Efficient neighbourhood computing for discrete rigid transformation graph search

Yukiko Kenmochi Phuc Ngo Hugues Talbot Nicolas Passat

September 10, 2014

Background and Motivation

Background

- Problem of rigid image registration is usually formulated in the **continuous** domain, in the context of optimisation.

Background and Motivation

Background

- Problem of rigid image registration is usually formulated in the **continuous** domain, in the context of optimisation.
- This often requires **re-digitizing** results after transformation, and sometimes causes unwanted artifacts.

Background and Motivation

Background

- Problem of rigid image registration is usually formulated in the **continuous** domain, in the context of optimisation.
- This often requires **re-digitizing** results after transformation, and sometimes causes unwanted artifacts.

Motivation and clues

- Is it possible to avoid this re-digitization ?

Background and Motivation

Background

- Problem of rigid image registration is usually formulated in the **continuous** domain, in the context of optimisation.
- This often requires **re-digitizing** results after transformation, and sometimes causes unwanted artifacts.

Motivation and clues

- Is it possible to avoid this re-digitization ?

  Yes, a **fully discrete approach** allows to transform images pixel by pixel.

Background and Motivation

Background

- Problem of rigid image registration is usually formulated in the **continuous** domain, in the context of optimisation.
- This often requires **re-digitizing** results after transformation, and sometimes causes unwanted artifacts.

Motivation and clues

- Is it possible to avoid this re-digitization ?

  Yes, a **fully discrete approach** allows to transform images pixel by pixel.
- How to **explore** efficiently such a discrete parameter space ?

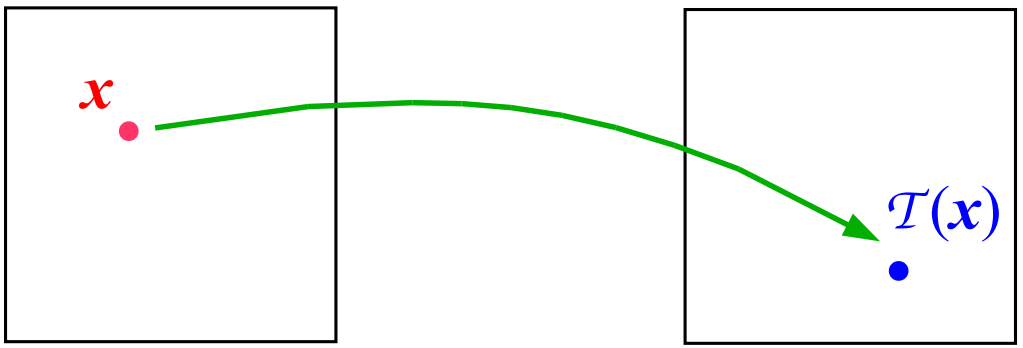
Rigid transformation on $\mathbb{R}^2$

Definition (Rigid transformation $\mathcal{T} : \mathbb{R}^2 \to \mathbb{R}^2$)

A rigid transformation is a bijection defined for any $\boldsymbol{x} = (x, y) \in \mathbb{R}^2$ *as*

$$\mathcal{T}(\boldsymbol{x}) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

with $a_1, a_2 \in \mathbb{R}$ *and* $\theta \in [0, 2\pi[$.



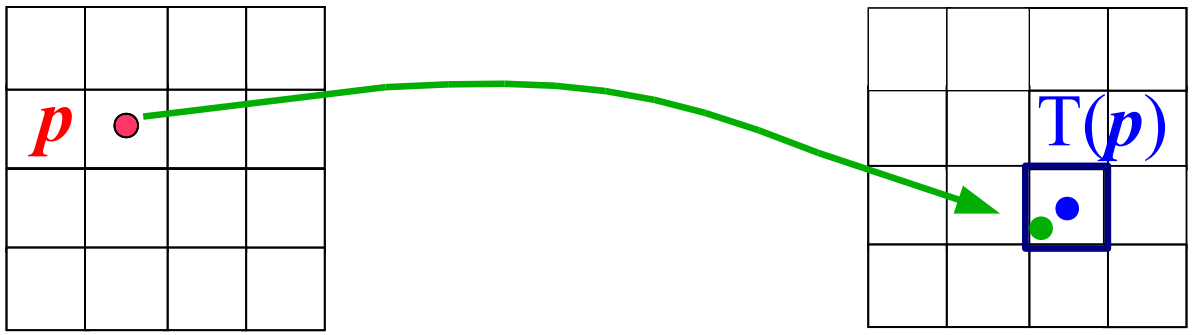
Rigid transformation on $\mathbb{Z}^2$

Definition (Digital rigid transformation $T : \mathbb{Z}^2 \to \mathbb{Z}^2$)

A digital rigid transformation on $\mathbb{Z}^2$ is defined for any $\boldsymbol{p} = (p, q) \in \mathbb{Z}^2$ as

$$T(\boldsymbol{p}) = D \circ \mathcal{T}(\boldsymbol{p}) = \begin{pmatrix} [p\cos\theta - q\sin\theta + a_1] \\ [p\sin\theta + q\cos\theta + a_2] \end{pmatrix}$$

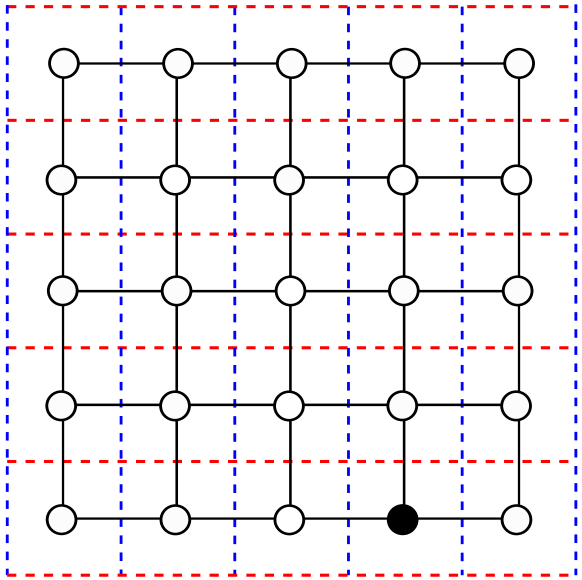
where $D : \mathbb{R}^2 \to \mathbb{Z}^2$ is a digitization (a rounding function).

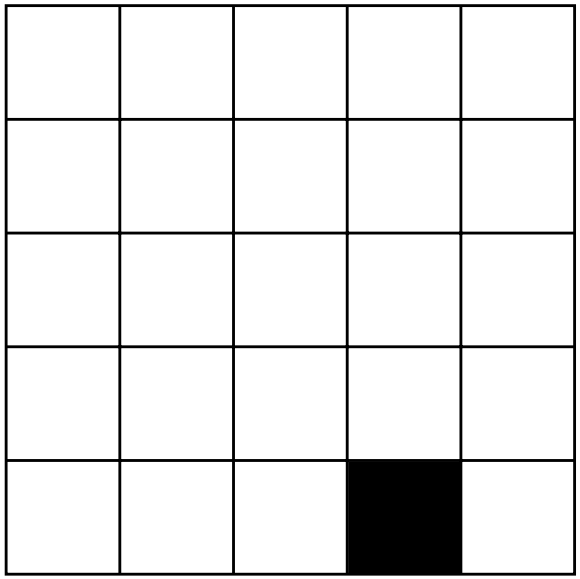


Discontinuities of rigid transformations in $\mathbb{Z}^2$

Discontinuities of rigid transformations on $\mathbb{Z}^2$

$$T(\boldsymbol{p}) = D \circ \mathcal{T}(\boldsymbol{p}) = \begin{pmatrix} [p\cos\theta - q\sin\theta + a_1] \\ [p\sin\theta + q\cos\theta + a_2] \end{pmatrix}$$

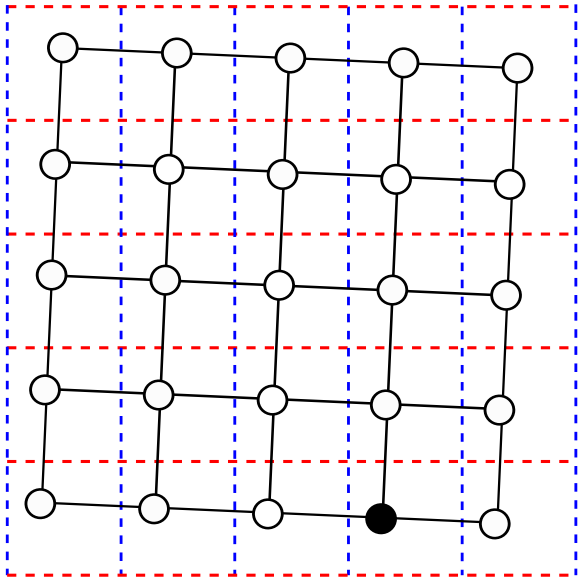


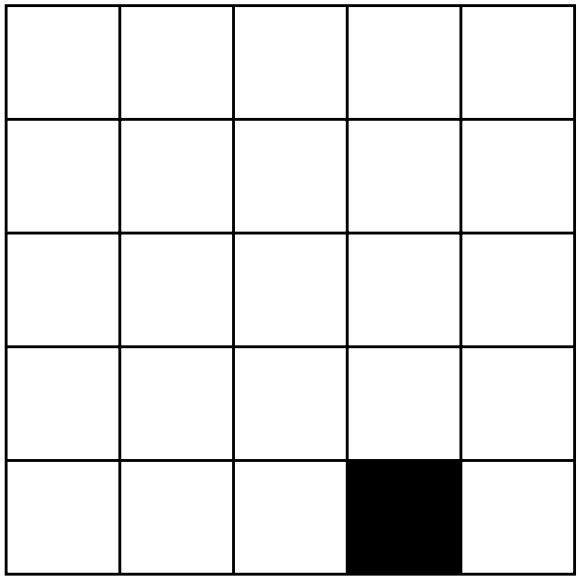


Discontinuities of rigid transformations in $\mathbb{Z}^2$

Discontinuities of rigid transformations on $\mathbb{Z}^2$

$$T(\boldsymbol{p}) = D \circ \mathcal{T}(\boldsymbol{p}) = \begin{pmatrix} [p\cos\theta - q\sin\theta + a_1] \\ [p\sin\theta + q\cos\theta + a_2] \end{pmatrix}$$

