A Fast Algorithm for Liver Surgery Planning

Fajie Li, **Xinbo Fu**, **Gisela Klette**, and **Reinhard Klette**

Xiamen (China) and Auckland (New Zealand)

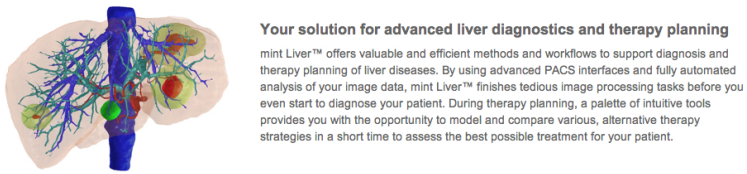
DGCI, Sevilla - March 22, 2013

Slide 1: Introduction

- Liver cancer is the 5th most common malignancy in men and the 8th in women worldwide. In 2000 there were 564,000 new cases.
- **Liver resection** often the cure for primary liver cancer.
- Existing **liver surgery planning** usually requires surgeons' personal expertise to interact during surgery.
- For example, there is branch labelling at some planning stage; this is the most time-consuming step in the planning procedure.

Slide 2: Related work

Mint Liver, 3D image analysis software for liver resection from Germany for hepatic surgeons



- (1) a deformable 2D manifold for resection; 3D interaction to specify and modify this manifold by medical doctors
- (2) a probabilistic atlas
- (3) squared Euclidean distance transform for approximately computing the liver part that should be removed
- (4) calculates the vascular perfusion area, based on direction and diameter of the portal vein branch

Existing algorithms compute only ‘heuristically’ the liver part that should be removed.

Slide 3: The problem

We model the problem in terms of digital geometry and propose an algorithm for computing the diseased part of a liver.

The problem to be solved is as follows:

- Let S_l be the set of cells (i.e., voxels) in the given 3D input image classified to be **liver cells**.
- Set S_h contains all cells classified to be **healthy vein cells**.
- Set S_d contains all the detected **diseased vein cells**.
- We have to specify and then to calculate that part of the liver which is **affected by** diseased vein cells.

Slide 4: Considered subsets

Our **accurate** solution for this problem is defined by the maximum subset $A \subseteq S_l$ such that A is **affected by** the set S_d of diseased cells.

- We compute three subsets $S_{a_{h+d}}$, S_{a_h}, and S_{a_d} such that

$$S_l = S_{a_{h+d}} \cup S_{a_h} \cup S_{a_d}$$

  where $S_{a_{h+d}}$ is affected by both S_h and S_d, S_{a_h} by S_h only, and S_{a_d} by S_d

- $S_{a_{h+d}}$ are boundary cells between healthy liver cells and diseased liver cells.
- Set $S_{a_{h+d}} \cup S_{a_d}$ of liver cells should be removed (our solution).

Slide 5: Definition of "affected by"

Discussion for plane and 2D images only for simplicity.

$d_e(p, q)$ – Euclidean distance between p and q in the plane

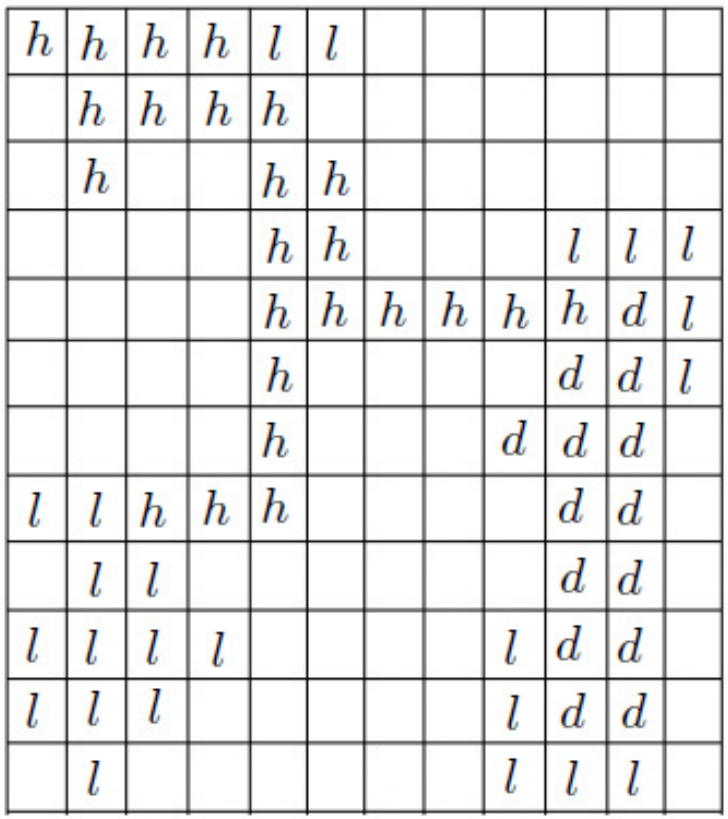
Planar sets A and B: $d_{min}(A, B) = \min\{d_e(p, q) : p \in A \wedge q \in B\}$

