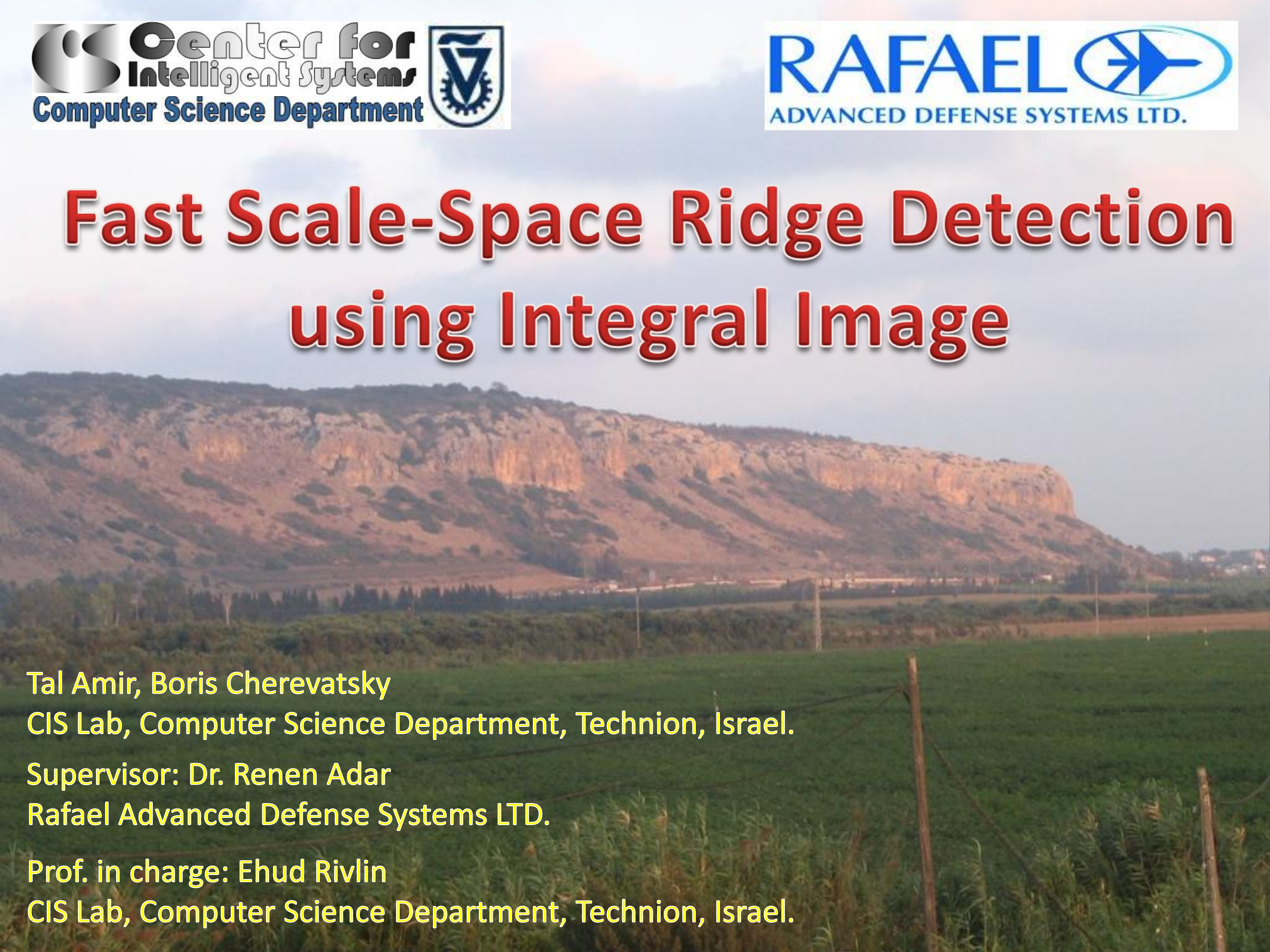


Center for Intelligent Systems
Computer Science Department

RAFAEL
ADVANCED DEFENSE SYSTEMS LTD.

Fast Scale-Space Ridge Detection using Integral Image

Tal Amir, Boris Cherevatsky
CIS Lab, Computer Science Department, Technion, Israel.

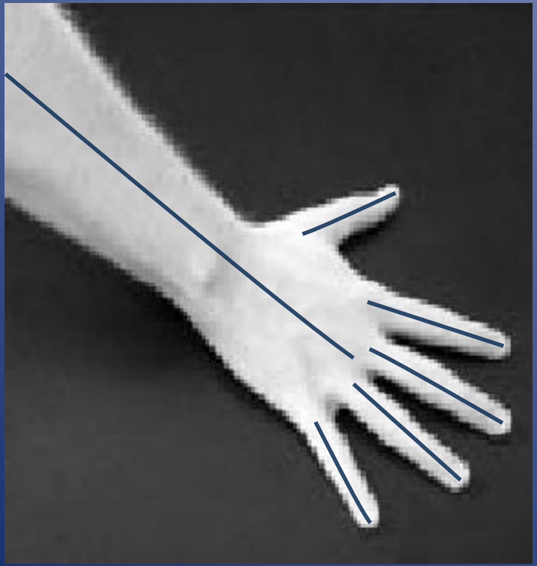
Supervisor: Dr. Renen Adar
Rafael Advanced Defense Systems LTD.

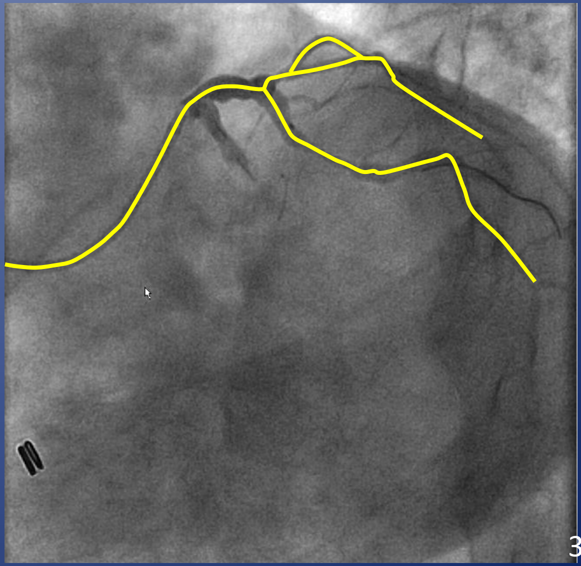
Prof. in charge: Ehud Rivlin
CIS Lab, Computer Science Department, Technion, Israel.

Background

- We would like to detect roads in aerial photos by using ridge detection.
- Due to different road widths, we would like to detect ridges in different scales.
- In [1] T. Lindeberg (1996) proposed a scale-space ridge detection with automatic scale selection by using Gaussian derivatives.
- We propose a faster approach that approximates Lindeberg's solution, plus some additions.