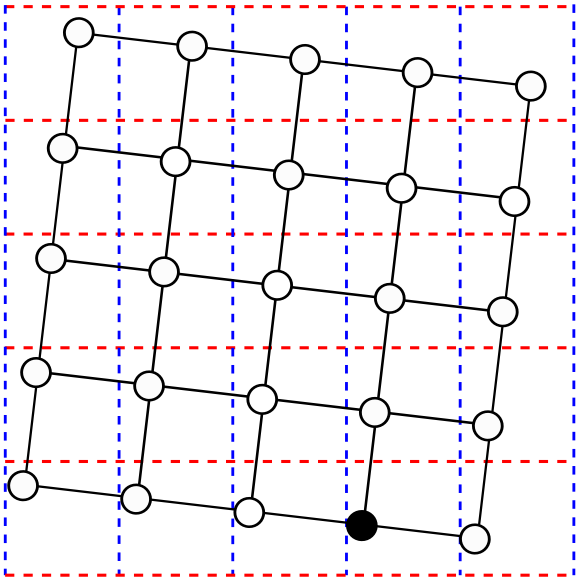


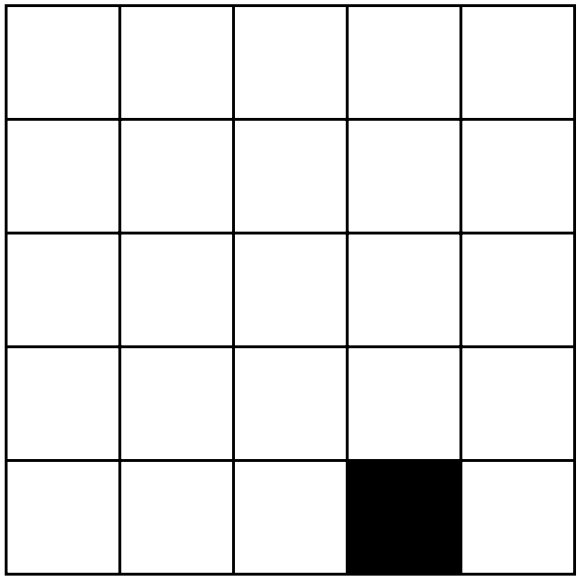


Discontinuities of rigid transformations in $\mathbb{Z}^2$

Discontinuities of rigid transformations on $\mathbb{Z}^2$

$$T(\boldsymbol{p}) = D \circ \mathcal{T}(\boldsymbol{p}) = \begin{pmatrix} [p\cos\theta - q\sin\theta + a_1] \\ [p\sin\theta + q\cos\theta + a_2] \end{pmatrix}$$

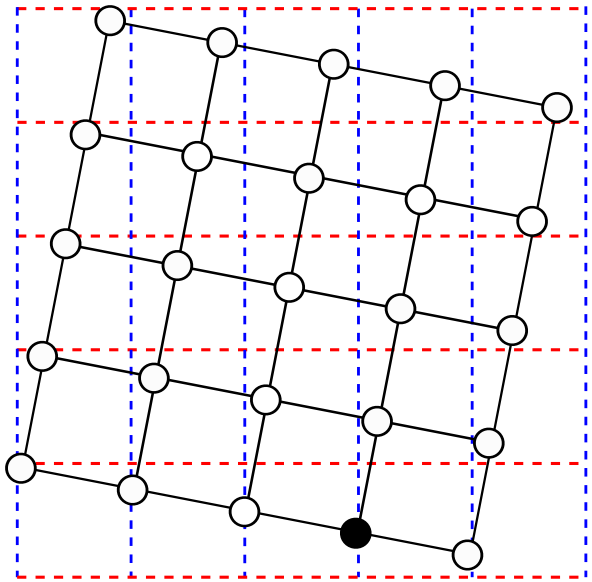


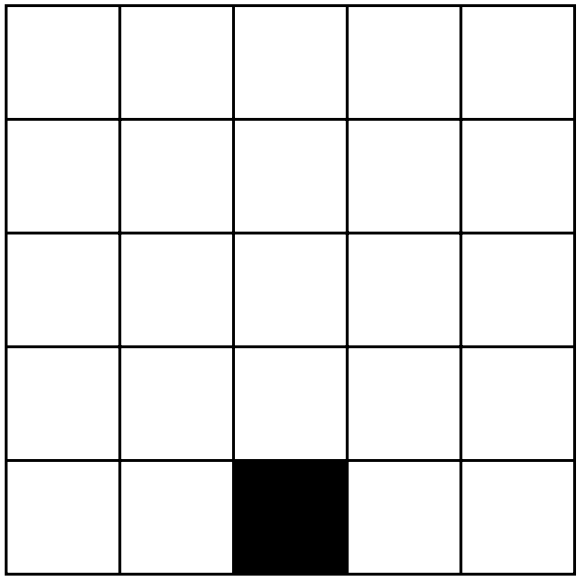


Discontinuities of rigid transformations in $\mathbb{Z}^2$

Discontinuities of rigid transformations on $\mathbb{Z}^2$

$$T(\boldsymbol{p}) = D \circ \mathcal{T}(\boldsymbol{p}) = \begin{pmatrix} [p\cos\theta - q\sin\theta + a_1] \\ [p\sin\theta + q\cos\theta + a_2] \end{pmatrix}$$

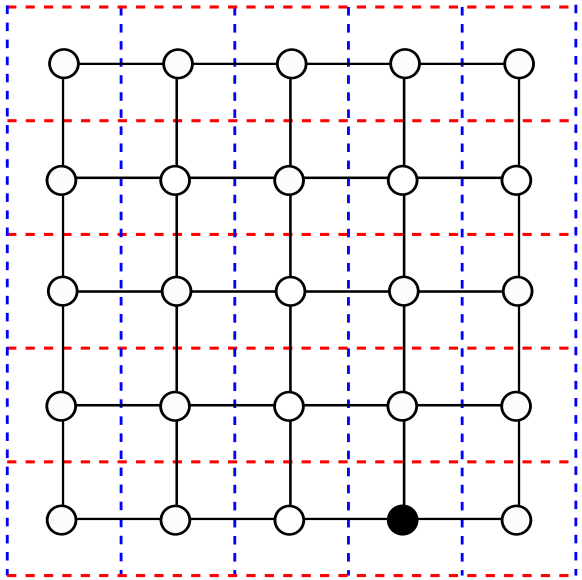


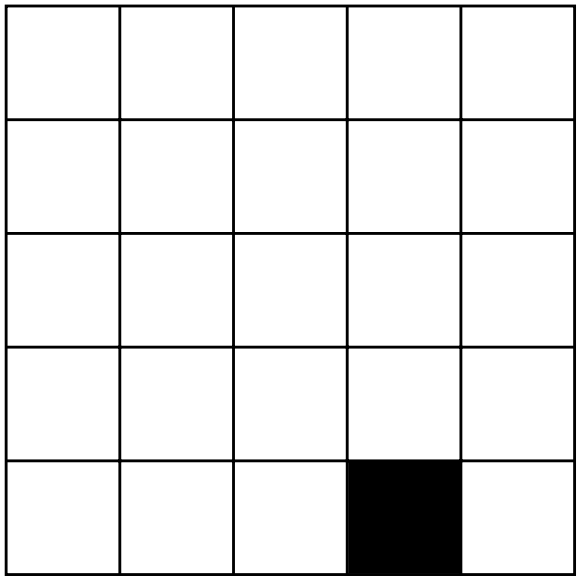


Discrete rigid transformation

Definition

*A **discrete rigid transformation** (DRT) is the set of all the rigid transformations that generate a same image.*

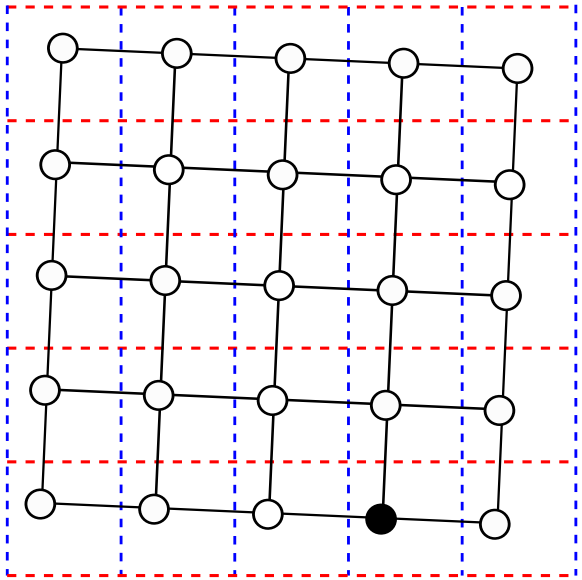


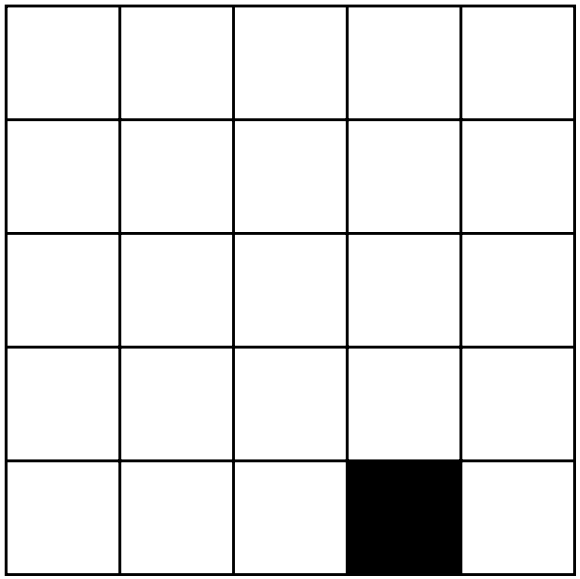


Discrete rigid transformation

Definition

*A **discrete rigid transformation** (DRT) is the set of all the rigid transformations that generate a same image.*

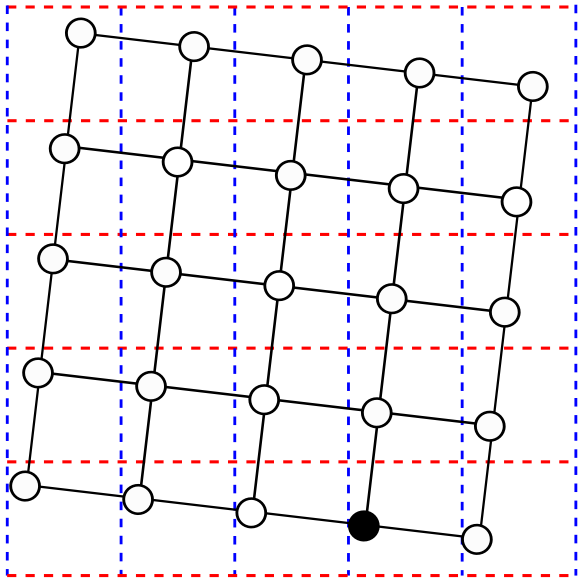


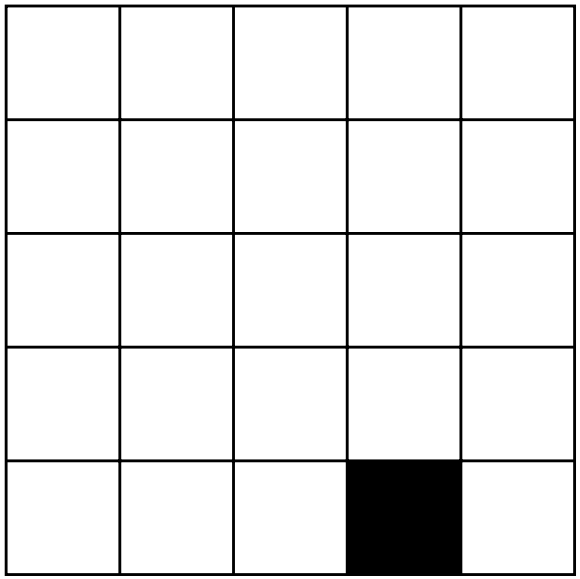


Discrete rigid transformation

Definition

*A **discrete rigid transformation** (DRT) is the set of all the rigid transformations that generate a same image.*