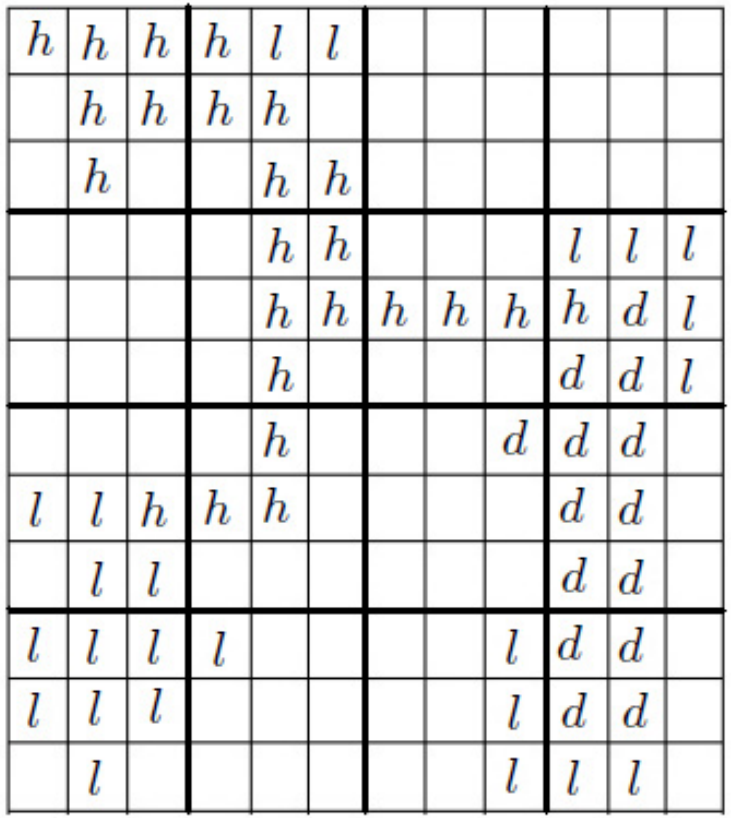
$d_{min}(p, B) = d_{min}(A, B)$ if $A = \{p\}$.

- Set $S \subseteq S_l$ is *only affected* by S_h if for each cell $p_l \in S$, $d_{min}(p_l, S_h) < d_{min}(p_l, S_d)$.
- Analogously: set $S \subseteq S_l$ is only affected by S_d.
- $S \subseteq S_l$ is *affected* by both S_h and S_d if for each cell $p_l \in S$, $d_{min}(p_l, S_h) = d_{min}(p_l, S_d)$.

