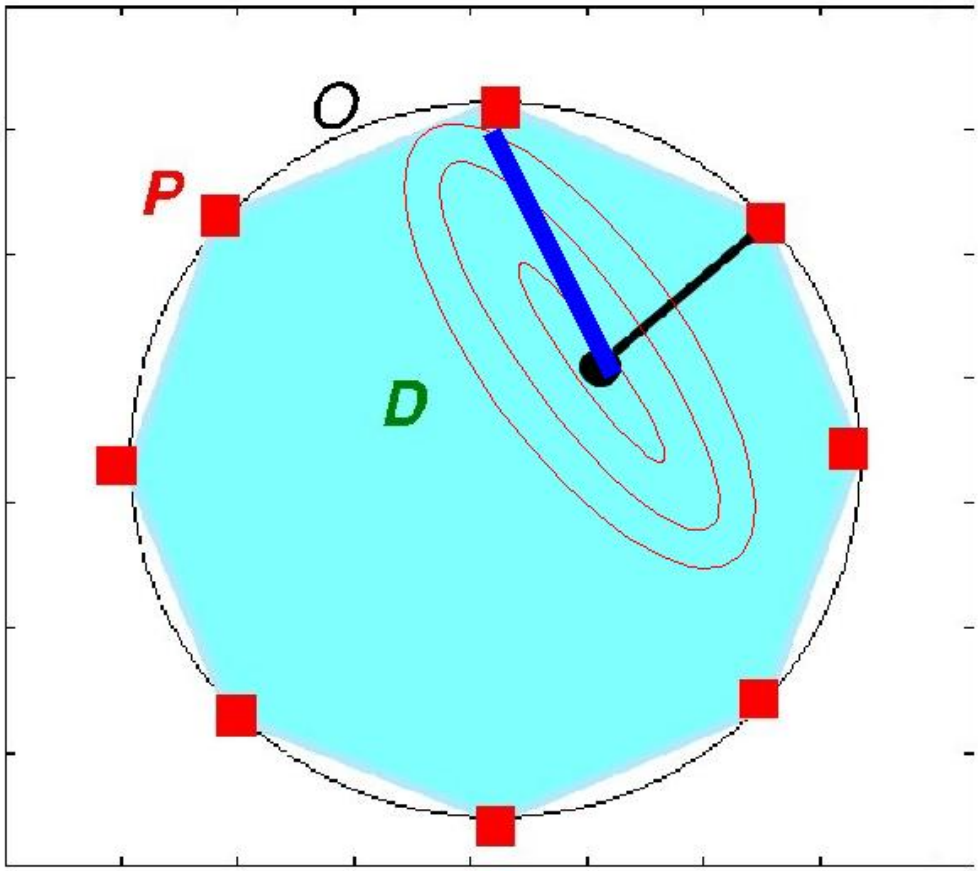
$$F(P) = ||A_G P - P A_H||_F^2 = \text{vec}(P)^T Q \text{vec}(P)$$

Problem structure:convex relaxation



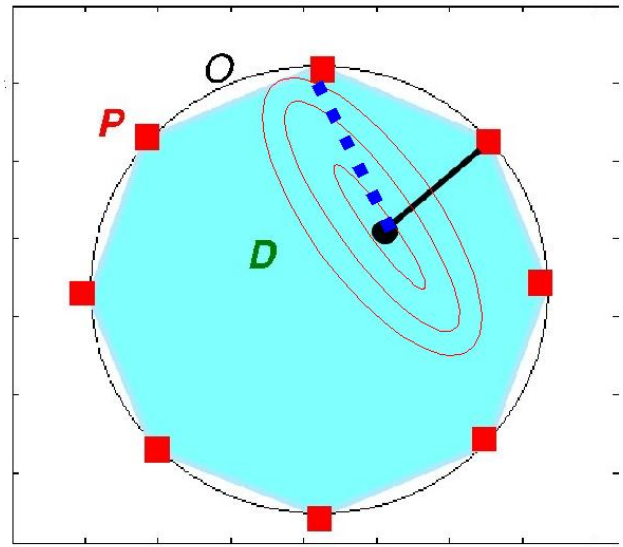
$$F(P) = ||A_G P - P A_H||_F^2 = \text{vec}(P)^T Q \text{vec}(P)$$

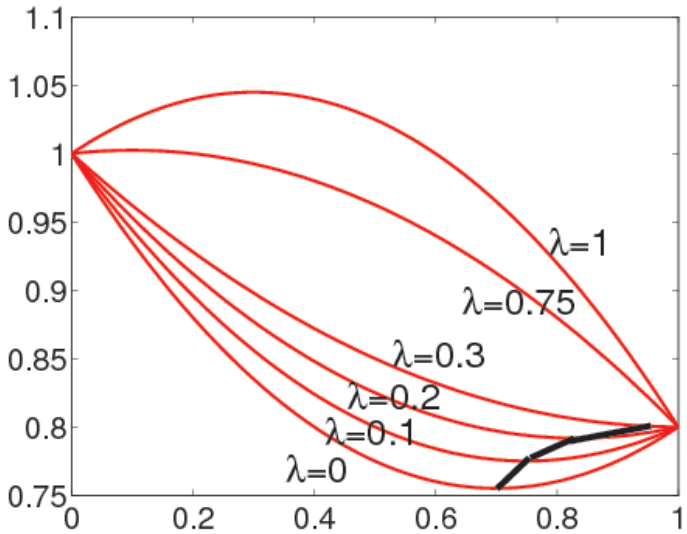
Problem structure:the global minimum



$$F(P) = ||A_G P - P A_H||_F^2 = \text{vec}(P)^T Q \text{vec}(P)$$

Solution: concave reshaping





$$F_0(P) = ||A_G P - P A_H||_F^2 = \text{vec}(P)^T Q \text{vec}(P)$$
$$F_1(P) = -tr(\Delta P) - \text{vec}(P)^T (L_G \otimes L_H) \text{vec}(P)$$
$$F_\lambda(P) = (1-\lambda) F_0(P) + \lambda F_1(P)$$