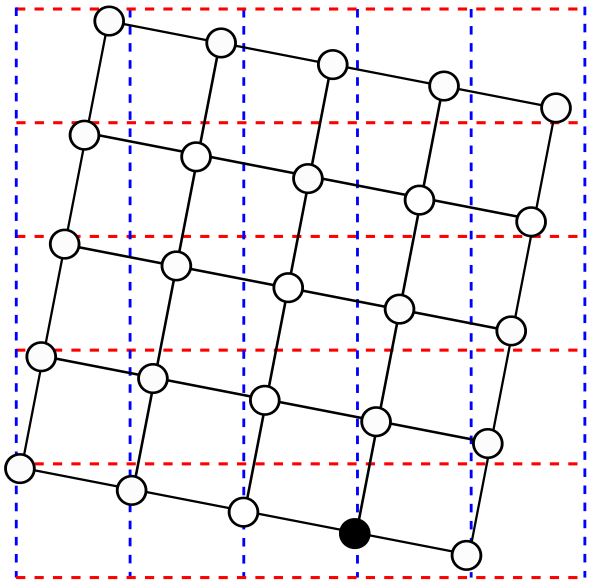


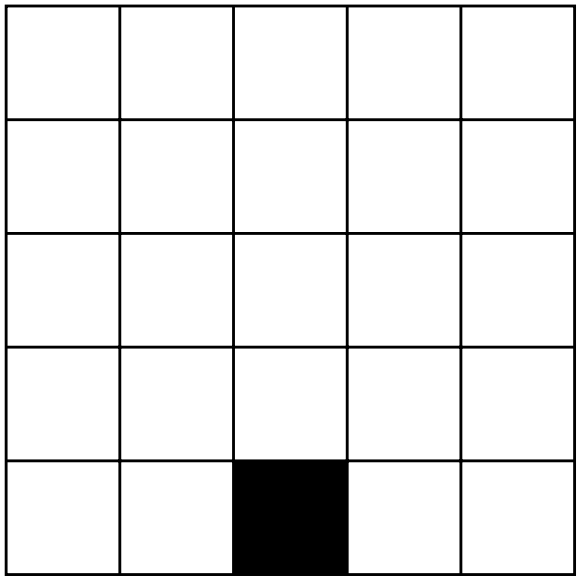


Discrete rigid transformation

Definition

*A **discrete rigid transformation** (DRT) is the set of all the rigid transformations that generate a same image.*

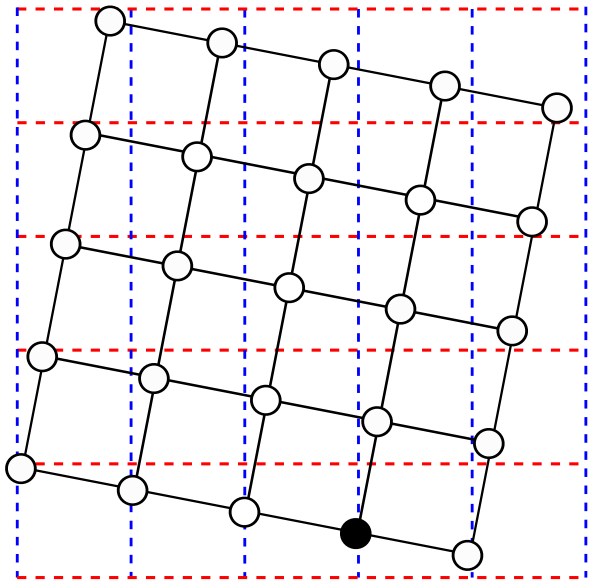


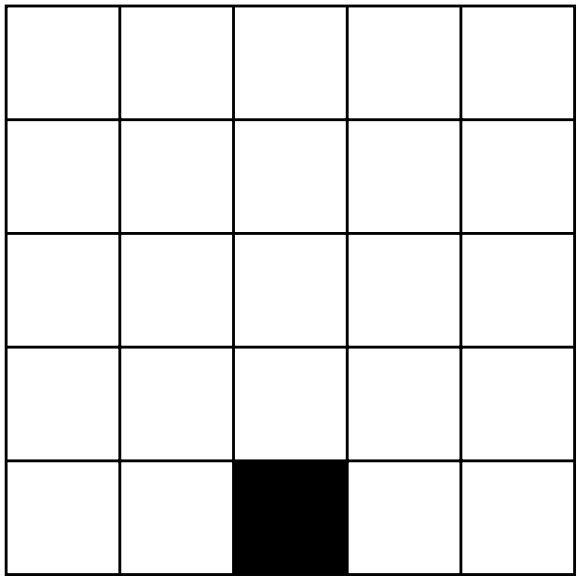


Discrete rigid transformation

Definition

*A **discrete rigid transformation** (DRT) is the set of all the rigid transformations that generate a same image.*



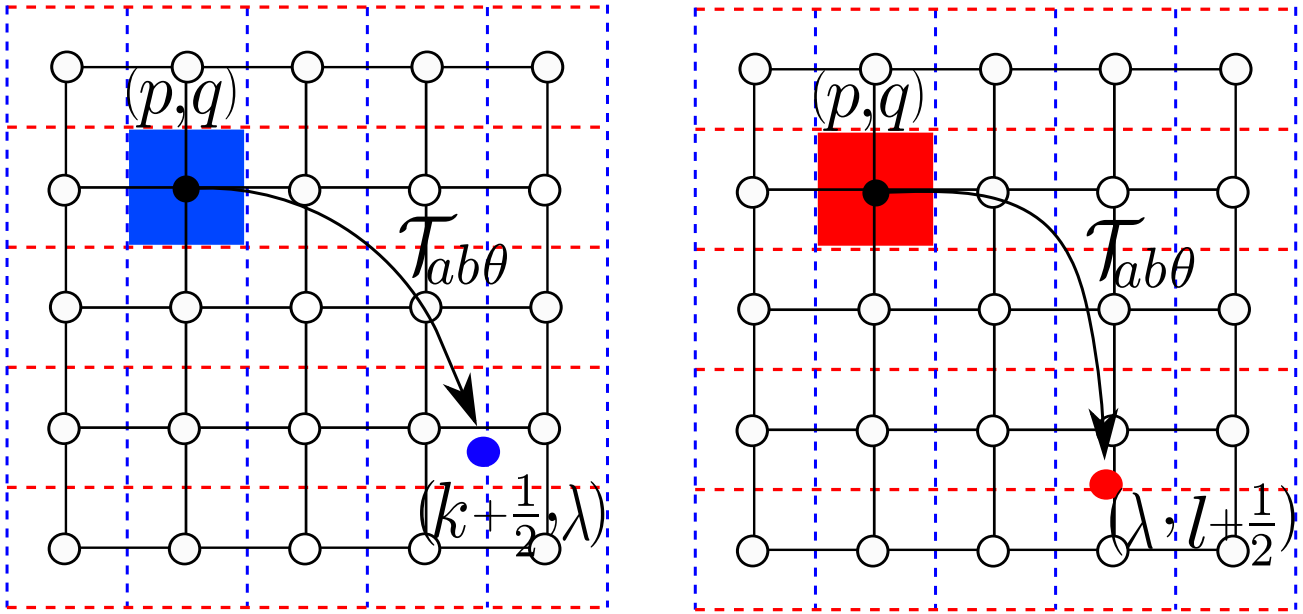


The parameter space (a_1, a_2, θ) is partitioned by the disjoint set of DRTs.

Critical rigid transformations

Definition

*A **critical rigid transformation** moves at least one point of $\mathbb{Z}^2$ to a point on the vertical or horizontal half-grid.*



Tipping surfaces

Definition

*The **tipping surfaces** are the surfaces associated to critical transformations in the parameter space* (a_1, a_2, θ) *:*

$$a_1 = k + \frac{1}{2} + q \sin\theta - p \cos\theta, \qquad \textit{(vertical)}$$

$$a_2 = l + \frac{1}{2} - p \sin\theta - q \cos\theta, \qquad \textit{(horizontal)}$$

for $p, q, k, l \in \mathbb{Z}$.

Tipping surfaces

Definition

*The **tipping surfaces** are the surfaces associated to critical transformations in the parameter space* (a_1, a_2, θ) *:*

$$a_1 = k + \frac{1}{2} + q \sin\theta - p \cos\theta, \qquad \textit{(vertical)}$$