Slide 6: We use cells and supercells

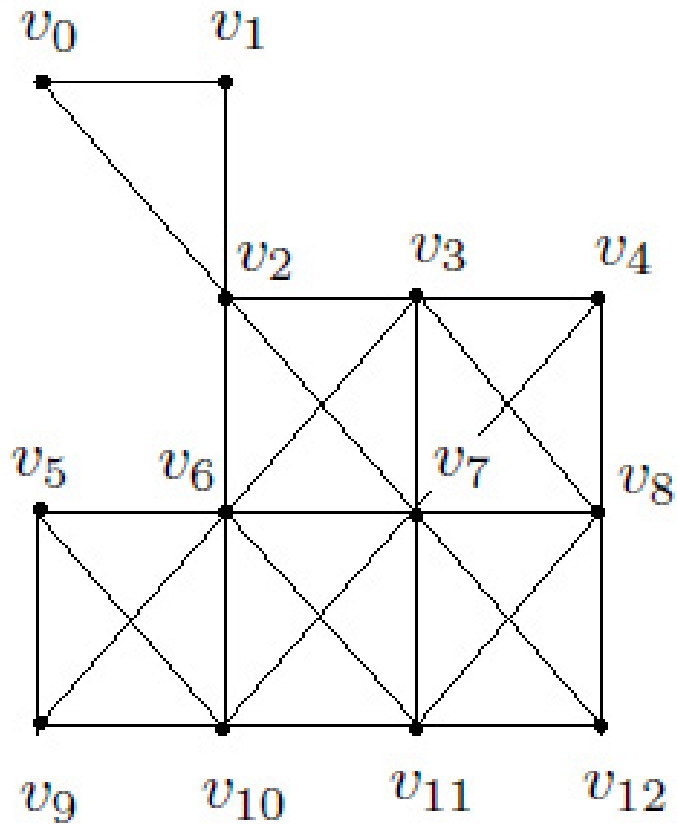
h for cell of type-*h*, *d* for type-*d*, and *l* for type-*l*.





Left: Set S of labelled cells for original grid constant θ_0.
Right: Supercells for grid constant $\theta = \theta_0 \times 3$.

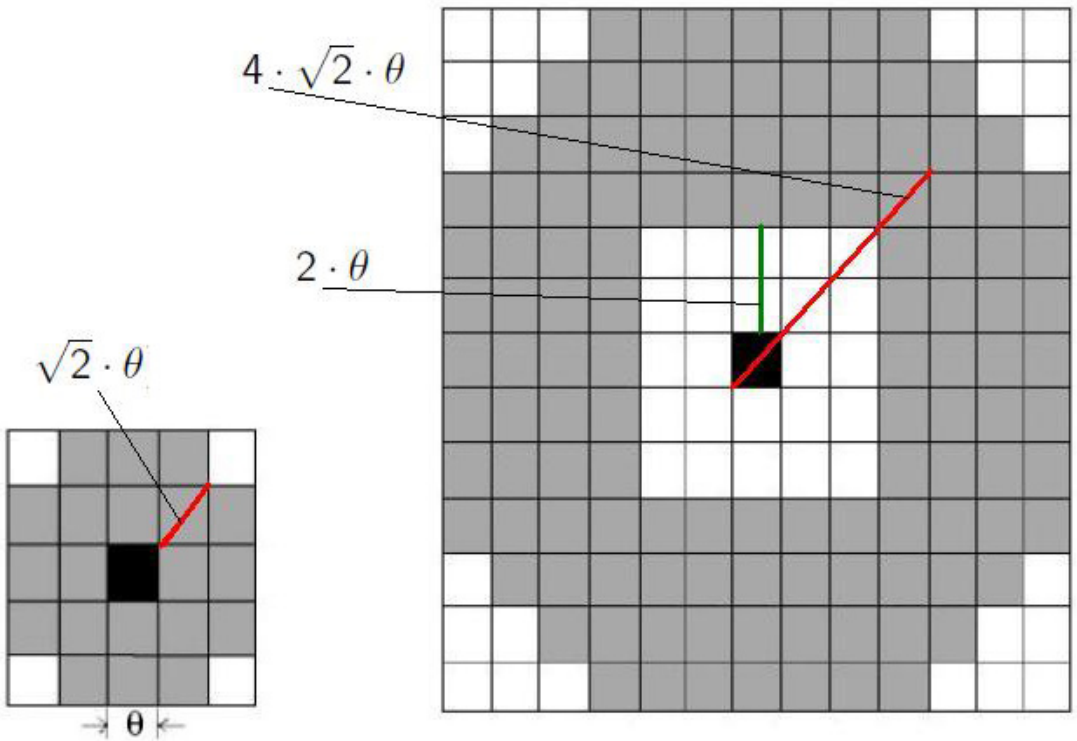
Slide 7: We use θ-graphs for 8-adjacent supercells



8-adjacent occupied θ-supercells (for figure before)
weights are either θ or $\theta\sqrt{2}$

Slide 8: Two types of used adjacency sets for supercells

Just for illustrating that we also use two types of adjacencies.