Our goal



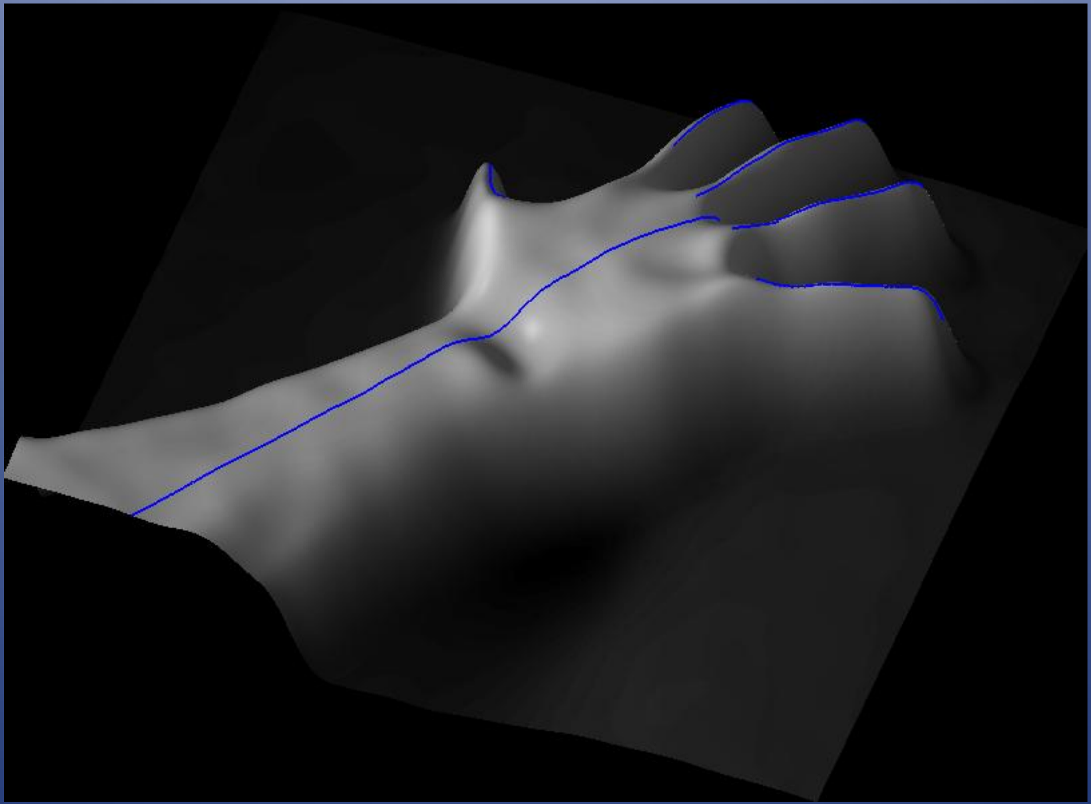


What is a ridge?

Intuitively, a ridge is a long region which is brighter than its external surrounding.

What is a ridge?

If we look at the image as a surface, we aim to find its watersheds:



Single-scale ridge

Definition of a single-scale ridge:

Let $I(x, y): \mathbb{R}^2 \to \mathbb{R}$ be an image.

Let $\lambda_{\max}$ be the principal eigenvalue of the Hessian matrix of I at point (x, y) and let $v_{\max}$ be its corresponding eigenvector.

A point (x, y) is a *ridge point* if the following conditions are satisfied:

1. $\left\langle \nabla I(x, y), v_{\max} \right\rangle = 0$

2. $\lambda_{\max} < 0$ (for bright ridges)

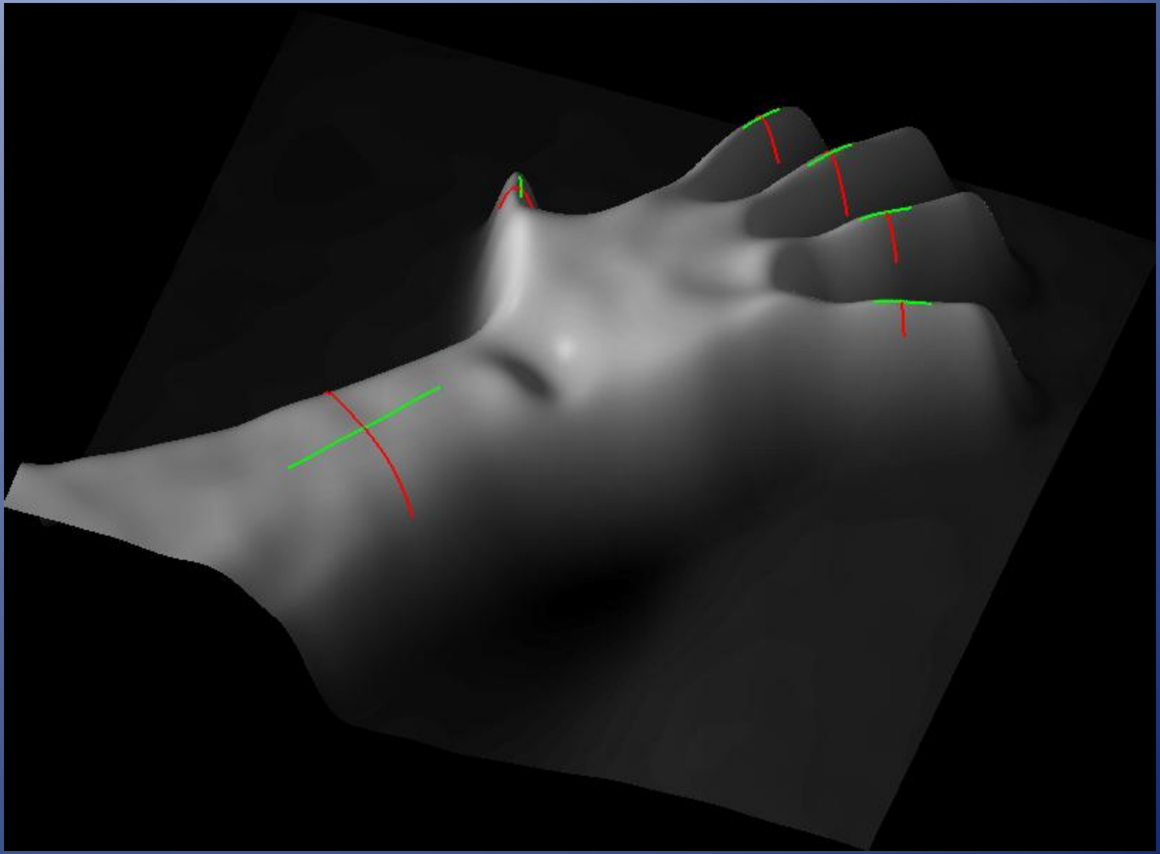
In order to set the scale, we convolve the image I with a Gaussian kernel prior to differentiating.