Vertex cost matrix

Let matrix C denote the node cost matrix. Then algorithm based only on this information:

$$\text{tr}(C^T P) \to \min_P . \tag{2}$$

Information on node similarities may be integrated in our method

$$F_\lambda^\alpha(P) = (1-\alpha)F_\lambda(P) + \alpha \text{tr}(C^T P) \to \min_P . \tag{3}$$

Random graphs:N=8

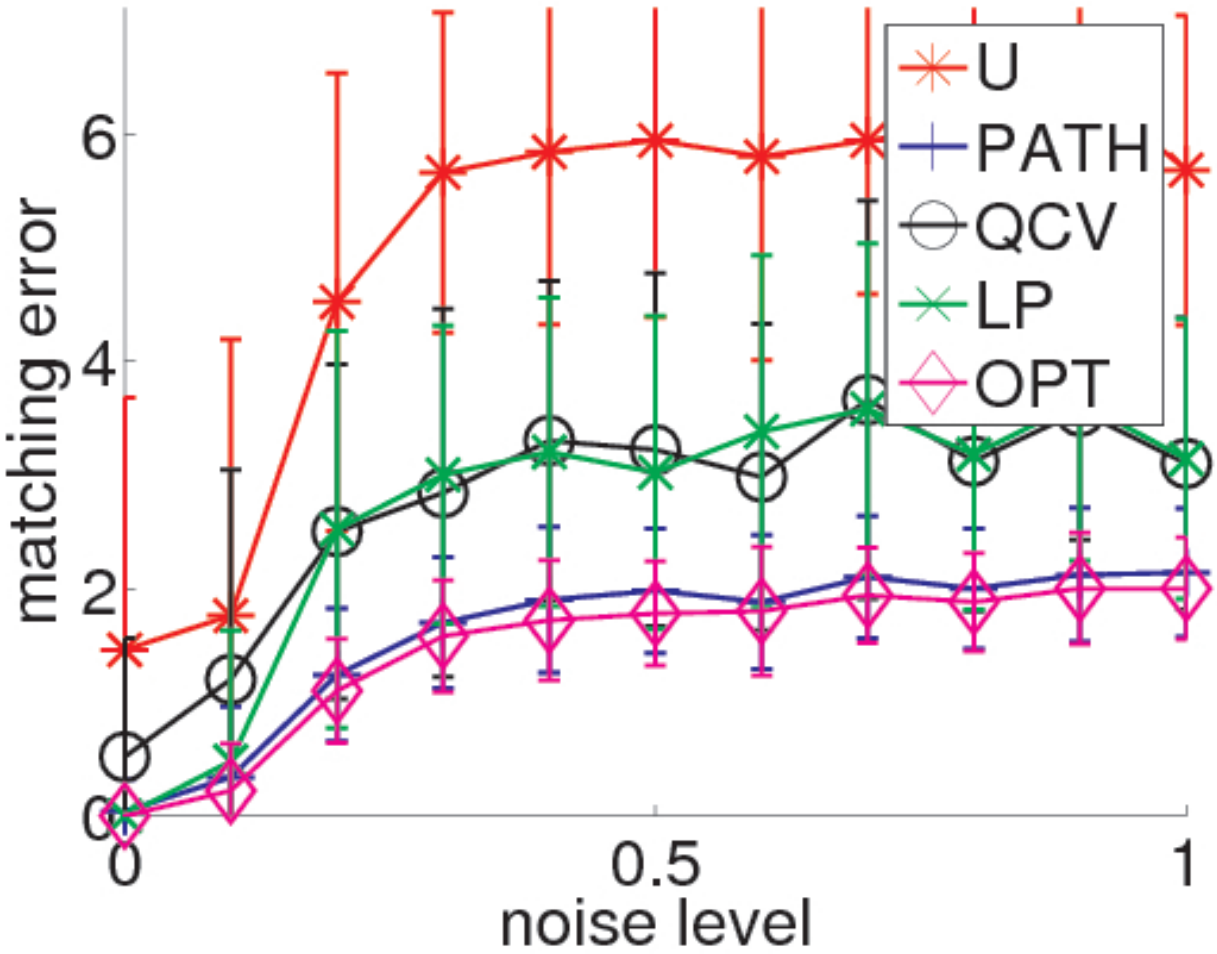


Figure: Precision as a noise function, U — Umeyama algorithm results, LP — linear programming algorithm, QCVL — convex function approach (F_0), PATH — path minimization algorithm, OPT — an exhaustive search (the global minimum).