Left: type-1 adjacency set $N(C, 3, \sqrt{2} \cdot \theta)$ for a type-*sldoh* supercell C.
Right: type-2 adjacency set $N(C, 4, 4, 2)$ for a type-*sl* supercell C.

Slide 9: A brute-force subroutine for subset assignments

The idea of this brute-force routine is simple:

- For each cell $p_l \in S_l$,
- go through S_h for computing $d_{min}(p_l, S_h)$;
- go through S_d for computing $d_{min}(p_l, S_d)$;
- If $d_{min}(p_l, S_h) < d_{min}(p_l, S_d)$, then let $S_{a_h} = S_{a_h} \cup \{p_l\}$;
- else, if $d_{min}(p_l, S_h) > d_{min}(p_l, S_d)$, then let $S_{a_d} = S_{a_d} \cup \{p_l\}$;
- otherwise $S_{a_{d+h}} = S_{a_{d+h}} \cup \{p_l\}$.

Slide 10: An improved version

The idea of its improved version is also simple. Assume that $|S_h| \leq |S_d|$.

- We may not have to go through S_d for computing $d_{min}(p_l, S_d)$:
- If there exists a cell p_d such that $d_e(p_l, p_d) < d_{min}(p_l, S_h)$ then let $S_{a_d} = S_{a_d} \cup \{p_l\}$,
- and we break then both this for-loop and the outer for-loop,
- and test the next cell after p_l in S_l.

Slide 11: The decomposition algorithm

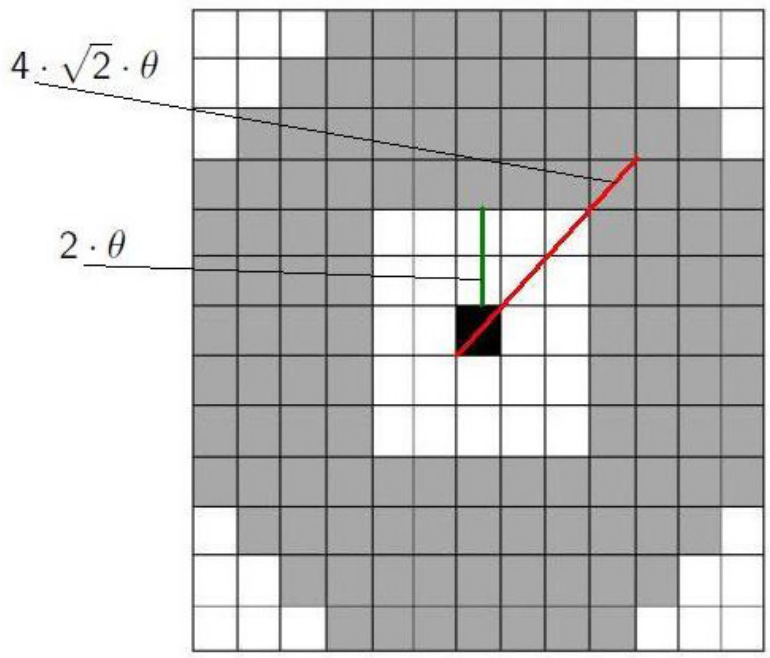
Main ideas:

- decompose the liver into some supercells
- remove unnecessary supercells (that are "too far" from the current supercell)
- reuse the improved version of the above brute-force routine

Main steps:

1. For each supercell C, if C is of type-*sldoh* then compute type-1 adjacency set $N(C, 3, \sqrt{2} \cdot \theta)$
2. The candidate sets are reduced from S_d and S_h to $S_d \cap N(C, 3, \sqrt{2} \cdot \theta)$ and $S_h \cap N(C, 3, \sqrt{2} \cdot \theta)$.