* A similar definition can be found in [1]

Single-scale ridge

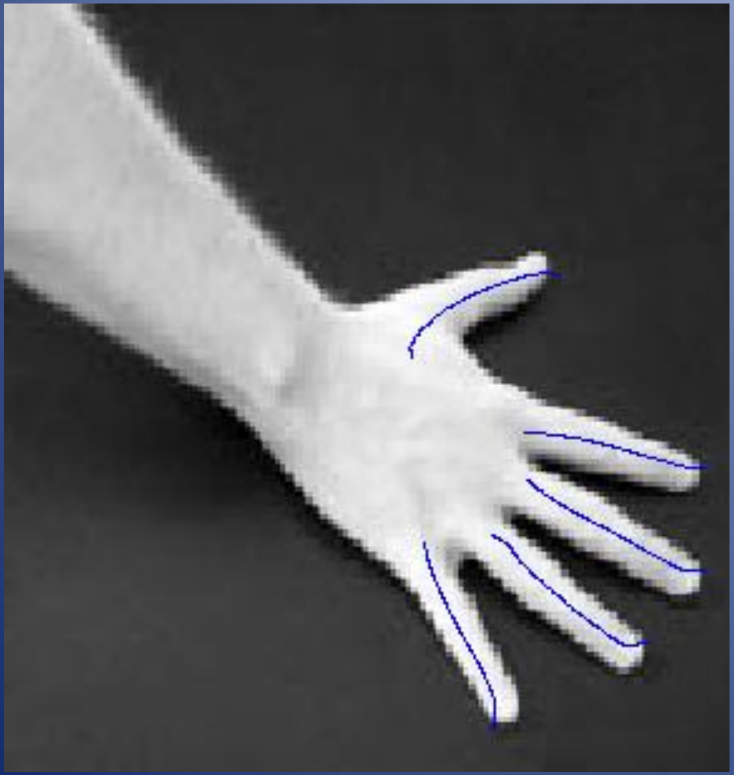
A point (x, y) is a ridge point if the following conditions are satisfied:

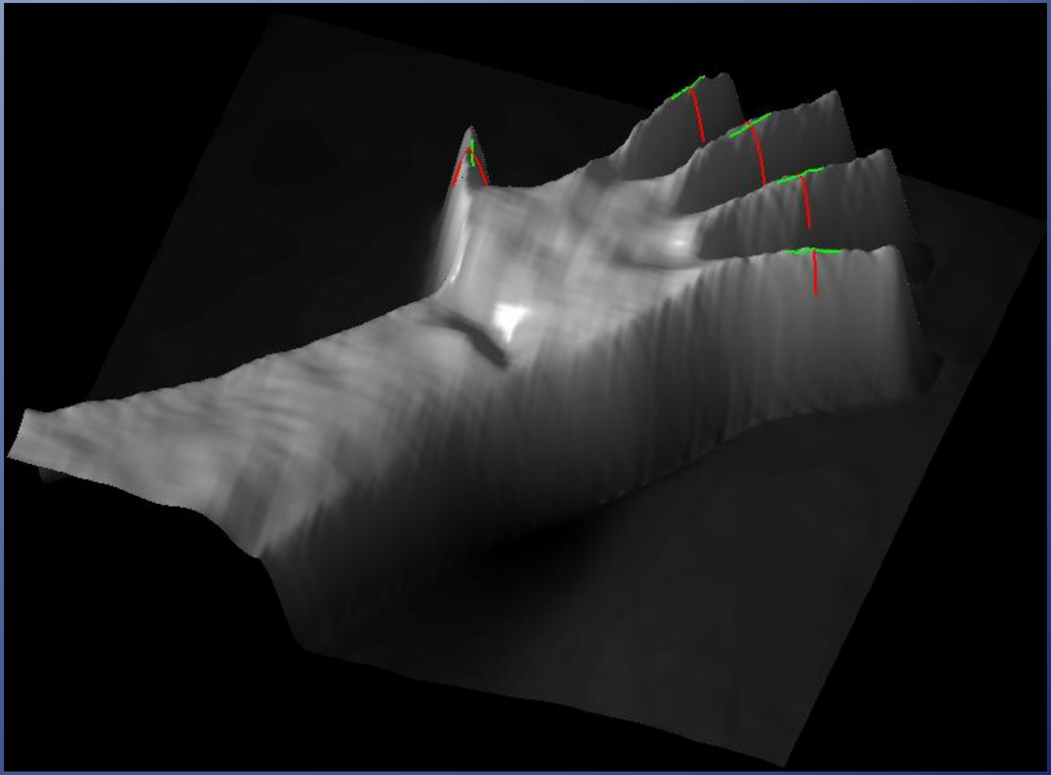
1. $\langle \nabla I, v_{\max} \rangle = 0$
2. $\lambda_{\max} < 0$



Single-scale ridge

Problem: A single scale cannot capture all the ridges.

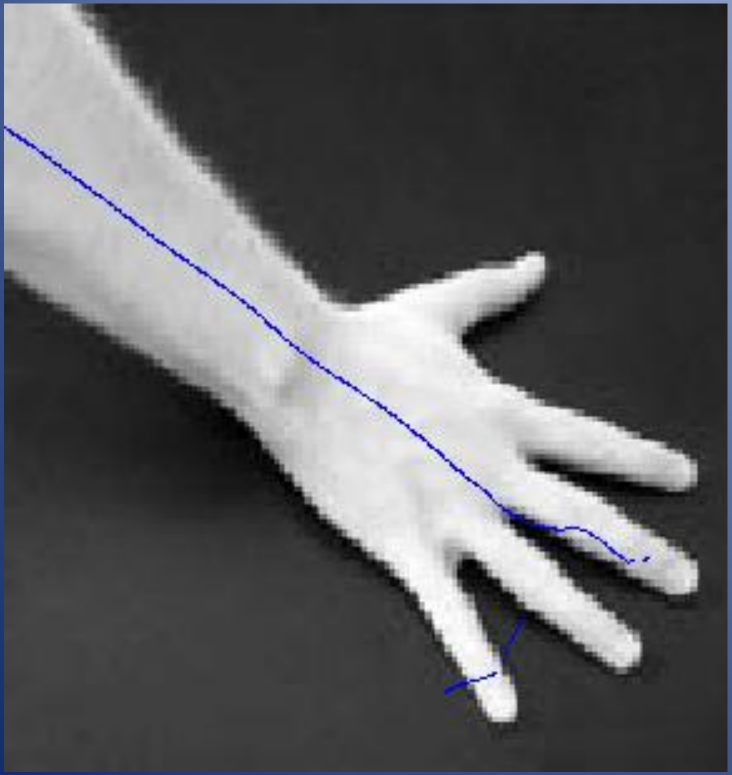


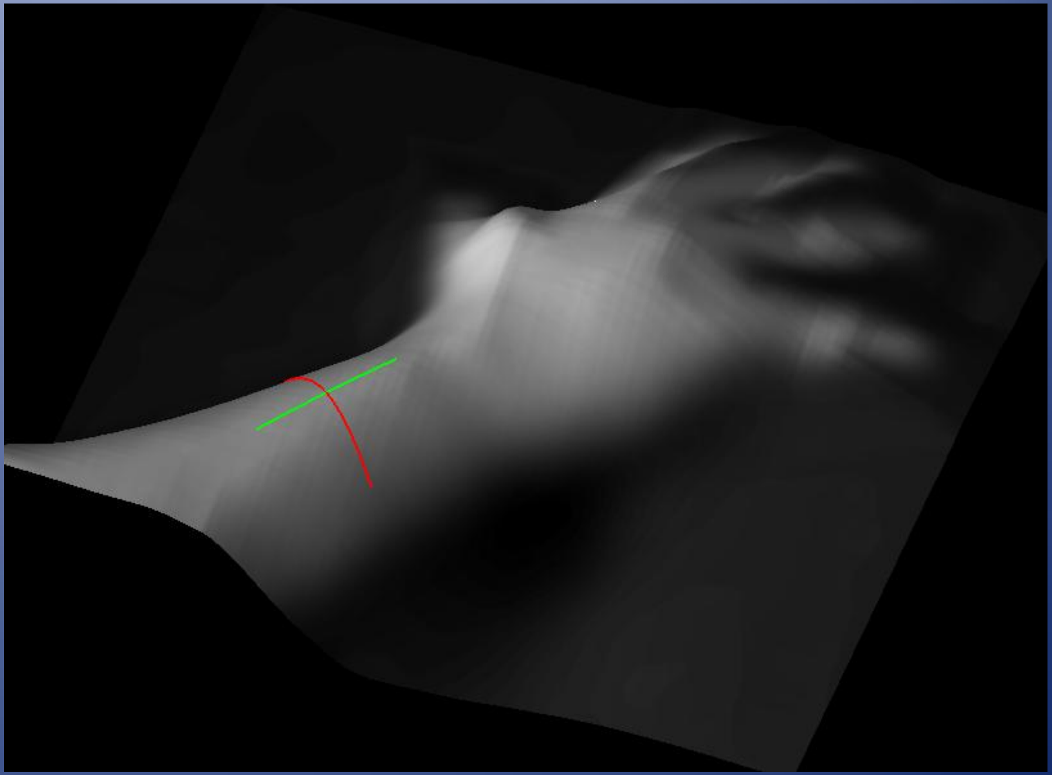


A small scale is good for the fingers, but leaves small unwanted peaks and summits on the arm.