Random graphs:N=20

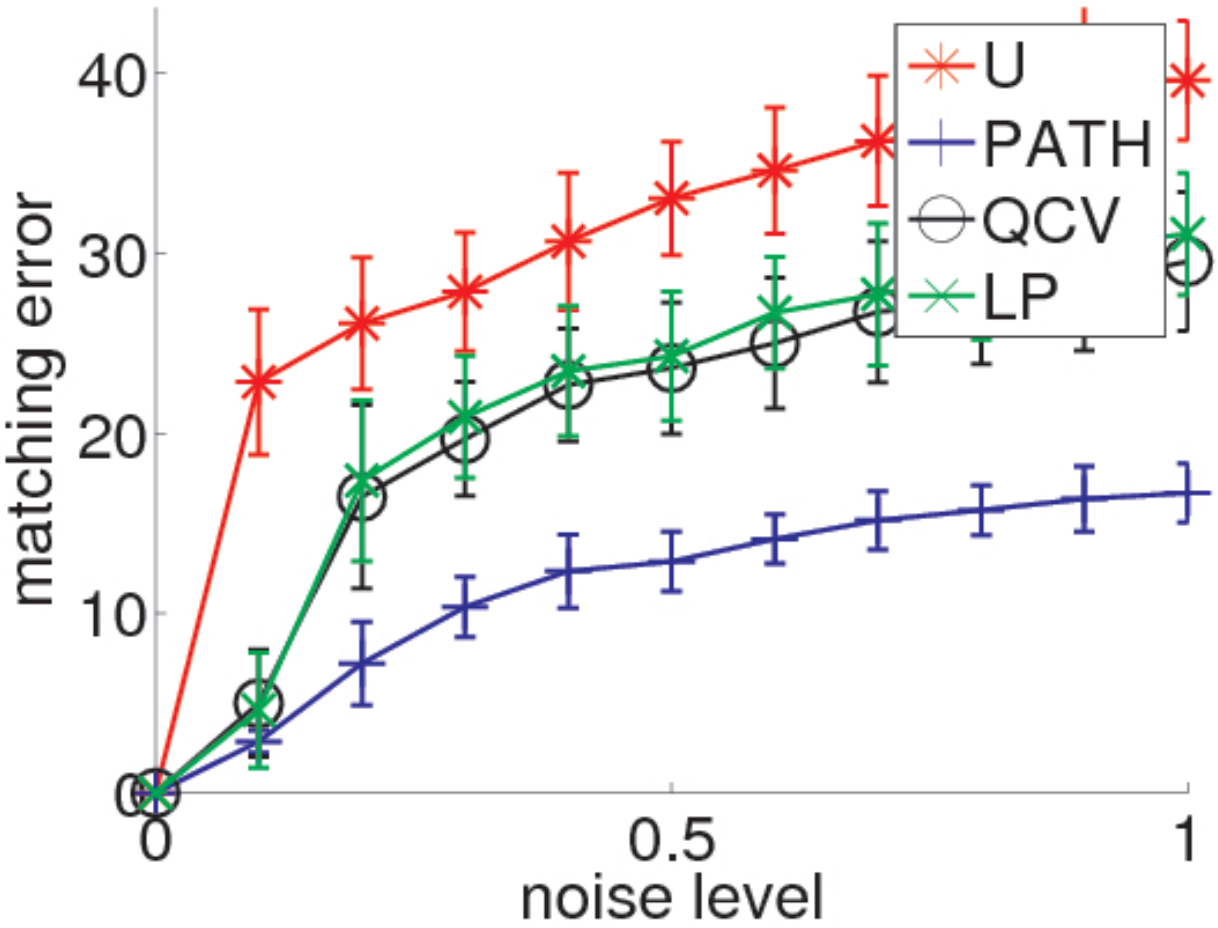


Figure: Precision as a noise function, U — Umeyama algorithm results, LP — linear programming algorithm, QCV — convex function approach (F_0), PATH — path minimization algorithm.