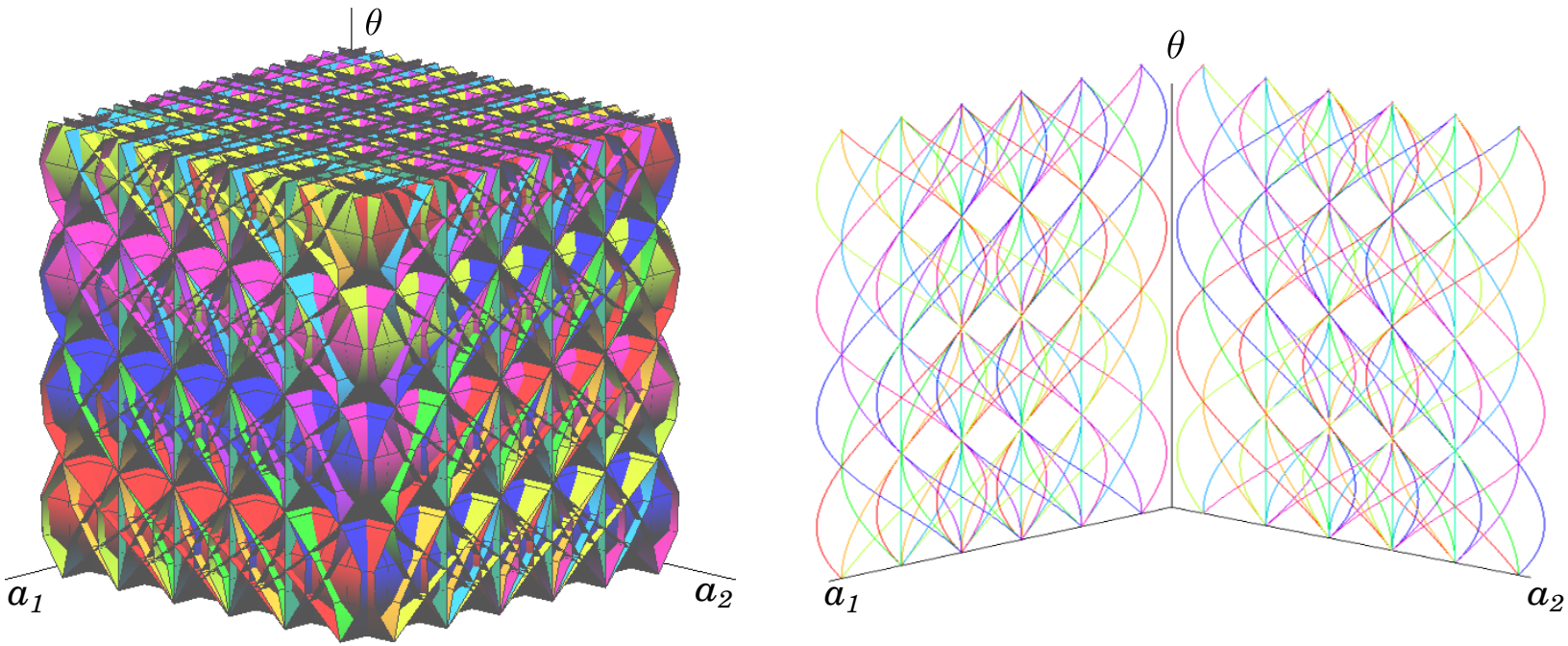
$$a_2 = l + \frac{1}{2} - p \sin\theta - q \cos\theta, \qquad \textit{(horizontal)}$$

for $p, q, k, l \in \mathbb{Z}$.

Each **tipping surface**

- is indexed by a **triplet of integers** (p, q, k) (resp. (p, q, l)),
- indicates that the pixel (p, q) in a transformed image **changes its value** from the one at $(k, *)$ (resp. $(*, l)$) in an original image to the one at $(k + 1, *)$ (resp. $(*, l + 1)$).

Example of tipping surfaces



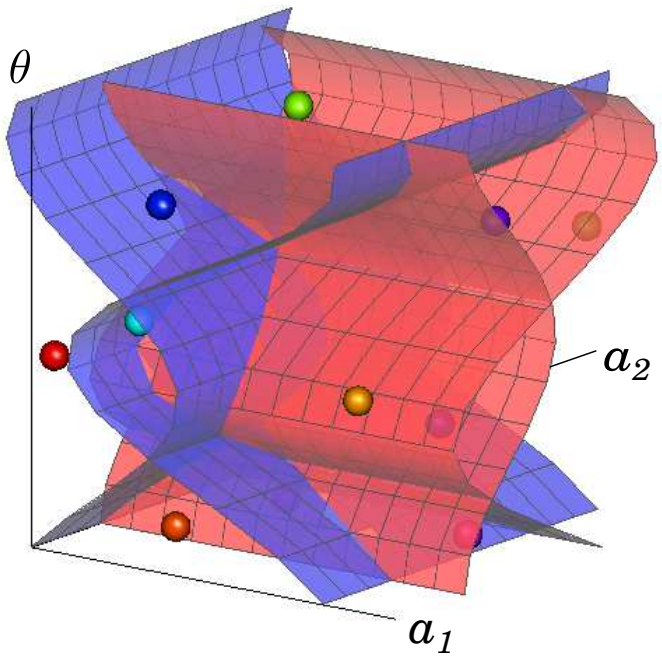
Vertical surfaces Φ_{pqk} and horizontal ones Ψ_{pql} for $p, q \in [0, 2]$ and $k, l \in [0, 3]$.

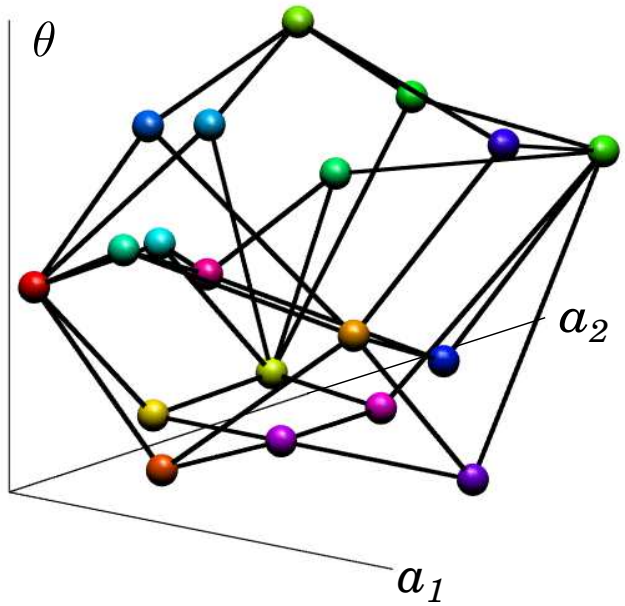
Graph of discrete rigid transformations

Definition

*A **graph of discrete rigid transformations** (DRT graph) is a graph $G = (V, E)$ such that :*

- *each vertex $v \in V$ corresponds to a DRT,*
- *each edge $e \in E$ connects two DRTs sharing a tipping surface.*





Properties of discrete rigid transformation graph

Advantages

- Discrete rigid transformations are **computed in a fully discrete process**.

Properties of discrete rigid transformation graph

Advantages

- Discrete rigid transformations are **computed in a fully discrete process**.
- Their **combinatorial structure** is represented by a **DRT graph** G whose complexity is $O(N^9)$ for images of size $N \times N$.

Properties of discrete rigid transformation graph

Advantages

- Discrete rigid transformations are **computed in a fully discrete process**.
- Their **combinatorial structure** is represented by a **DRT graph** G whose complexity is $O(N^9)$ for images of size $N \times N$.
- G models all the discrete rigid transformations with their **topological information** such that :
  - a vertex corresponds to a different transformed image,
  - an edge corresponds to one pixel change, *i.e. a tipping surface*, (each edge possesses such a pixel transition information).

Properties of discrete rigid transformation graph

Advantages

- Discrete rigid transformations are **computed in a fully discrete process**.
- Their **combinatorial structure** is represented by a **DRT graph** G whose complexity is $O(N^9)$ for images of size $N \times N$.
- G models all the discrete rigid transformations with their **topological information** such that :
  - a vertex corresponds to a different transformed image,
  - an edge corresponds to one pixel change, *i.e. a tipping surface*, (each edge possesses such a pixel transition information).
- It enables to **generate exhaustively and incrementally all the transformed images** in linear time.

Registration as a combinatorial optimisation problem

Problem formulation

Given two images A and B of size $N \times N$, image registration consists of finding a DRT such that

$$v^* = \arg\min_{v \in V} d(A, T_v(B))$$

where T_v is the digital rigid transformation of a DRT v, and d is a given distance between two images.

Registration as a combinatorial optimisation problem

Problem formulation

Given two images A and B of size $N \times N$, image registration consists of finding a DRT such that

$$v^* = \arg\min_{v \in V} d(A, T_v(B))$$

where T_v is the digital rigid transformation of a DRT v, and d is a given distance between two images.

We have a choice for d ; here we use **signed distance**. (Boykov *et al.*, 2006)

Registration as a combinatorial optimisation problem

Problem formulation

Given two images A and B of size $N \times N$, image registration consists of finding a DRT such that

$$v^* = \arg\min_{v \in V} d(A, T_v(B))$$

where T_v is the digital rigid transformation of a DRT v, and d is a given distance between two images.

We have a choice for d ; here we use **signed distance**. (Boykov *et al.*, 2006)

Disadvantage

Exhaustive search on DRT graph costs $O(N^9)$ in complexity.

Registration as a combinatorial optimisation problem

Problem formulation

Given two images A and B of size $N \times N$, image registration consists of finding a DRT such that

$$v^* = \arg\min_{v \in V} d(A, T_v(B))$$

where T_v is the digital rigid transformation of a DRT v, and d is a given distance between two images.

We have a choice for d ; here we use **signed distance**. (Boykov *et al.*, 2006)

Disadvantage

Exhaustive search on DRT graph costs $O(N^9)$ in complexity.

Advantage

A local search on DRT graph can determine a **local optimum**.

Local search on discrete rigid transformation graph

Local search

- **Input** : An initial DRT $v_0 \in V$.
- **Output** : A local optimum $\widehat{v} \in V$.
- **Approach** : Gradient descent : find a better solution in neighbours.

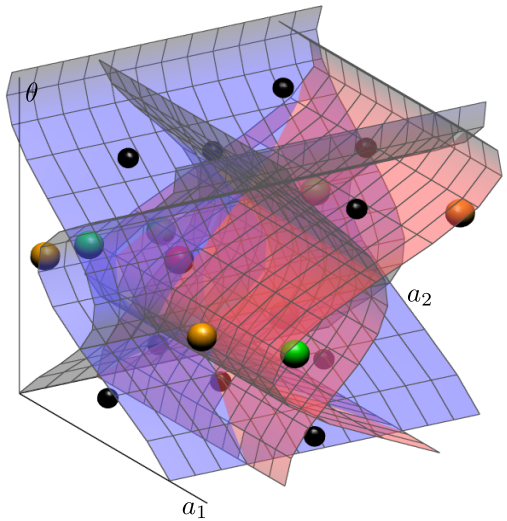
Local search on discrete rigid transformation graph

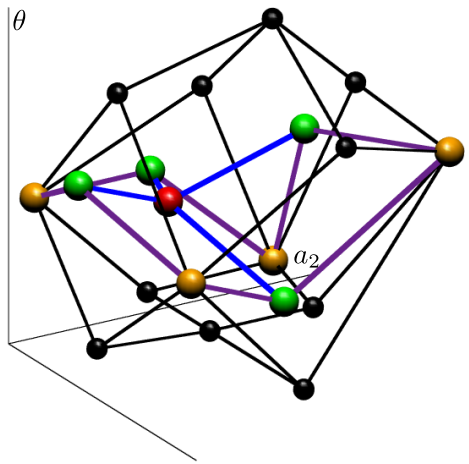
Local search

- **Input** : An initial DRT $v_0 \in V$.
- **Output** : A local optimum $\widehat{v} \in V$.
- **Approach** : Gradient descent : find a better solution in neighbours.