Single-scale ridge

Problem: A single scale cannot capture all the ridges.

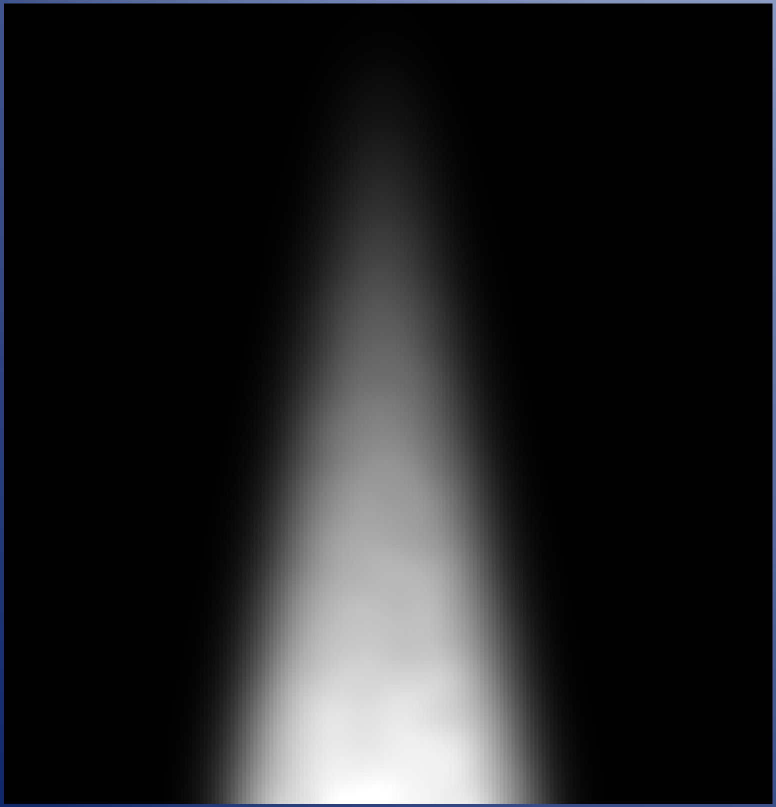


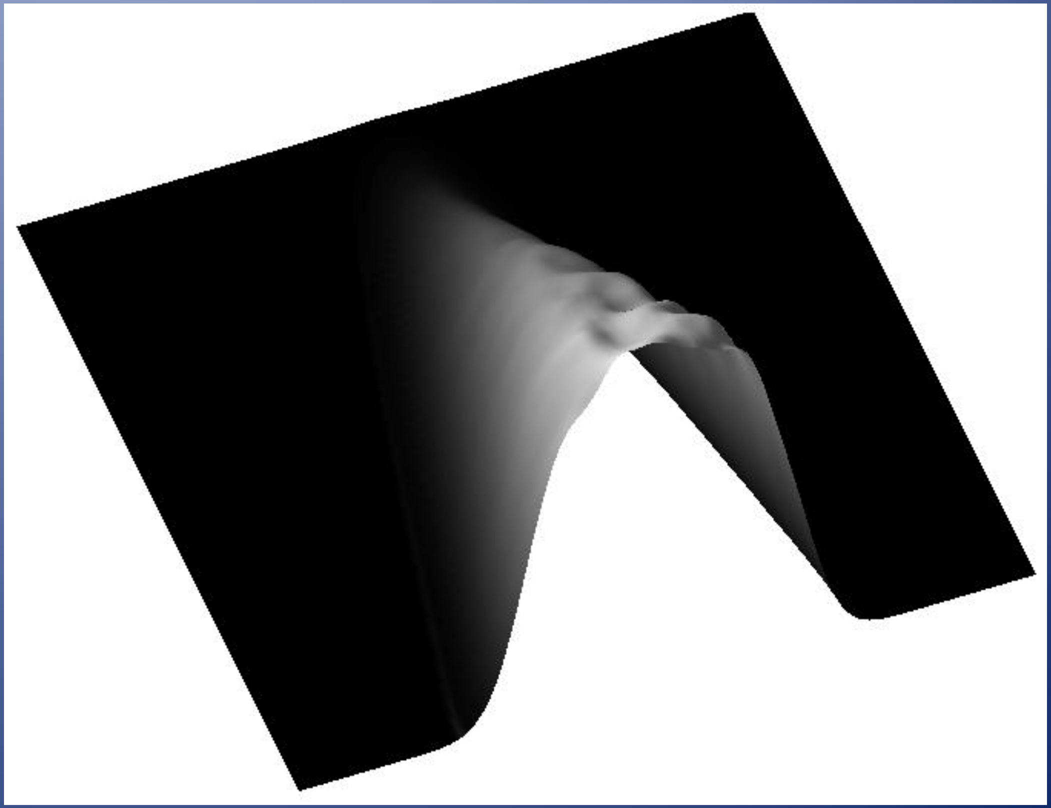


A larger scale is good for the arms, but wipes out the fingers.

Single-scale ridge

So why not perform single-scale ridge detection at each scale separately and combine the results?





A ridge's width can change along its path.