Random graphs:N=100

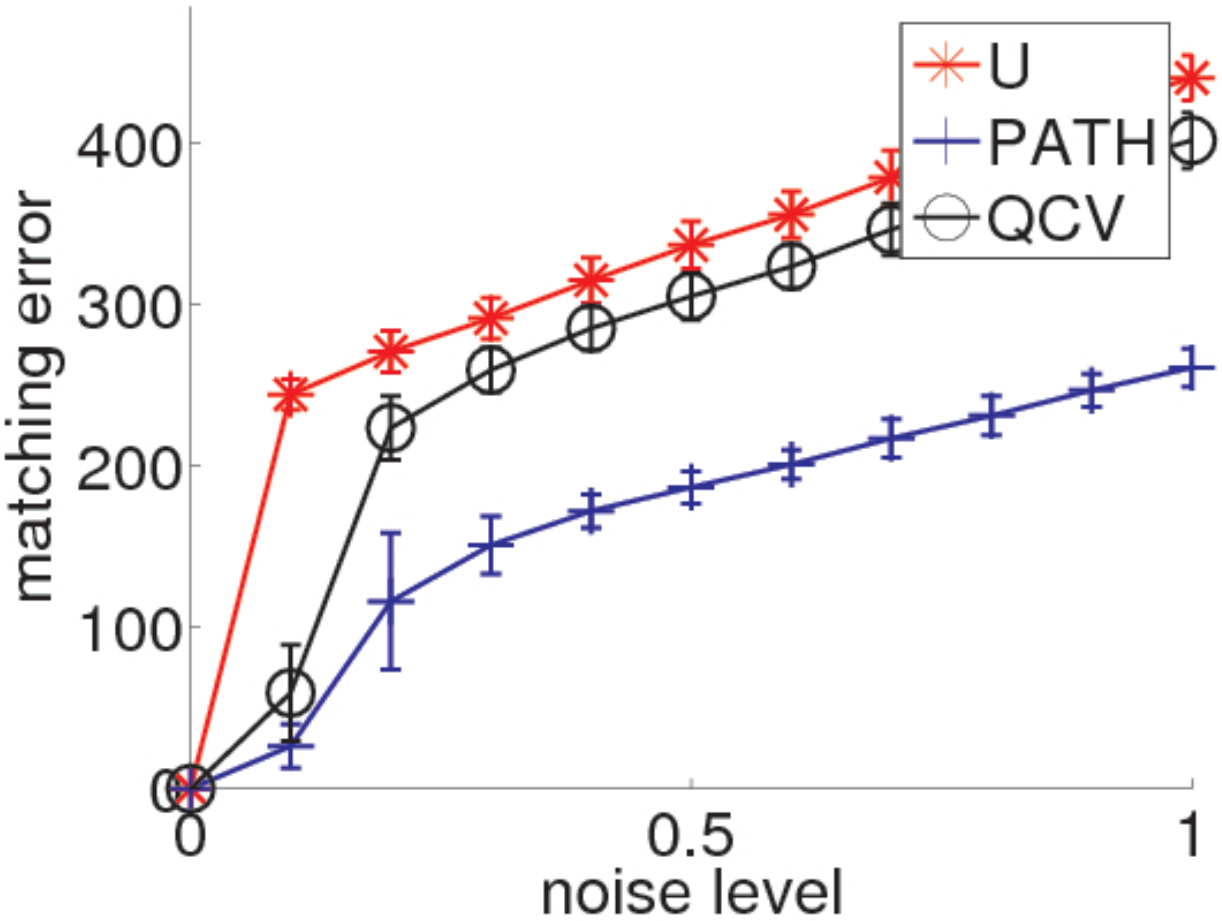


Figure: Precision as a noise function, U — Umeyama algorithm results, QCV — convex function approach (F_0), PATH — path minimization algorithm.