Slide 12: Algorithm continued

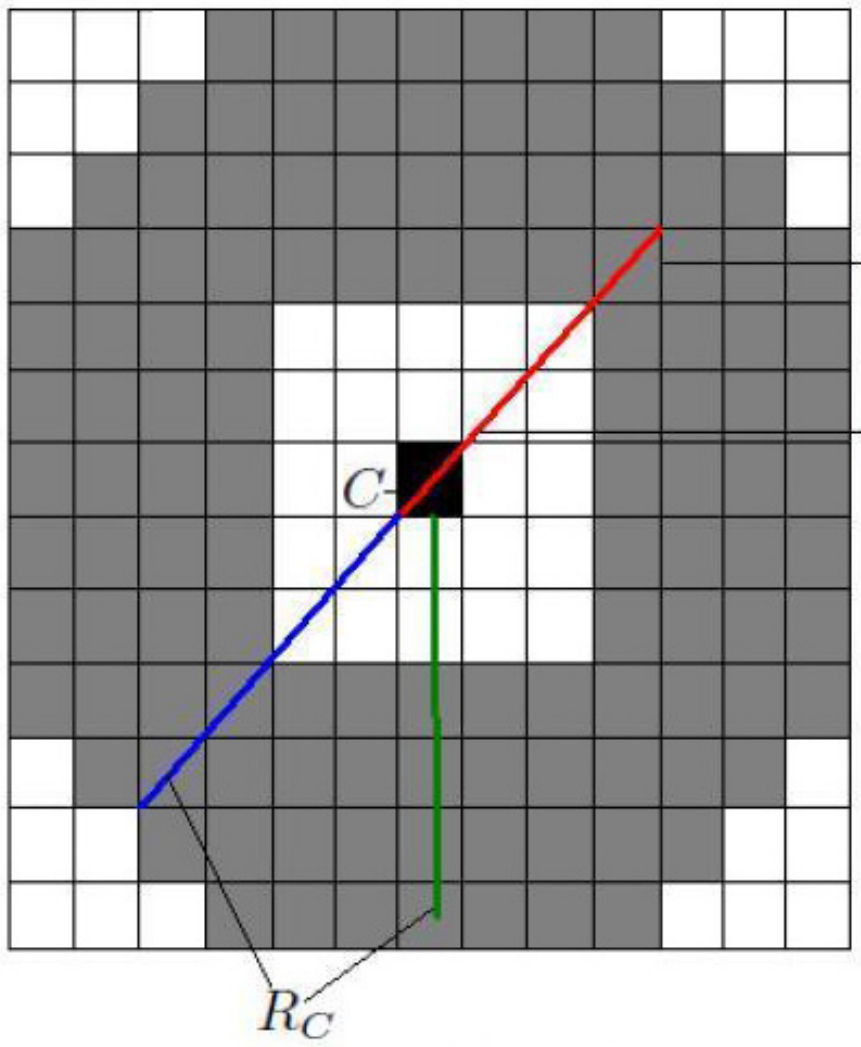


$N(C, 4, x, y)$, where $x = 4$, $y = 2$

3 For each supercell C, if C is of type-*sl* then compute type 2 adjacency set $N(C, 4, x, y)$, where parameters x and y depend on the supercell C.

4 The candidate sets are reduced from S_h and S_d to $S_h \cap N(C, 4, x, y)$ and $S_d \cap N(C, 4, x, y)$, respectively.

Slide 13: Correctness of the algorithm.