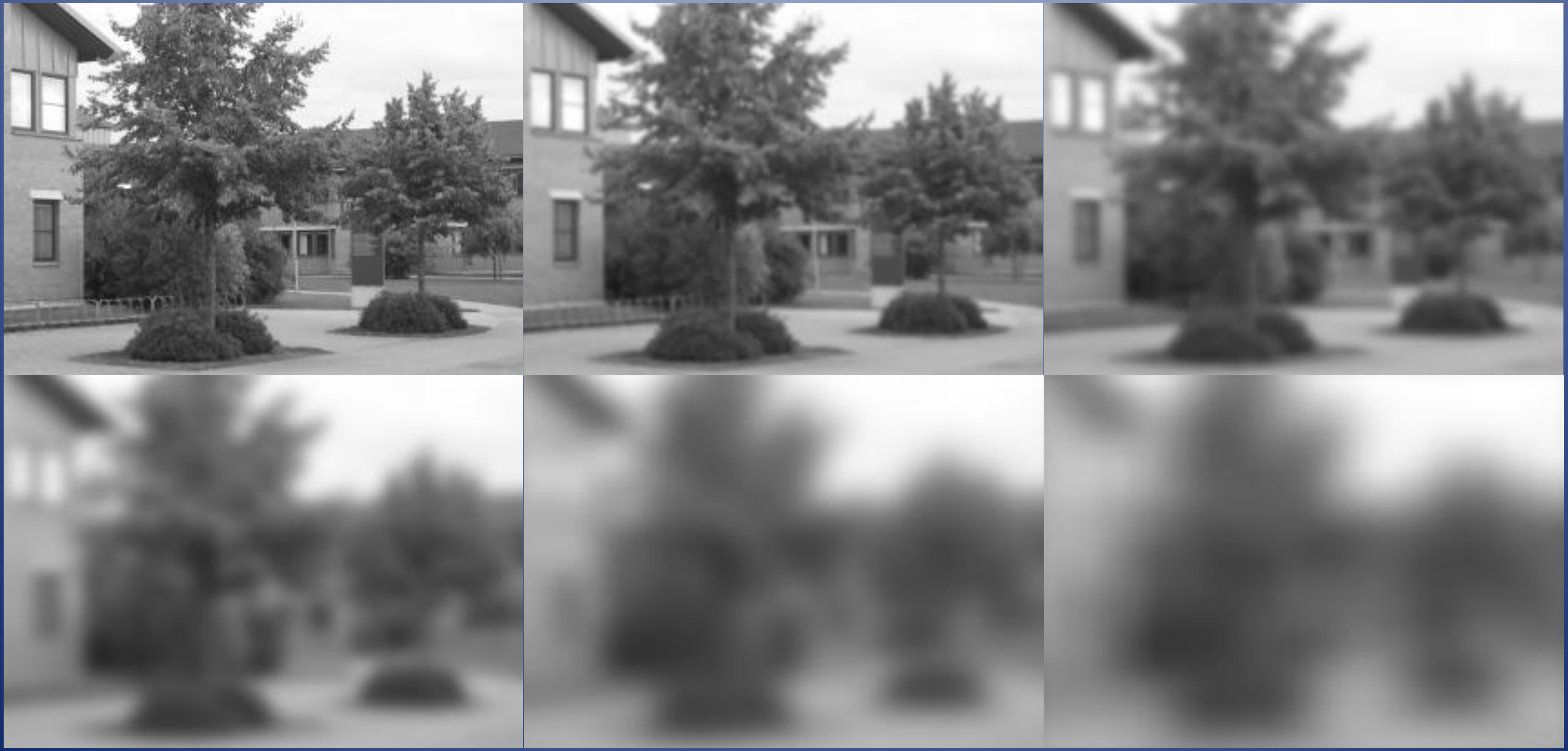
Linear Scale Space

- In [1] Lindeberg used Gaussian smoothing for defining the scale-space according to Witkin and Koenderink [2,3].
- For a given image *I*(*x*,*y*), its *linear (Gaussian) scale-space representation* is a family of signals *L*(*x*,*y*;*t*) defined by the convolution of *I*(*x*,*y*) with the Gaussian kernel:

$$L\left(x,y,t\right)=I\left(x,y\right)*\frac{1}{2\pi t}e^{-\frac{\left(x^2+y^2\right)}{2t}}$$

Linear Scale Space

Images at different scales:

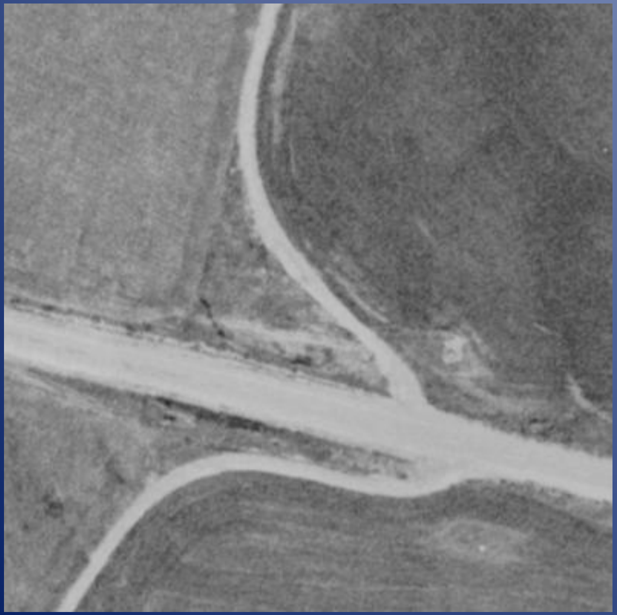


Ridge strength function

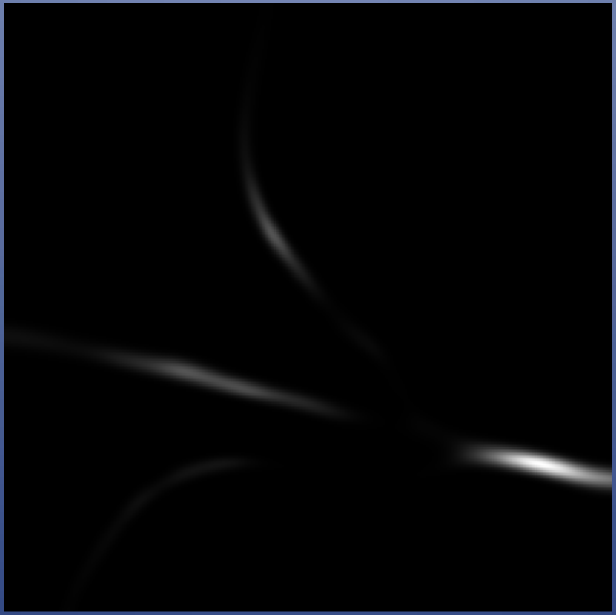
To our assistance, we have a function that measures the ridge strength at each scale-space point:

$$N(x,y,t) = \lambda_{\max}{}^2 - \lambda_{\min}{}^2 = \left|L_{xx} + L_{yy}\right| \sqrt{\left(L_{xx} - L_{yy}\right) + 4L_{xy}{}^2}$$