Algorithm complexity

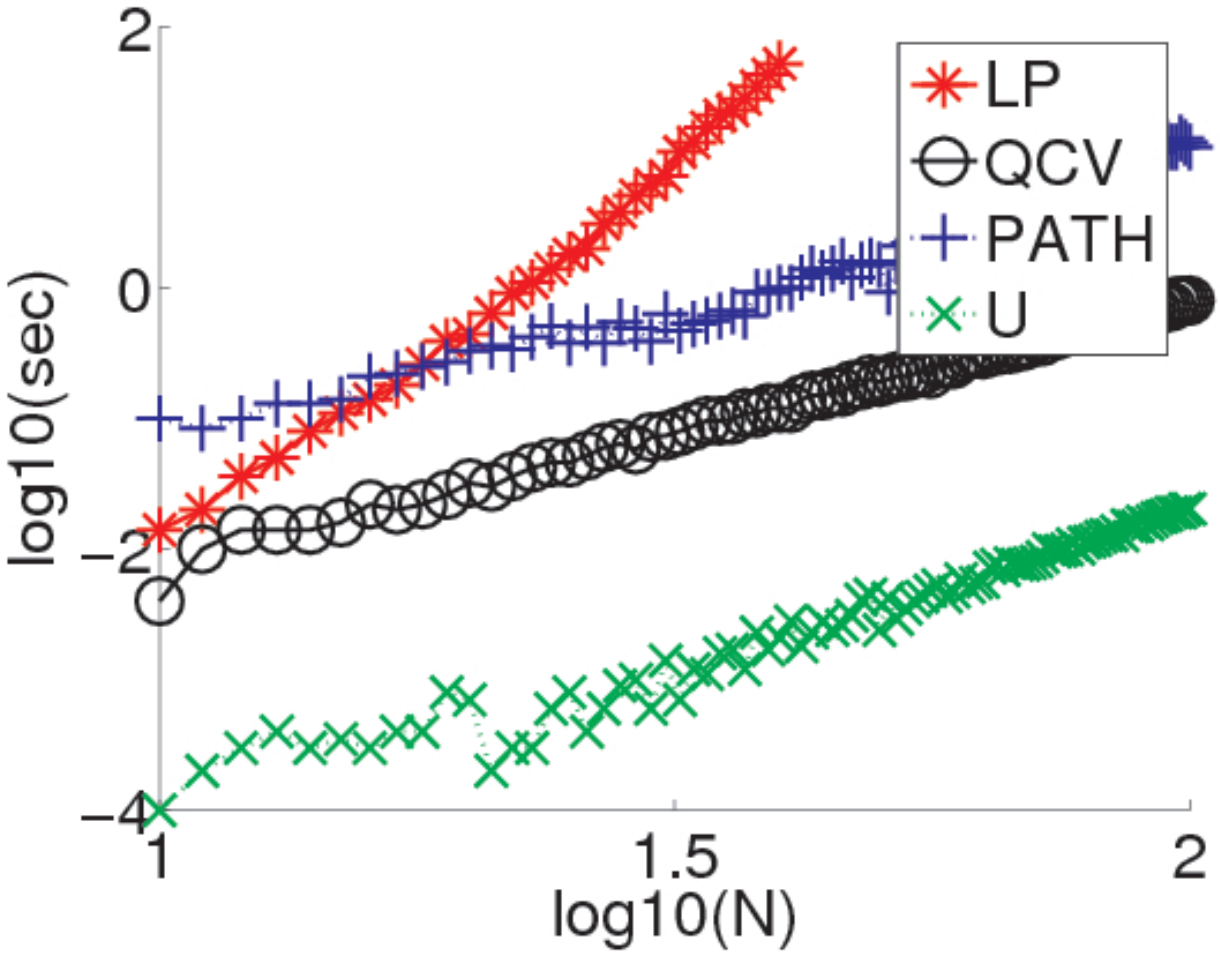
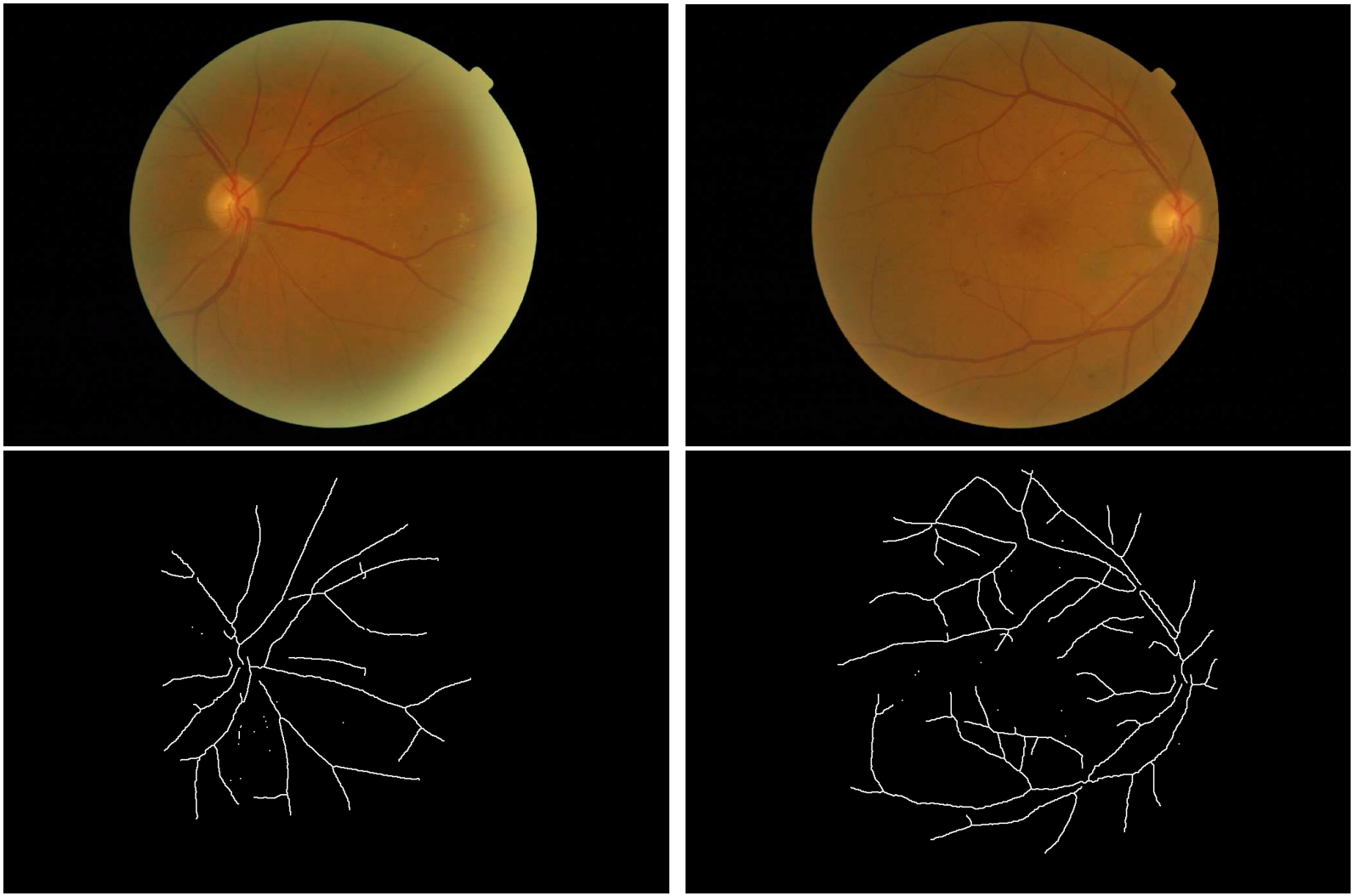
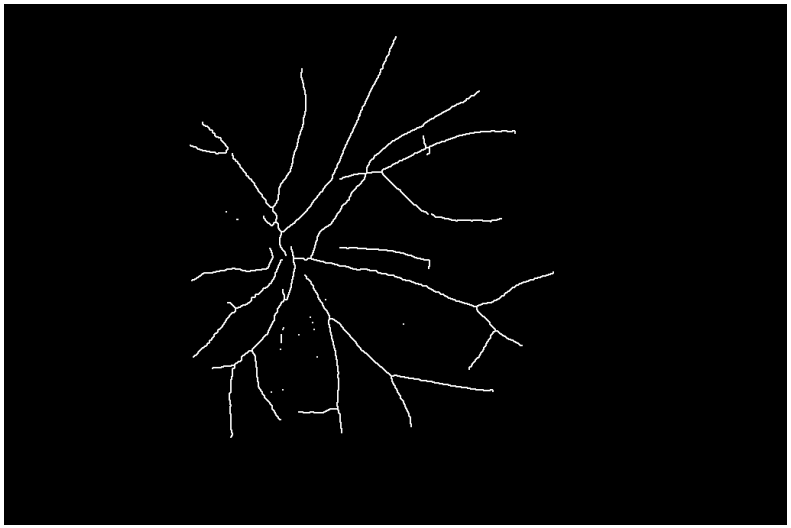