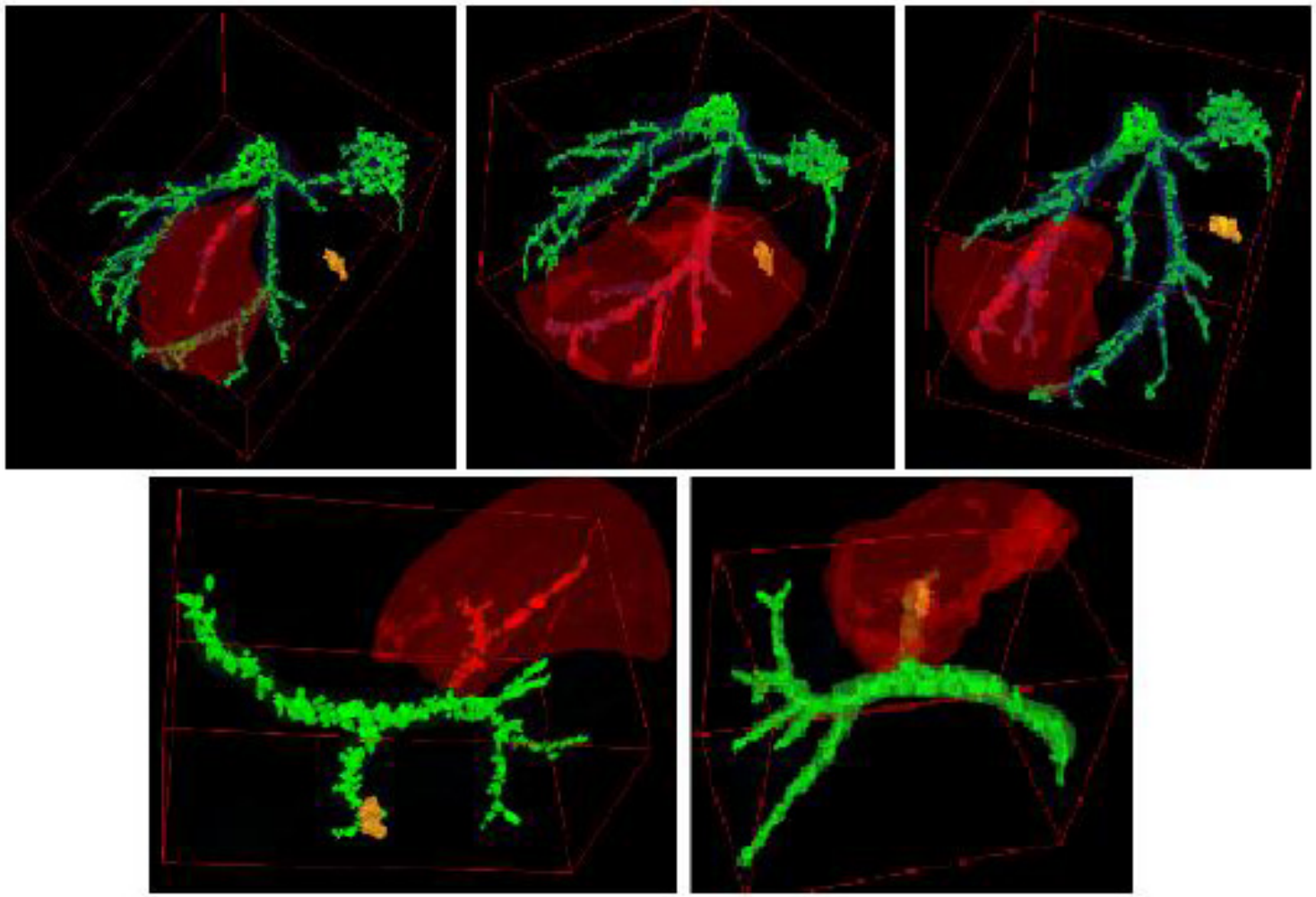
$$N_C = N(C, 4, 4, 2)$$

Slide 14: Time Complexity

The runtime of algorithm is in $\mathcal{O}(m^i \times n \times n_{ave})$, where m satisfies $\theta = m \times \theta_0$, n is the number of supercells, n_{ave} is the average value of all supercells in type 2 adjacency sets $N(C, 4, x, y)$'s, and $i = 2$ for the 2D case, and $i = 3$ for the 3D case.

- Exact Euclidean Distance Transform takes $\mathcal{O}(m^i \times n)$ operations.
- The decomposition algorithm may be slower than the exact Euclidean Distance Transform depending on the value of n_{ave} that depends on the distribution of input type-j cells, for $j = d, h, l$.
- Some literature reports about an average labelling time of about 17 minutes, depending on the data set.
- Our algorithm takes about 30 seconds on the data set $324 \times 243 \times 129$ (PC: 2.4GHz CPU, 2.0Gb RAM)

Slide 15: Examples



Slide 16: Concluding remarks and future work

Concluding remarks:

- propose a time-efficient algorithm for separating liver cells
- In contrast to existing methods, our algorithm is not only independent of a surgeons' personal interactive manipulations,
- but also outputs an exact solution.
- Exact Euclidean Distance Transform may not work correctly if the input liver is non-convex.

Future work:
Generalize the algorithm for non-convex liver inputs.