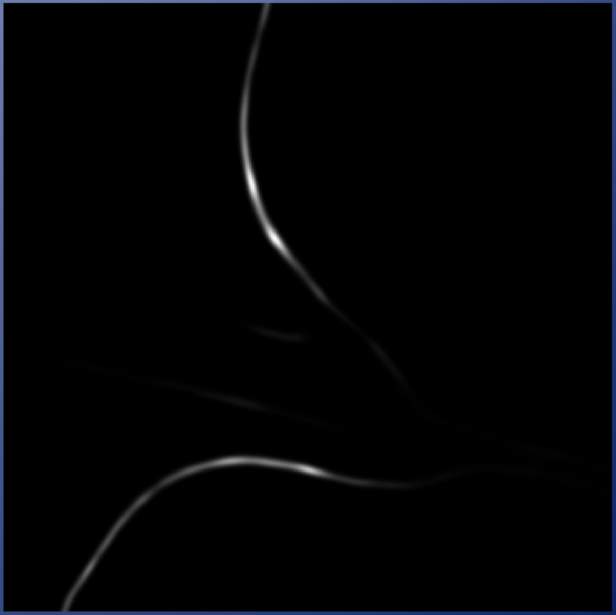
Original image



N(x,y,t) in large scale



N(x,y,t) in small scale

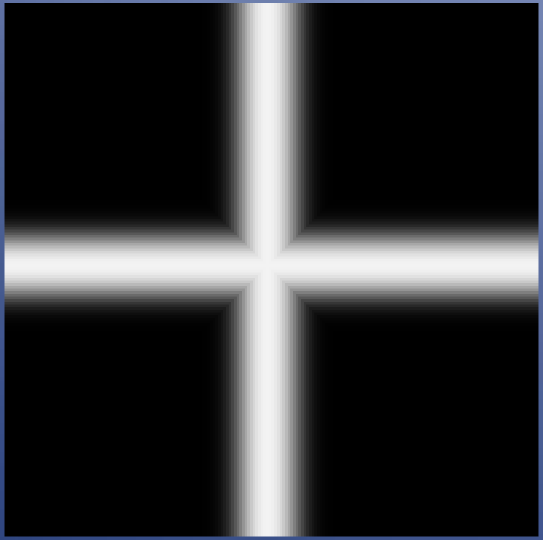


Ridge strength function

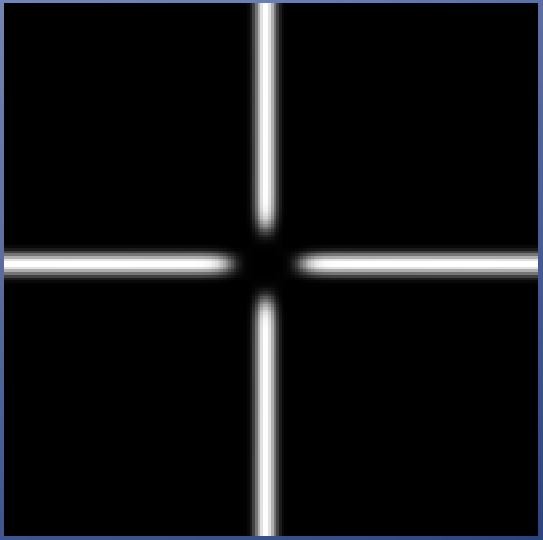
To our assistance, we have a function that measures the ridge strength at each scale-space point:

$$N(x,y,t) = \lambda_{\max}{}^2 - \lambda_{\min}{}^2$$

Original image



N(x,y,t)

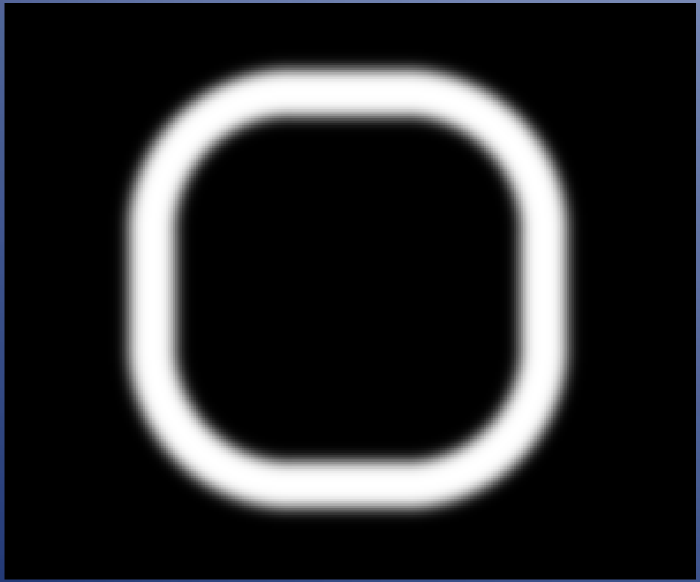


Ridge strength function

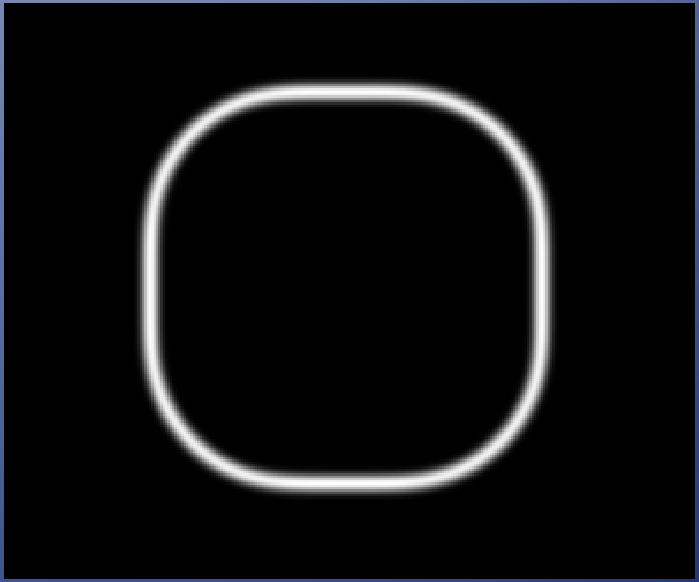
To our assistance, we have a function that measures the ridge strength at each scale-space point:

$$N(x,y,t) = \lambda_{\max}{}^2 - \lambda_{\min}{}^2$$

Original image



N(x,y,t)



Scale Space Ridge

Definition of a scale-space ridge:

Let $L(x,y,t):\mathbb{R}^3 \to \mathbb{R}$ be a linear scale-space image.

Let $\lambda_{\max}$ be the principal eigenvalue of the Hessian matrix of L with respect to x, y at point (x,y,t). Let $v_{\max}$ be an eigenvector corresponding to $\lambda_{\max}$.

A scale-space point (x,y,t) is a *ridge point* if the following conditions are satisfied:

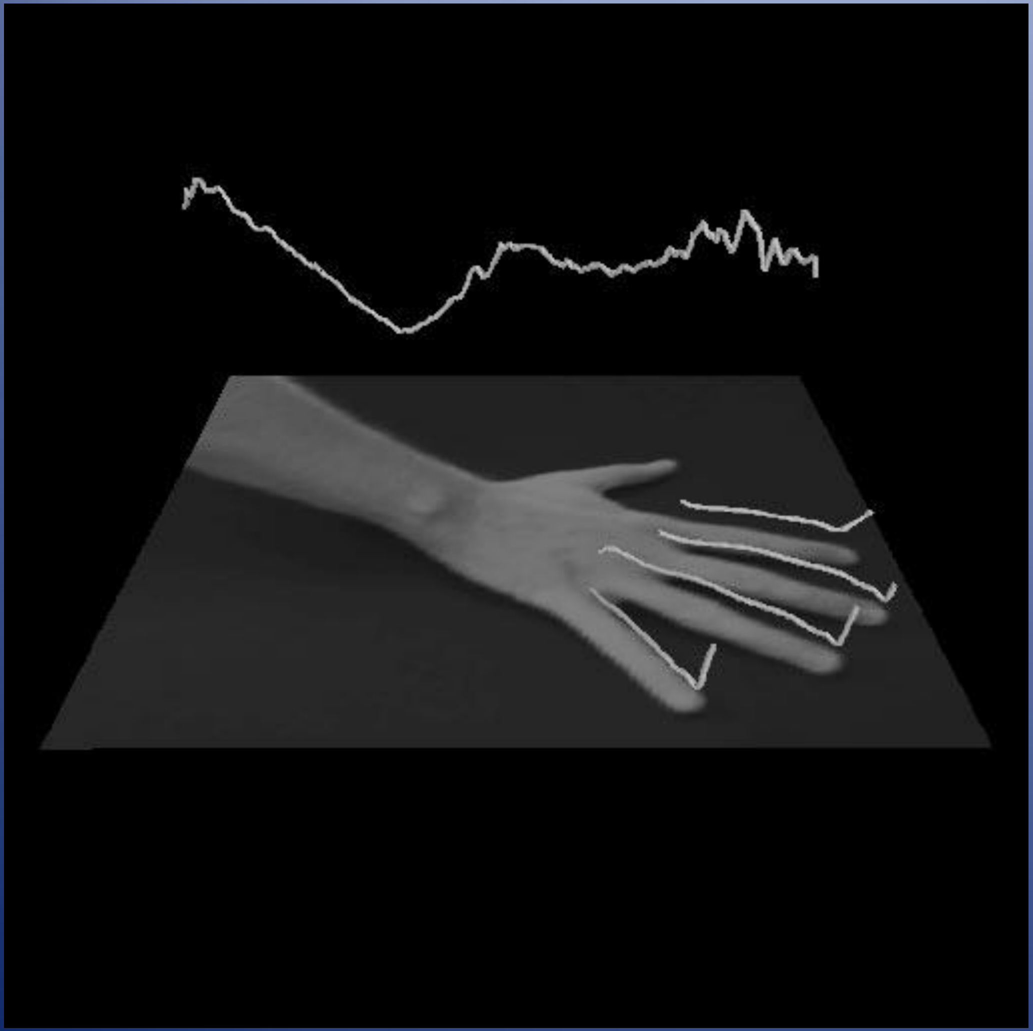
1. $\left\langle \nabla_{x,y} L(x,y,t), v_{\max} \right\rangle = 0$