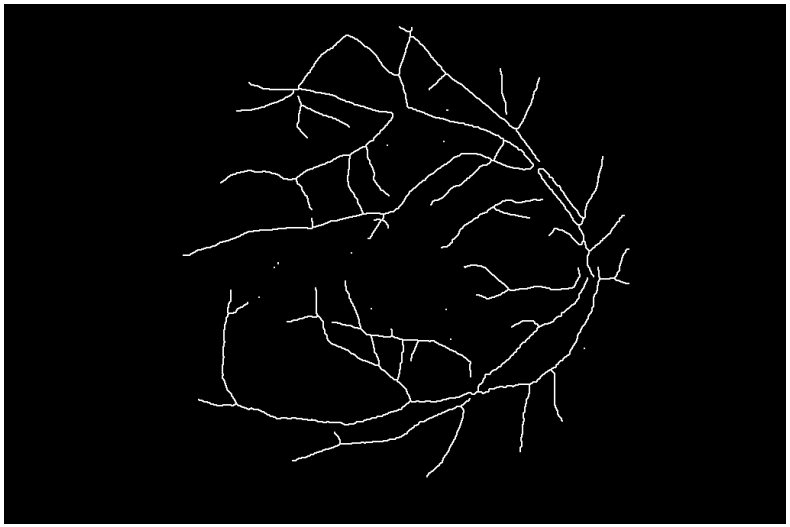
Figure: Timing of U, LP, QCV and PATH algorithms as a function of graph size. Noise level is 0.3. Slope: $tan_{LP} = 6.67, tan_U = tan_{QCV} = tan_{PATH} = 3.3$