DRT graph provides

- **neighbourhood structure** $N(v)$
- **efficient computation of** d





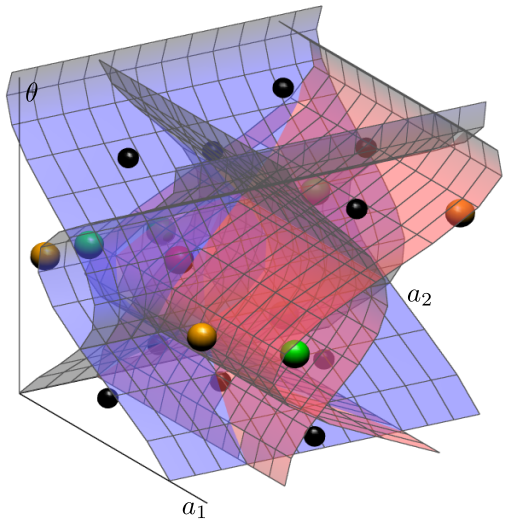
Local search on discrete rigid transformation graph

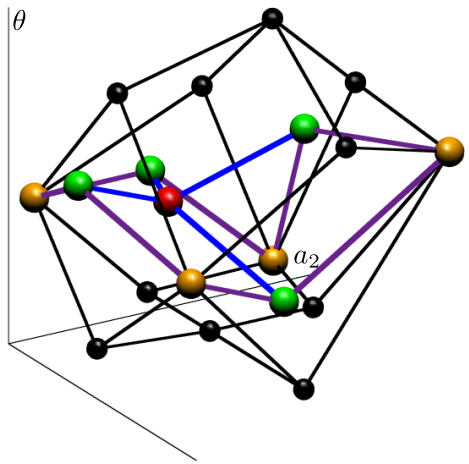
Local search

- **Input** : An initial DRT $v_0 \in V$.
- **Output** : A local optimum $\widehat{v} \in V$.
- **Approach** : Gradient descent : find a better solution in neighbours.

DRT graph provides

- **neighbourhood structure** $N^k(v) = N^{k-1}(v) \cup \bigcup_{u \in N^{k-1}(v)} N(u)$
- **efficient computation of** d



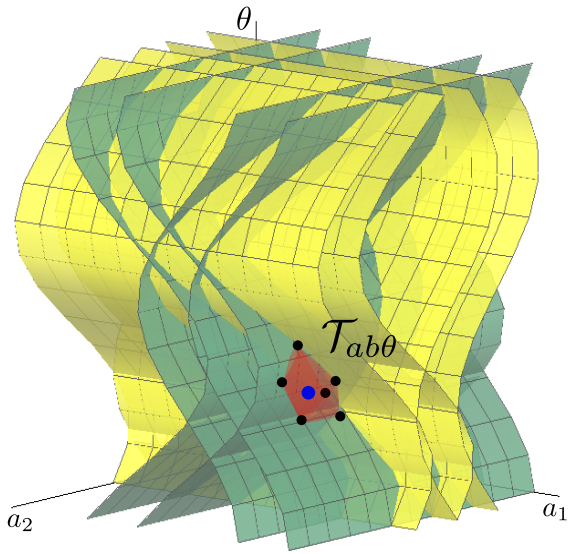


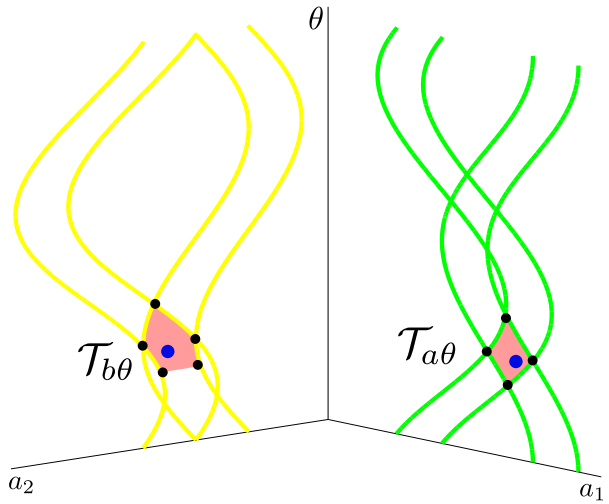
k-neighbourhood construction in a DRT graph

Around a DRT v, we can consider $4N^2$ inequalities :

$$p'_i - \frac{1}{2} < p_i \cos\theta - q_i \sin\theta + a_1 < p'_i + \frac{1}{2}$$
$$q'_i - \frac{1}{2} < p_i \sin\theta + q_i \cos\theta + a_2 < q'_i + \frac{1}{2}$$

due to the pixel correspondence between (p_i, q_i) in the transformed image and (p'_i, q'_i) in the original image.





k-neighbourhood construction in a DRT graph

Around a DRT v, we can consider $4N^2$ inequalities :

$$p'_i - \frac{1}{2} < p_i \cos\theta - q_i \sin\theta + a_1 < p'_i + \frac{1}{2}$$
$$q'_i - \frac{1}{2} < p_i \sin\theta + q_i \cos\theta + a_2 < q'_i + \frac{1}{2}$$

due to the pixel correspondence between (p_i, q_i) in the transformed image and (p'_i, q'_i) in the original image.

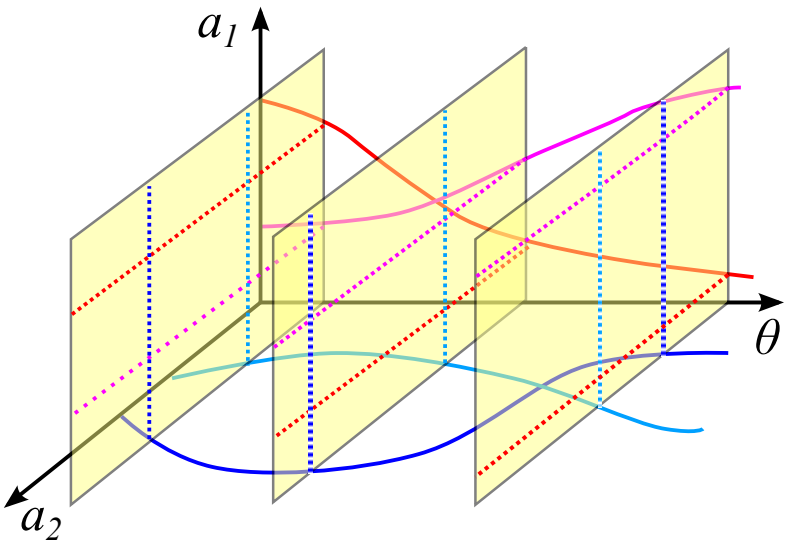
k-neighbourhood construction problem

- **Input** : $4N^2$ tipping surfaces around v
- **Output** : $N^k(v) = N^{k-1}(v) \cup \bigcup_{u \in N^{k-1}(v)} N(u)$

Previous time complexity $O(m^k N^2)$ is improved to $O(mkN^2)$ where m is vertex degree.

Sweeping plane algorithm for DRT sub-graph construction

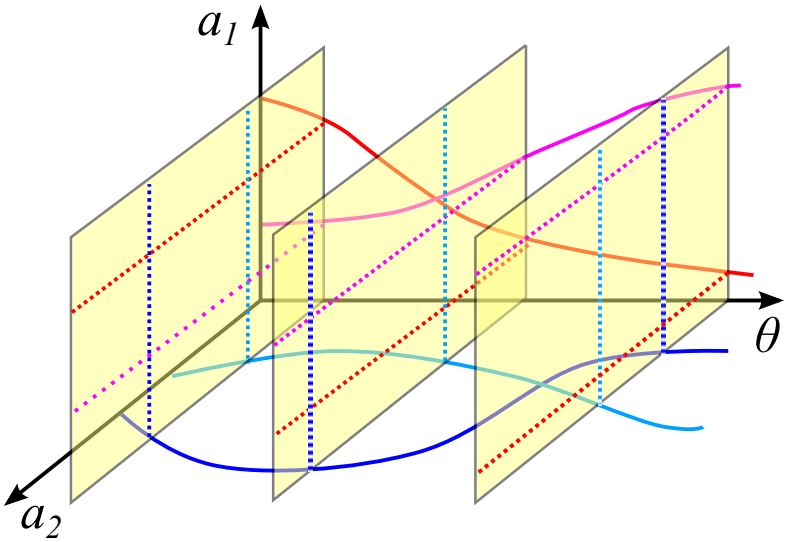
DRT sub-graph construction by sweeping plane algorithm, with 2 vertical and 2 horizontal tipping surfaces.



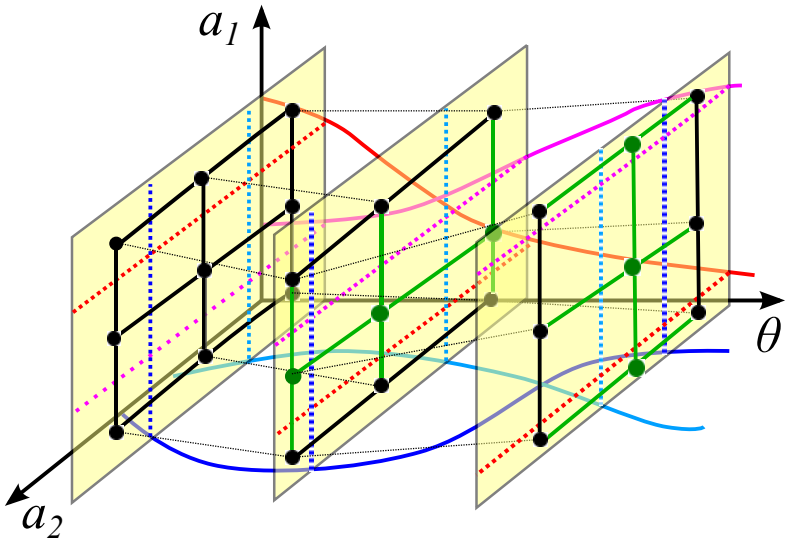
(a) 3×3 cells generated by tipping surfaces in each plane

Sweeping plane algorithm for DRT sub-graph construction

DRT sub-graph construction by sweeping plane algorithm, with 2 vertical and 2 horizontal tipping surfaces.