2. $\lambda_{\max} < 0$ (for bright ridges)

3. The strength function $N(x,y,t)$ obtains a local maximum along the t-axis at (x,y,t).

Linear Scale Space

Some of Lindeberg's results:



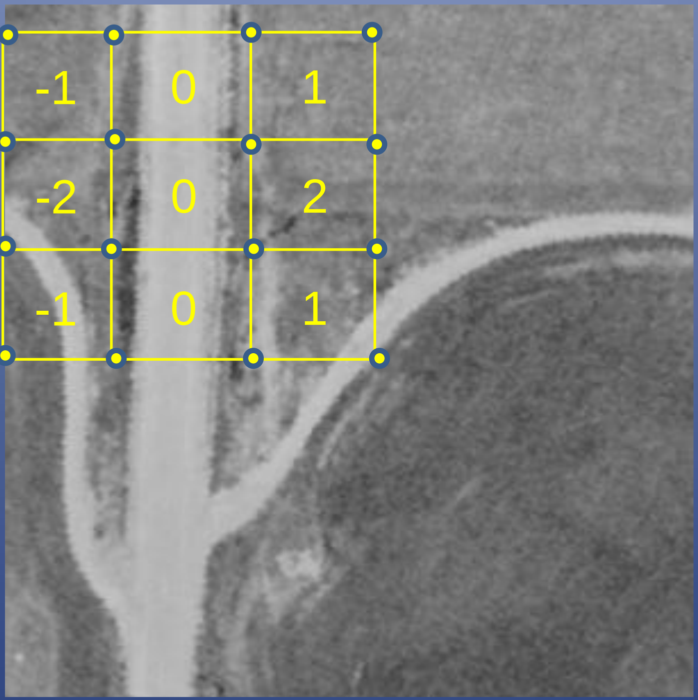
Linear Scale Space

Drawbacks of ridge-tracking using linear scale-space:

- Calculation of the convolution is time consuming, and needs to be done once for each scale.
- Smoothing reduces the magnitude of the ridges. Therefore, a non-intuitive normalization has to be applied to the derivatives.

Our approach

Suppose we would like apply a certain numerical differentiation filter on the image, e.g. Sobel, in a discrete scale S. We apply an enlarged version of the filter, with each element of the original filter inflated to a square of S x S pixels, containing the value 1/S^2, in order to obtain uniform averaging.



Our approach

Differentiating in this method can be implemented very efficiently using Integral Image:

$$II(x,y) = \iint\limits_{\substack{u\le x\\ v\le y}} I(u,v)\,dudv$$

By taking linear combinations of the integral image II(x,y) at different points, we can sum square regions on the image with time complexity O(1).

This property enables us to calculate the scale-space derivatives very efficiently – with time complexity O(k) whereas k is the number of elements in the numerical differentiation filter.

Our approach

Pros:

- Efficient implementation using Integral Image.
- Ridges in different scales which share the same intensity produce the same magnitude of derivative when applied in their appropriate scales.

Cons:

- This method is not rotation-invariant.
- Uniform averaging is a bit less informative than Gaussian averaging.

Tracking algorithm

Since we are interested at curves, and not points, we need to use a tracking algorithm that would track the zero-crossing curves of the function $\langle \nabla_{x,y} L, v_{\max} \rangle$, and take only those which have $\lambda_{\max} < 0$.

For this we need to develop formulae for $v_{\max}$ and $\lambda_{\max}$.

$$\sigma = sign(\Delta I), \;\; d = \left(L_{xx} - L_{yy}\right)/2, \;\; S = \sqrt{d^2 + L_{xy}^{\;\;2}}$$

$$\lambda_{\max} = \frac{\Delta L}{2} + \sigma S$$

Since $v_{\max}$ is defined up-to a multiplication by (-1), we have two equivalent formulae for obtaining the direction $\mathrm{v}_{\max}$, each is a negative multiple of the other:

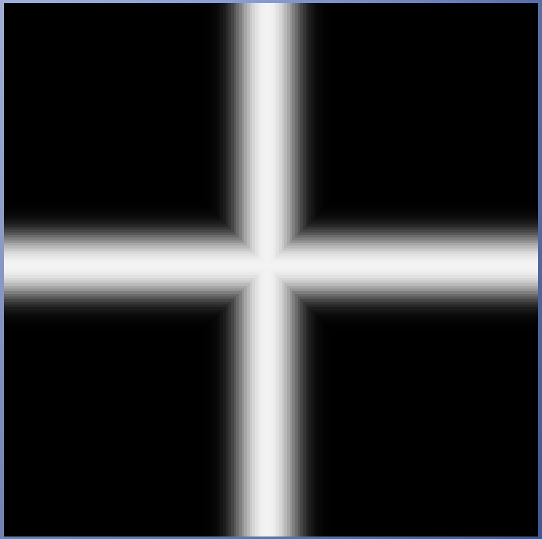
$$v_1 = \left(d + \sigma S, L_{xy}\right)$$

$$v_2 = \left(L_{xy}, -d + \sigma S\right)$$

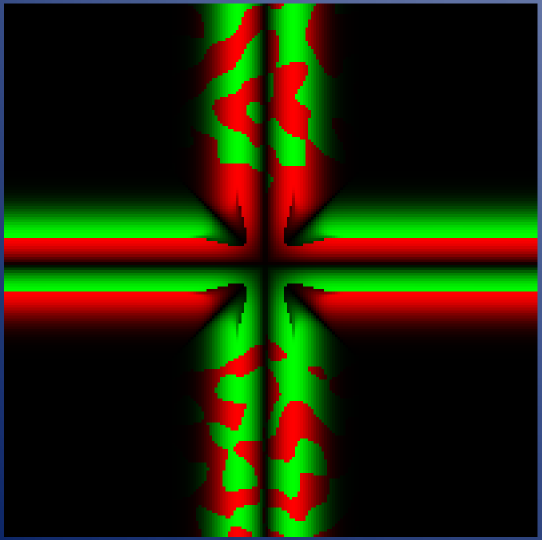
First try at ridge tracking:
In order to track the ridges, we can seek the zero-crossings of the functions