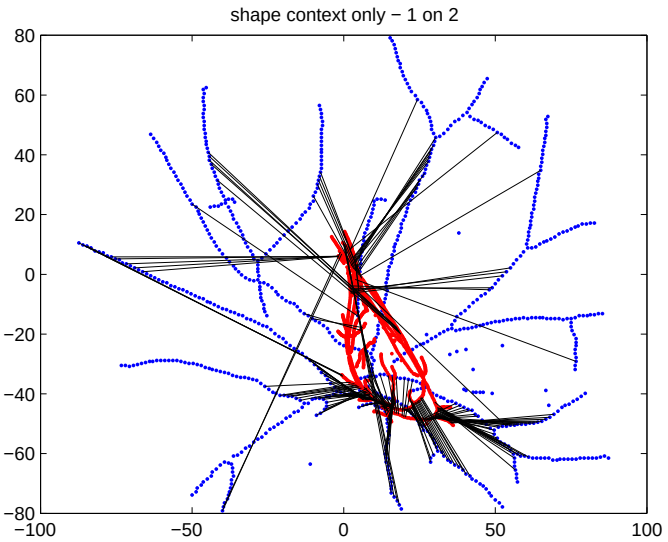
Eye vessels image processing

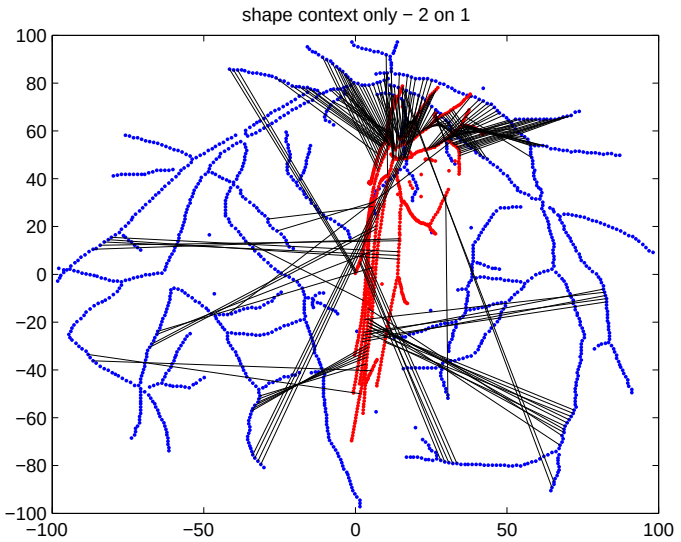


Eye vessels image processing: Shape context

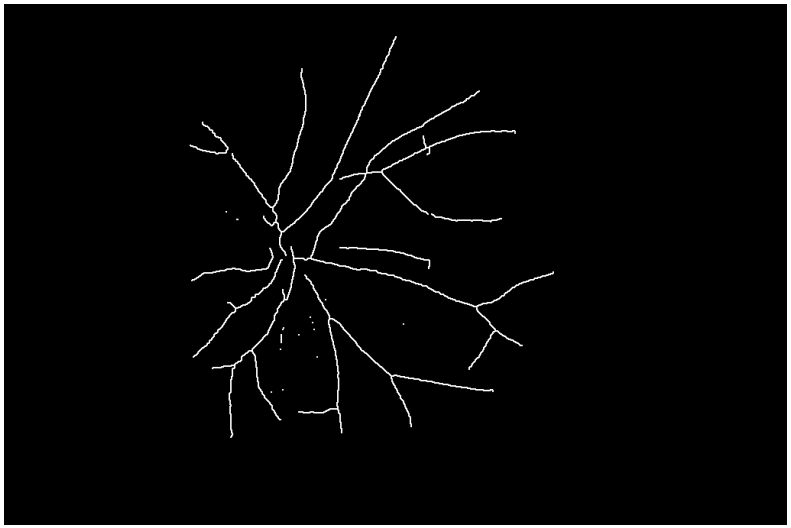


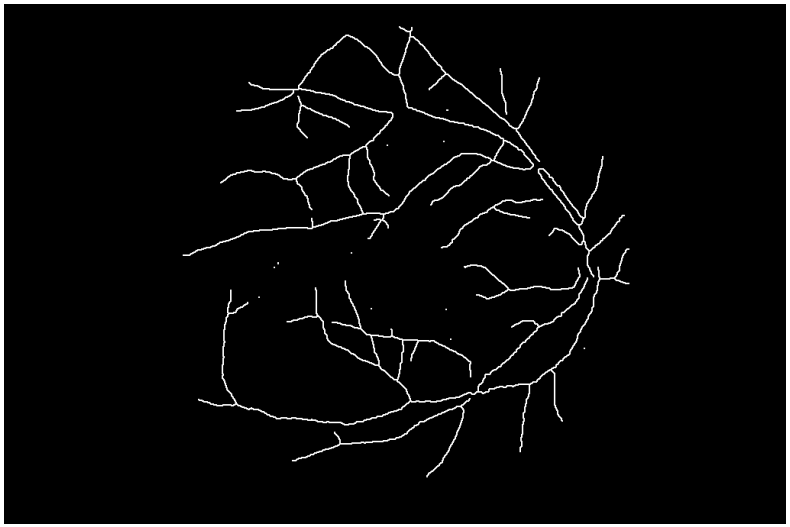


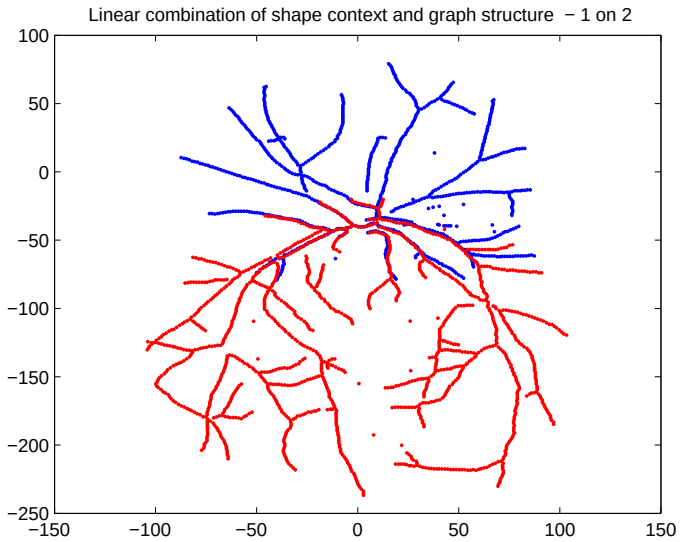


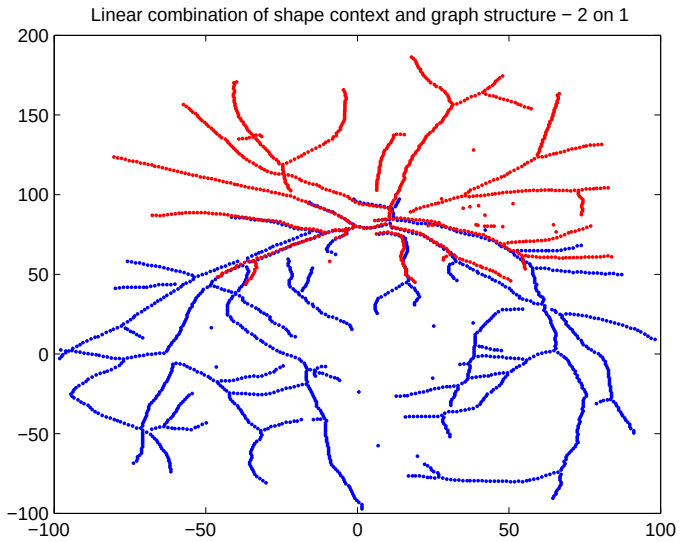


Combination of shape context and structural information









Recognition of chinese characters

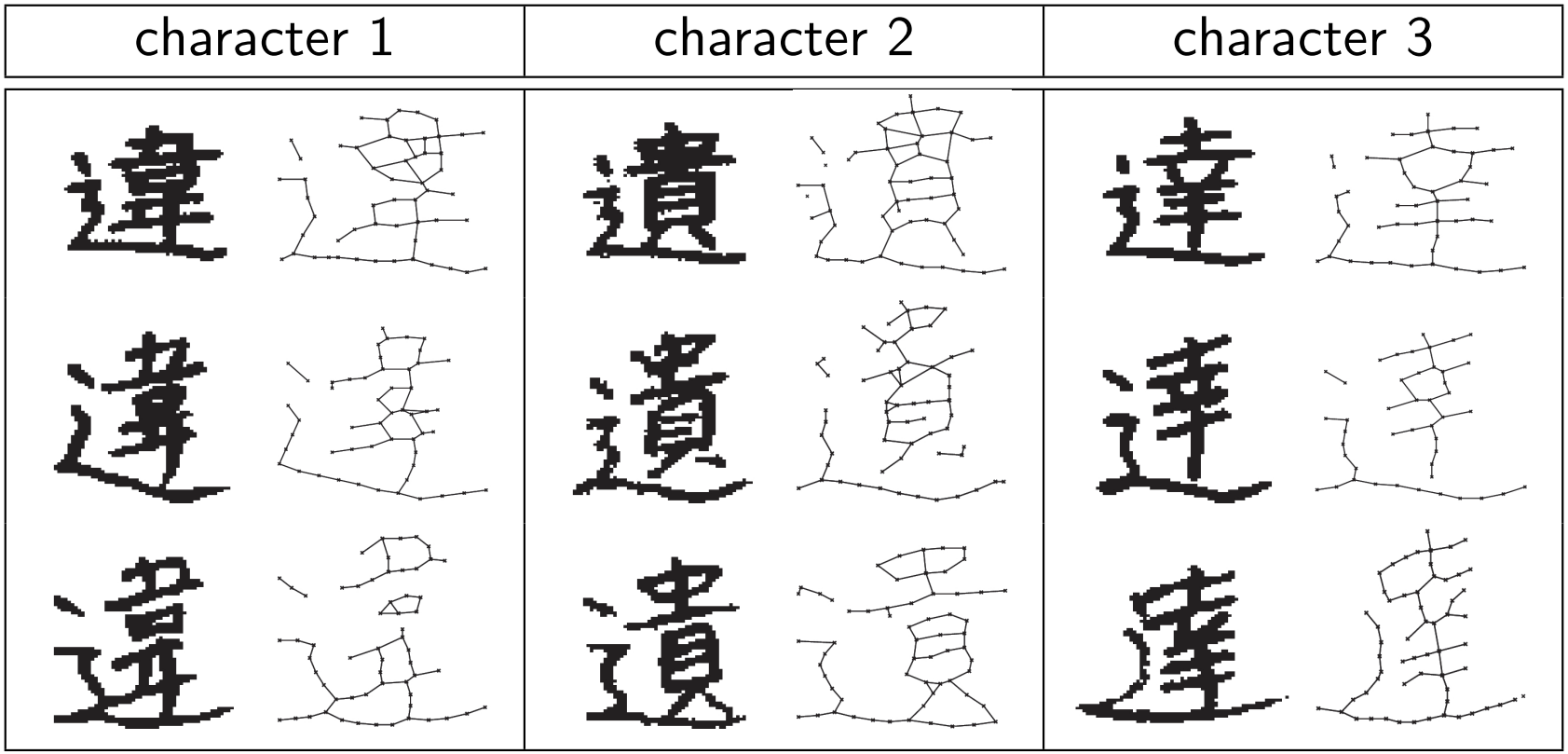


Figure: Chinese characters from the ETL9B dataset.

Recognition of chinese characters

Table: Classification of chinese characters. (CV, STD)—mean and standard deviation of test error over cross-validation runs (five folds, 50 repetitions)

Method	CV	STD
Linear SVM	0.377	$\pm$ 0.090
SVM with gaussian kernel	0.359	$\pm$ 0.076
KNN (PATH) (α=1): shape context	0.399	$\pm$ 0.081
KNN (PATH) (α=0.4)	0.248	$\pm$ 0.075
KNN (PATH) (α=0): pure graph matching	0.607	$\pm$ 0.072
KNN (U) (α=0.9): α best choice	0.382	$\pm$ 0.077