(a) 3×3 cells generated by tipping surfaces in each plane



(b) Associated DRT graph in each plane

k-neighbourhood construction algorithm

Algorithm (k-neighbourhood construction (in left-hand side of θ-axis))

- ***Input*** *: a DRT v and its initial position θ_v.*
- ***Output*** *: The k-neighbours of v*

1. *Set $\theta_{prev} = \theta_v$.*

k-neighbourhood construction algorithm

Algorithm (k-neighbourhood construction (in left-hand side of θ-axis))

- ***Input*** *: a DRT v and its initial position θ_v.*
- ***Output*** *: The k-neighbours of v*

1. *Set $\theta_{prev} = \theta_v$.*
2. *Find $4(k+1)$ closest tipping surfaces around v.*

k-neighbourhood construction algorithm

Algorithm (k-neighbourhood construction (in left-hand side of θ-axis))

- ***Input*** *: a DRT v and its initial position θ_v.*
- ***Output*** *: The k-neighbours of v*

1. *Set $\theta_{prev} = \theta_v$.*
2. *Find $4(k+1)$ closest tipping surfaces around v.*
3. *Sort and put them in an ordered set S.*

k-neighbourhood construction algorithm

Algorithm (k-neighbourhood construction (in left-hand side of θ-axis))

- ***Input*** *: a DRT v and its initial position θ_v.*
- ***Output*** *: The k-neighbours of v*

1. *Set $\theta_{prev} = \theta_v$.*
2. *Find $4(k+1)$ closest tipping surfaces around v.*
3. *Sort and put them in an ordered set S.*
4. *Find the interval $[\theta_{prev}, \theta_{next}]$ where S is not changed.*

k-neighbourhood construction algorithm

Algorithm (k-neighbourhood construction (in left-hand side of θ-axis))

- ***Input*** *: a DRT v and its initial position θ_v.*
- ***Output*** *: The k-neighbours of v*

1. *Set $\theta_{prev} = \theta_v$.*
2. *Find $4(k+1)$ closest tipping surfaces around v.*
3. *Sort and put them in an ordered set S.*
4. *Find the interval $[\theta_{prev}, \theta_{next}]$ where S is not changed.*
5. *Sweep the plane from θ_{prev} to θ_{next}.*

k-neighbourhood construction algorithm

Algorithm (k-neighbourhood construction (in left-hand side of θ-axis))

- ***Input*** *: a DRT v and its initial position θ_v.*
- ***Output*** *: The k-neighbours of v*

1. *Set $\theta_{prev} = \theta_v$.*
2. *Find $4(k+1)$ closest tipping surfaces around v.*
3. *Sort and put them in an ordered set S.*
4. *Find the interval $[\theta_{prev}, \theta_{next}]$ where S is not changed.*
5. *Sweep the plane from θ_{prev} to θ_{next}.*
6. *If the neighbourhood depth of each generated vertex is not more than k, update S and go back to Step 4.*

k-neighbourhood construction algorithm

Algorithm (k-neighbourhood construction (in left-hand side of θ-axis))

- ***Input*** *: a DRT v and its initial position θ_v.*
- ***Output*** *: The k-neighbours of v*

1. *Set $\theta_{prev} = \theta_v$.*
2. *Find $4(k+1)$ closest tipping surfaces around v.* $O(N^2)$
3. *Sort and put them in an ordered set S.* $O(k \log k)$
4. *Find the interval $[\theta_{prev}, \theta_{next}]$ where S is not changed.* $O(N^2)$
5. *Sweep the plane from θ_{prev} to θ_{next}.* $O(k^2)$
6. *If the neighbourhood depth of each generated vertex is not more than k, update S and go back to Step 4.* $O(1)$

k-neighbourhood construction algorithm

Algorithm (k-neighbourhood construction (in left-hand side of θ-axis))

- ***Input*** *: a DRT v and its initial position θ_v.*
- ***Output*** *: The k-neighbours of v*

1. *Set $\theta_{prev} = \theta_v$.*
2. *Find $4(k+1)$ closest tipping surfaces around v.* $O(N^2)$
3. *Sort and put them in an ordered set S.* $O(k \log k)$
4. *Find the interval $[\theta_{prev}, \theta_{next}]$ where S is not changed.* $O(N^2)$
5. *Sweep the plane from θ_{prev} to θ_{next}.* $O(k^2)$
6. *If the neighbourhood depth of each generated vertex is not more than k, update S and go back to Step 4.* $O(1)$

Estimated number of iterations (Steps 4-6) : $m(2k+1)$

k-neighbourhood construction algorithm

Algorithm (k-neighbourhood construction (in left-hand side of θ-axis))

- ***Input*** *: a DRT v and its initial position θ_v.*
- ***Output*** *: The k-neighbours of v*

1. *Set $\theta_{prev} = \theta_v$.*
2. *Find $4(k+1)$ closest tipping surfaces around v.* $O(N^2)$
3. *Sort and put them in an ordered set S.* $O(k \log k)$
4. *Find the interval $[\theta_{prev}, \theta_{next}]$ where S is not changed.* $O(N^2)$
5. *Sweep the plane from θ_{prev} to θ_{next}.* $O(k^2)$
6. *If the neighbourhood depth of each generated vertex is not more than k, update S and go back to Step 4.* $O(1)$

Estimated number of iterations (Steps 4-6) : $m(2k+1)$
Total complexity : $O(mkN^2)$

Average vertex degree of DRT graphs

Let v, e and f be the numbers of vertices, edges and cells of the 2D DRT graph, which is the projected DRT graph onto the (a_i, θ) plane, $i = 1, 2$.

Average vertex degree of DRT graphs

Let v, e and f be the numbers of vertices, edges and cells of the 2D DRT graph, which is the projected DRT graph onto the (a_i, θ) plane, $i = 1, 2$.

From

- the Euler formula : $v - e + f = 2$,

Average vertex degree of DRT graphs

Let v, e and f be the numbers of vertices, edges and cells of the 2D DRT graph, which is the projected DRT graph onto the (a_i, θ) plane, $i = 1, 2$.

From

- the Euler formula : $v - e + f = 2$,
- the DRT graph property : $4f \leq 2e$,

Average vertex degree of DRT graphs

Let v, e and f be the numbers of vertices, edges and cells of the 2D DRT graph, which is the projected DRT graph onto the (a_i, θ) plane, $i = 1, 2$.

From

- the Euler formula : $v - e + f = 2$,
- the DRT graph property : $4f \leq 2e$,

we have the average degree

$$\frac{2e}{v} \leq 4 - \frac{8}{v} < 4$$

as $v \gg 8$ in the 2D DRT graph, and thus

Average vertex degree of DRT graphs

Let v, e and f be the numbers of vertices, edges and cells of the 2D DRT graph, which is the projected DRT graph onto the (a_i, θ) plane, $i = 1, 2$.

From

- the Euler formula : $v - e + f = 2$,
- the DRT graph property : $4f \leq 2e$,

we have the average degree

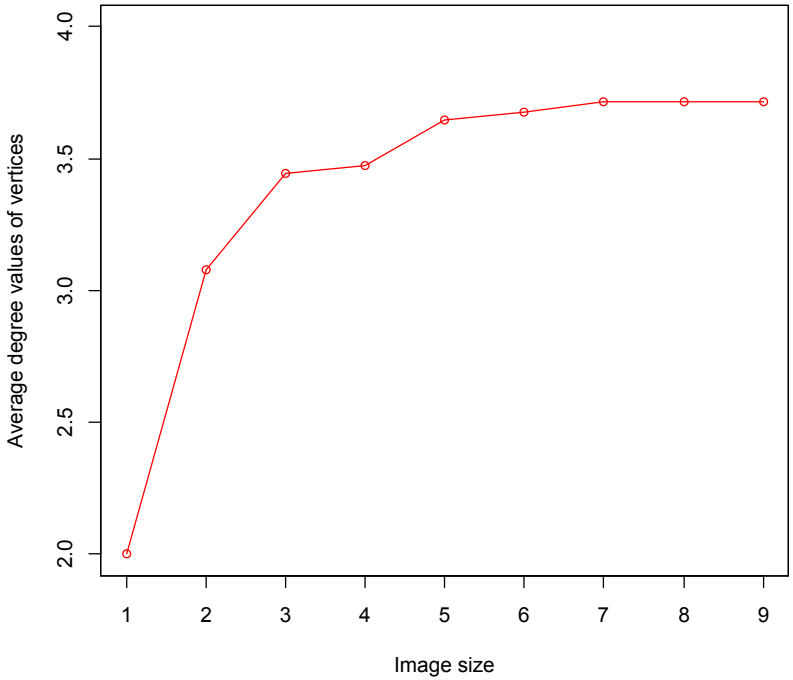
$$\frac{2e}{v} \leq 4 - \frac{8}{v} < 4$$

as $v \gg 8$ in the 2D DRT graph, and thus

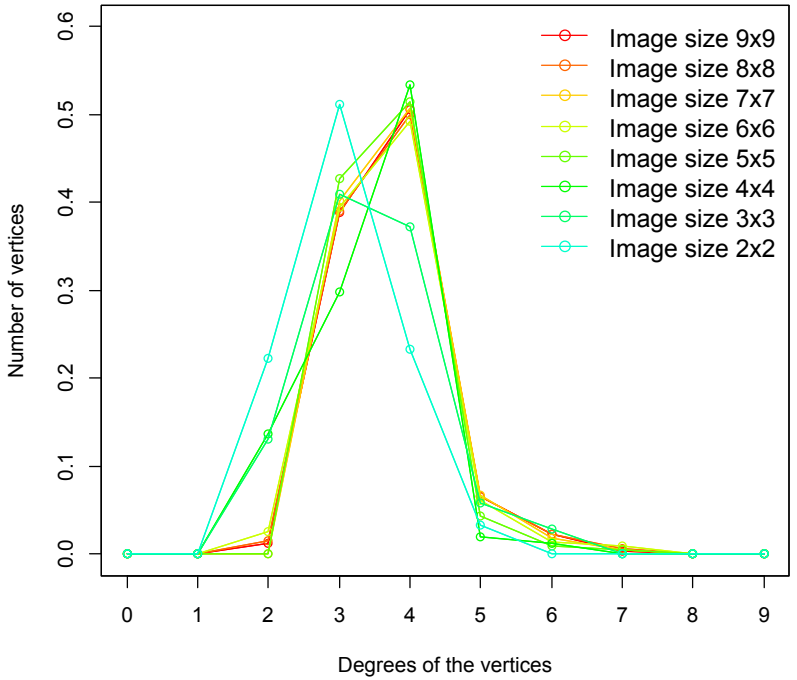
Property

The average vertex degree of DRT graphs is lower than 8.

Average vertex degrees of 2D DRT graphs in practice

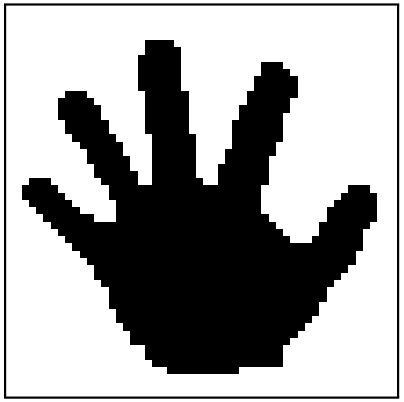


(a) Average vertex degree in a 2D DRT graph

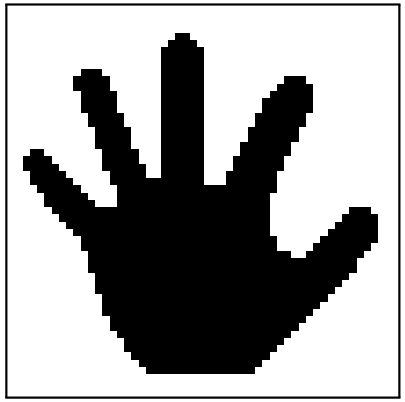


(b) Normalized vertex degree distribution in a 2D DRT graph

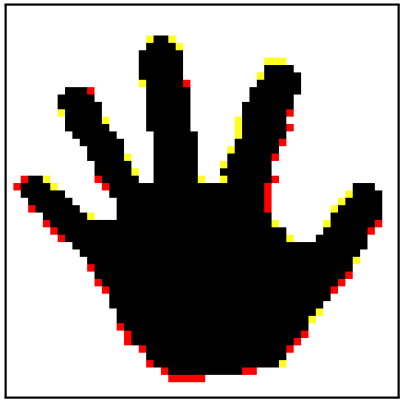
Experiment on binary images



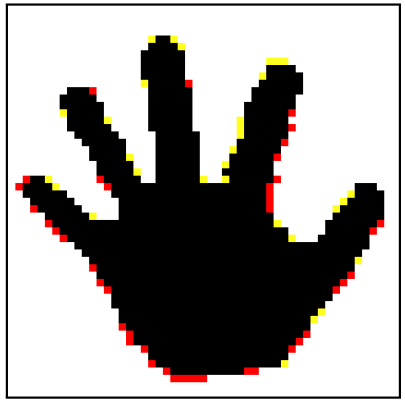
(a) reference image



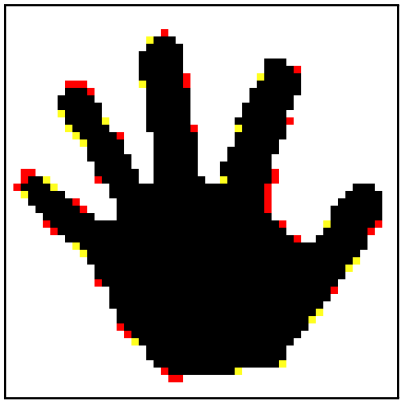
(b) target image



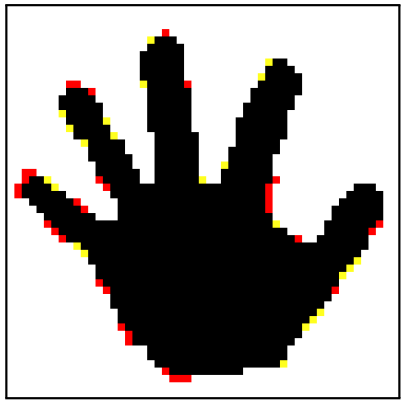
(c) initial solution



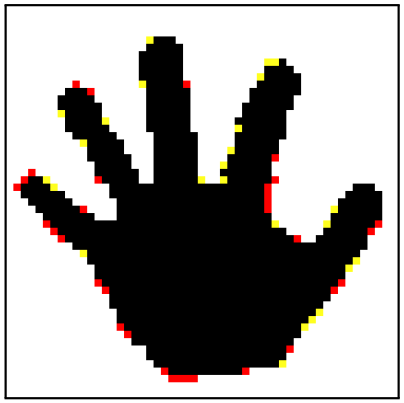
(d) solution : $k = 1$



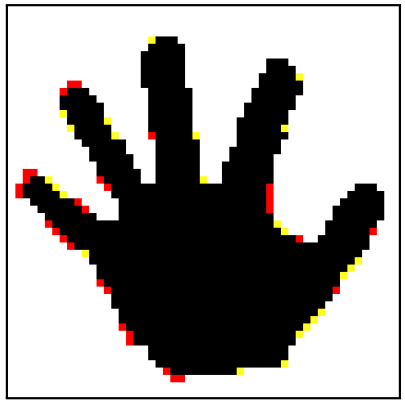
(e) $k = 3$



(f) $k = 5$

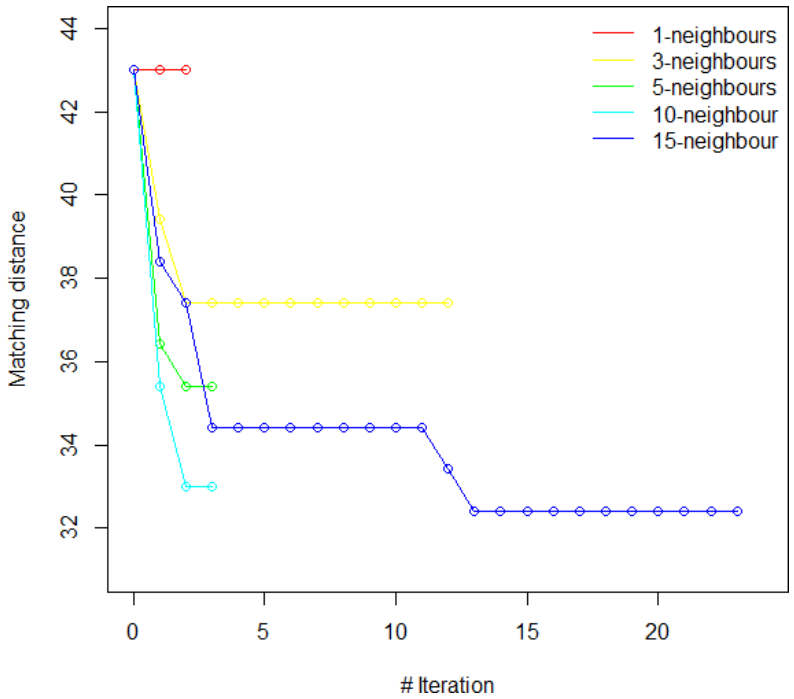


(g) $k = 10$

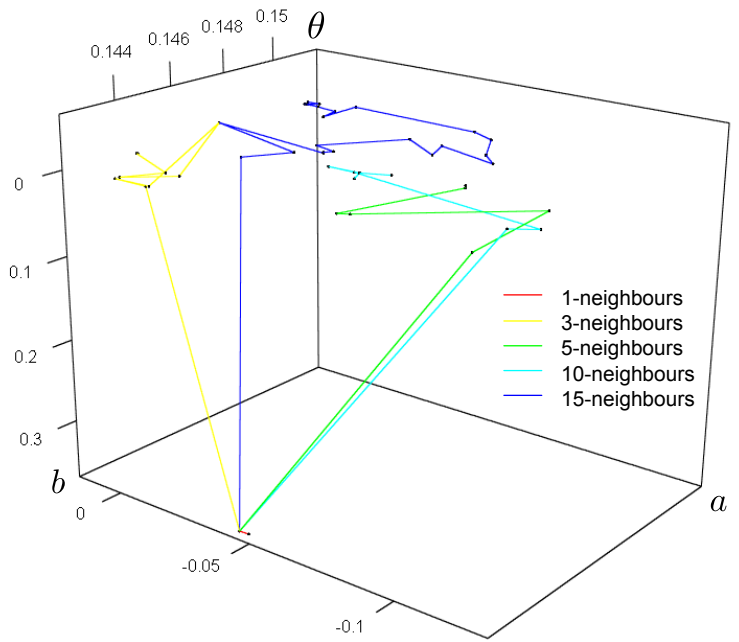


(h) $k = 15$

Experiment on binary images : distance evolutions

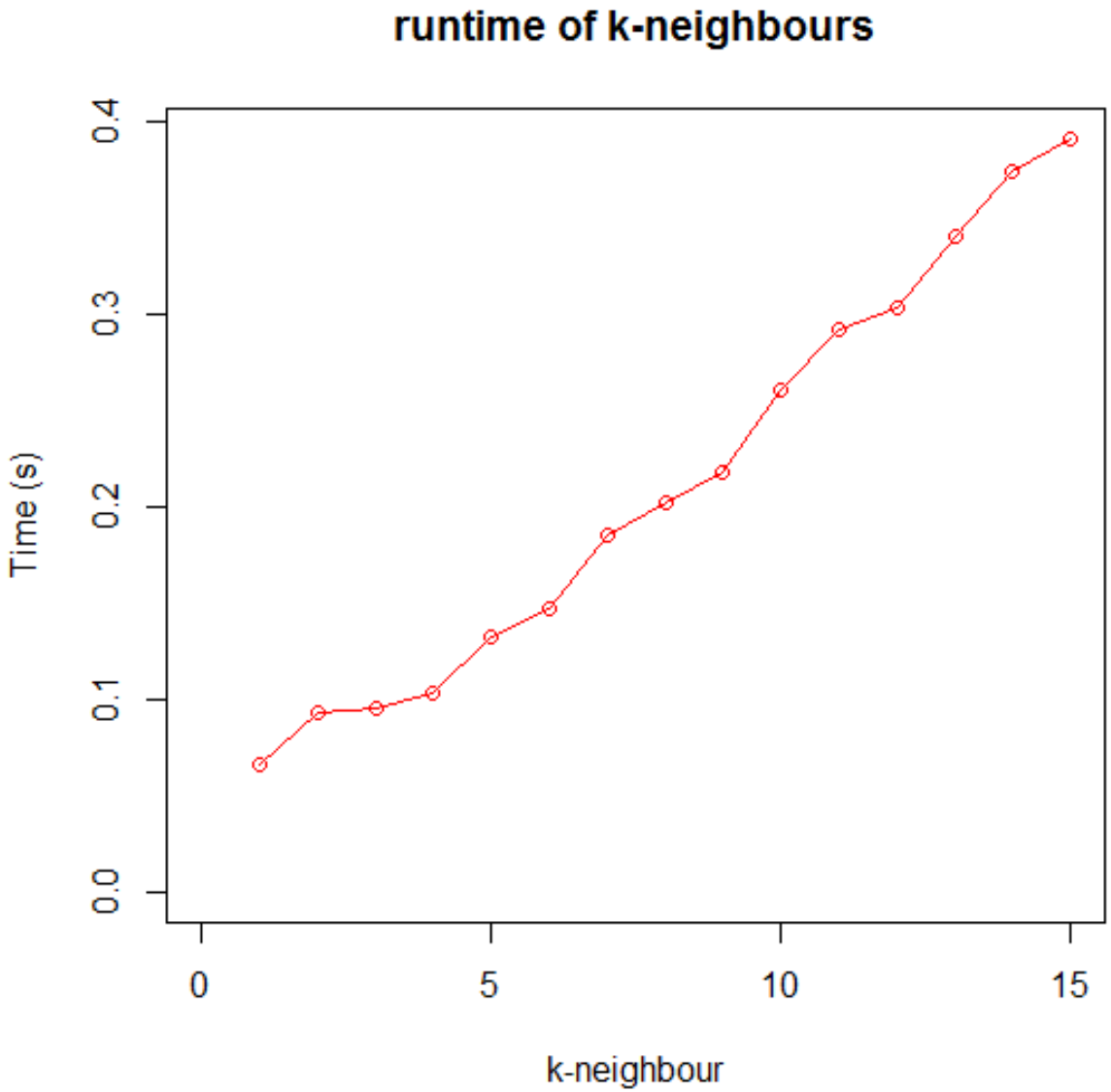


(a) Distance evolutions



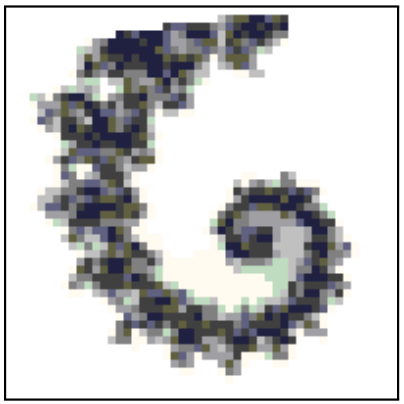
(b) Transformation sequences

Experiment on binary images : runtime complexity

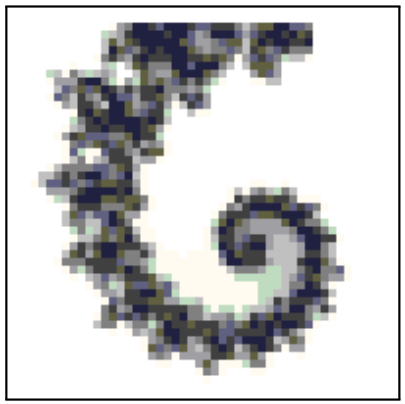


Experiment on binary images : image sequences for $k = 15$

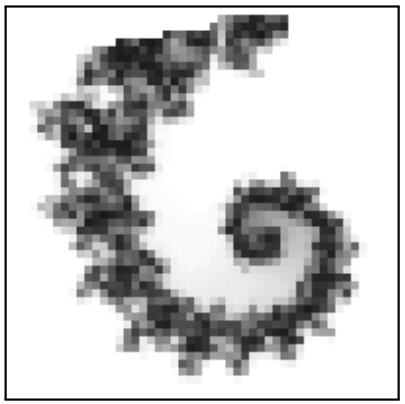
Experiment on gray images



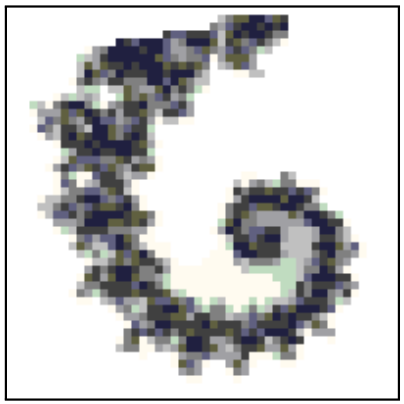
(a) reference image



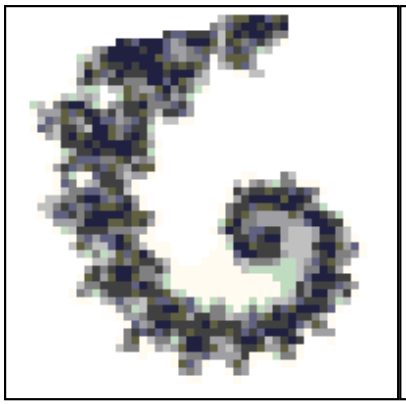
(b) target image



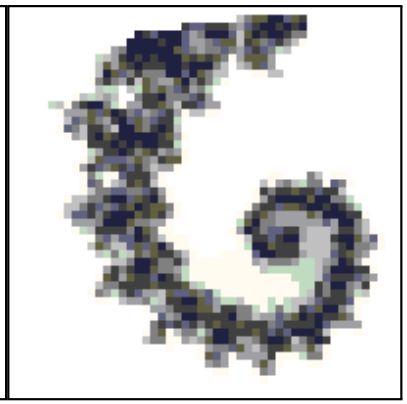
(c) initial solution



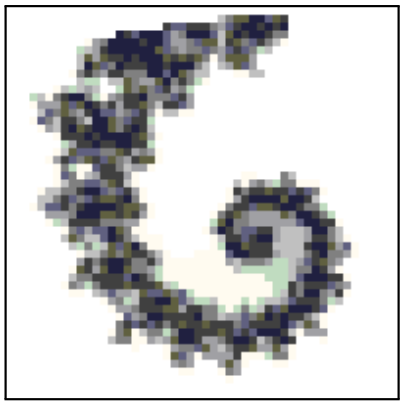
(d) solution : $k = 1$



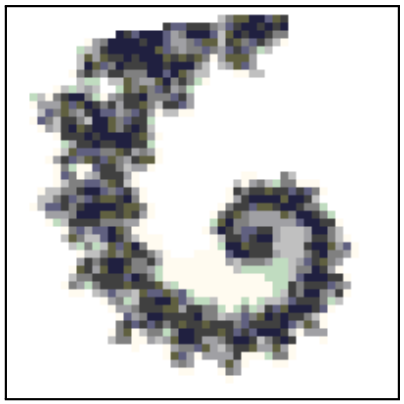
(e) $k = 3$



(f) $k = 5$

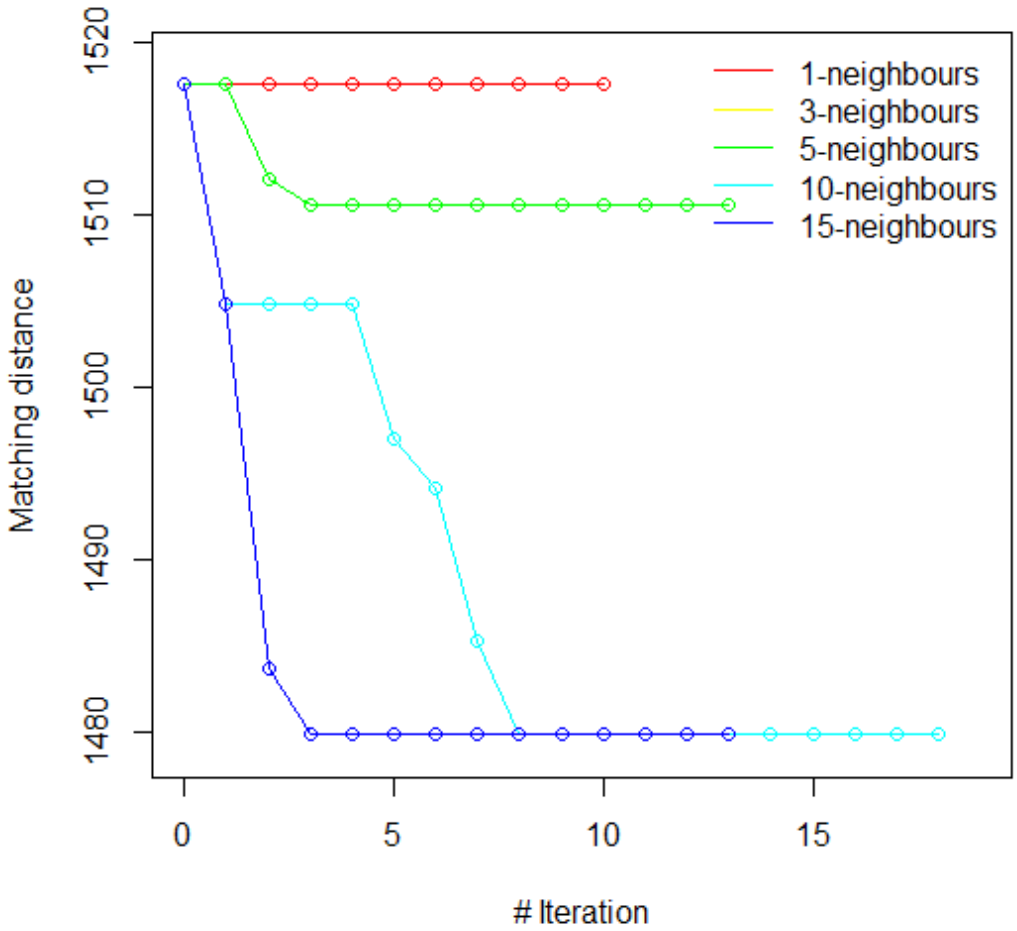


(g) $k = 10$



(h) $k = 15$

Experiment on gray images : distance evolutions



Conclusion and perspectives

Conclusion

- A purely discrete framework for 2D image registration under rigid transformation was proposed, based on DRT graph.

Conclusion and perspectives

Conclusion

- A purely discrete framework for 2D image registration under rigid transformation was proposed, based on DRT graph.