KNN (QCV) (α=0.3): α best choice	0.295	$\pm$ 0.061

Experiment results for QAPLIB benchmark datasets.

QAP	MIN	PATH	QPB	GRAD	U
chr12c	11156	**18048**	20306	19014	40370
chr15a	9896	**19086**	26132	30370	60986
chr15c	9504	**16206**	29862	23686	76318
chr20b	2298	**5560**	6674	6290	10022
chr22b	6194	**8500**	9942	9658	13118
esc16b	292	300	**296**	298	306
rou12	235528	**256320**	278834	273438	295752
rou15	354210	391270	**381016**	457908	480352
rou20	725522	**778284**	804676	840120	905246
tai10a	135028	**152534**	165364	168096	189852
tai15a	388214	**419224**	455778	451164	483596
tai17a	491812	**530978**	550852	589814	620964
tai20a	703482	**753712**	799790	871480	915144
tai30a	1818146	**1903872**	1996442	2077958	2213846
tai35a	2422002	**2555110**	2720986	2803456	2925390

Conclusion

- Precise
- Fast
- Scalable: $\mathcal{O}(N^2)$ in memory
- Works equally well on dense graphs
- GraphM package: U, QCV, LP, PATH algorithms

Future work

- Further optimization
- Tighter convex and concave approximations