- A gradient descent procedure constructs the exact k-neighbourhood at each step with a $O(mkN^2)$ time complexity.

Conclusion and perspectives

Conclusion

- A purely discrete framework for 2D image registration under rigid transformation was proposed, based on DRT graph.
- A gradient descent procedure constructs the exact k-neighbourhood at each step with a $O(mkN^2)$ time complexity.
- The average vertex degree m is less than 8.

Conclusion and perspectives

Conclusion

- A purely discrete framework for 2D image registration under rigid transformation was proposed, based on DRT graph.
- A gradient descent procedure constructs the exact k-neighbourhood at each step with a $O(mkN^2)$ time complexity.
- The average vertex degree m is less than 8.
- An efficient algorithm for computing the k-neighbours for the local search was proposed with linear time complexity.

Conclusion and perspectives

Conclusion

- A purely discrete framework for 2D image registration under rigid transformation was proposed, based on DRT graph.
- A gradient descent procedure constructs the exact k-neighbourhood at each step with a $O(mkN^2)$ time complexity.
- The average vertex degree m is less than 8.
- An efficient algorithm for computing the k-neighbours for the local search was proposed with linear time complexity.

Perspectives

- Combine our proposed method with other combinatorial approaches.

Conclusion and perspectives

Conclusion

- A purely discrete framework for 2D image registration under rigid transformation was proposed, based on DRT graph.
- A gradient descent procedure constructs the exact k-neighbourhood at each step with a $O(mkN^2)$ time complexity.
- The average vertex degree m is less than 8.
- An efficient algorithm for computing the k-neighbours for the local search was proposed with linear time complexity.

Perspectives

- Combine our proposed method with other combinatorial approaches.
- Extend to higher dimensions, and gray-level or labelled images.