$f_1 = \langle \nabla I, v_1 \rangle$ or $f_2 = \langle \nabla I, v_2 \rangle$

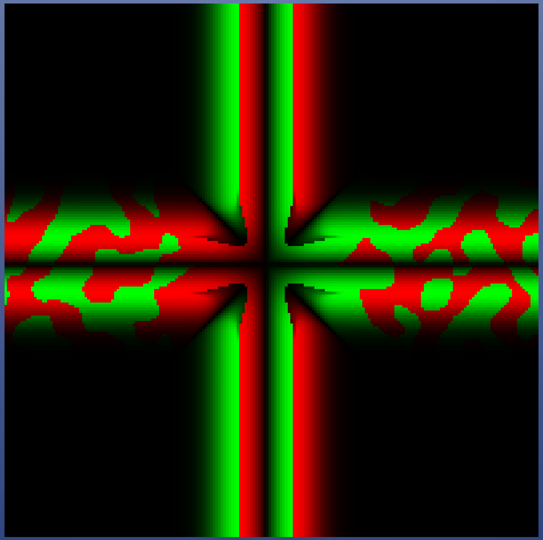
Original image



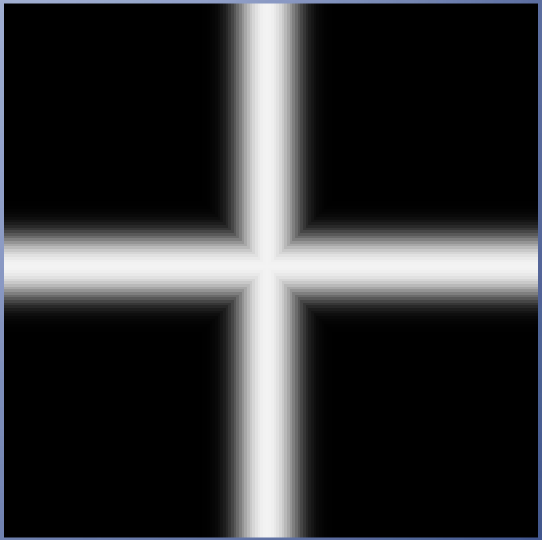
f_1



f_2



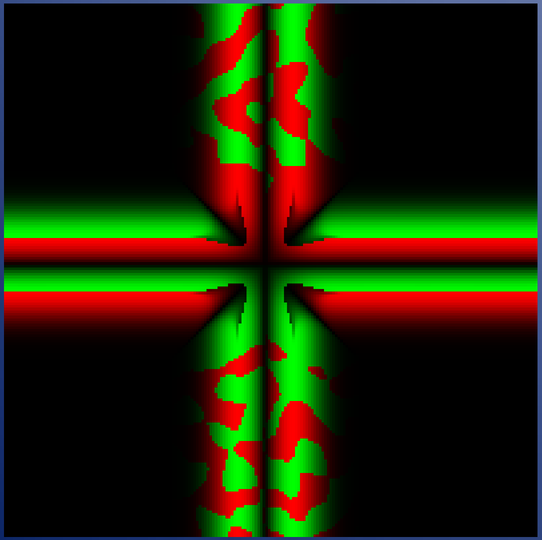
Original image



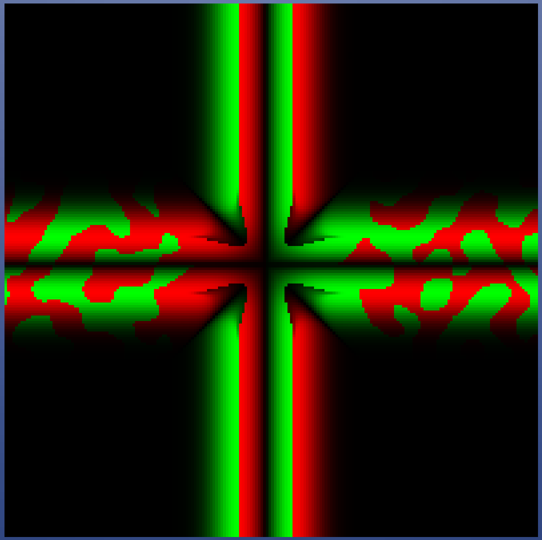
Second try:
Let's choose at each point the one of the two functions which obtains a larger absolute value:

$$f_{\max} = \arg\max_{f_1, f_2} |f_i|$$

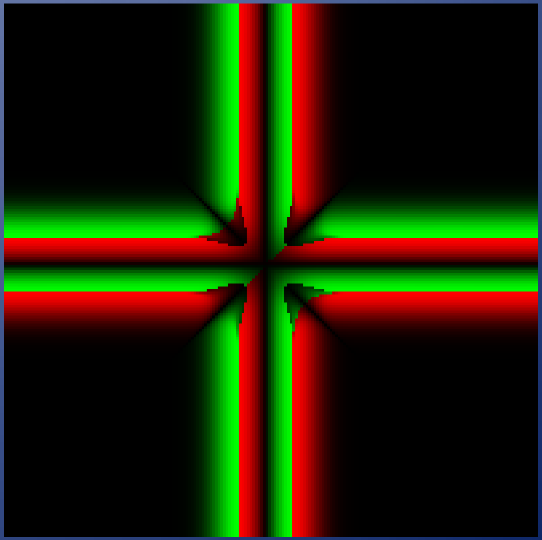
f_1



f_2

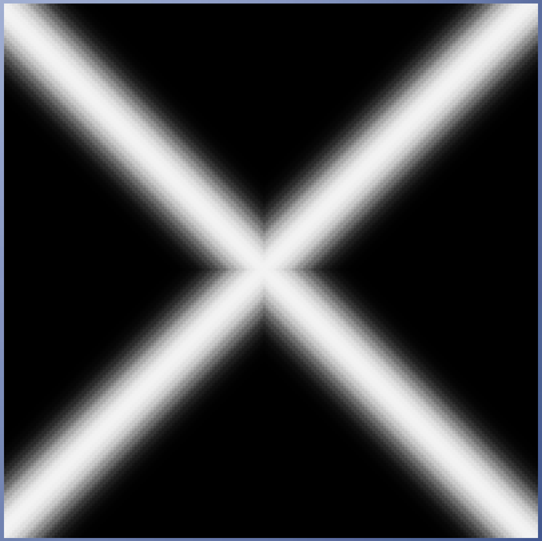


$f_{\max}$

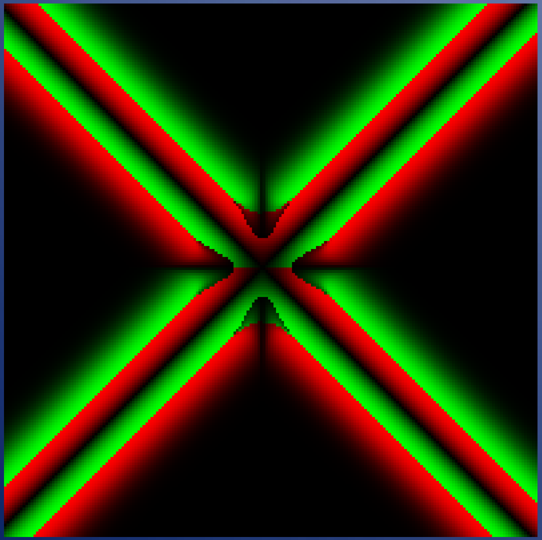


Second try:
That seemed to do the trick.
But what about this image?

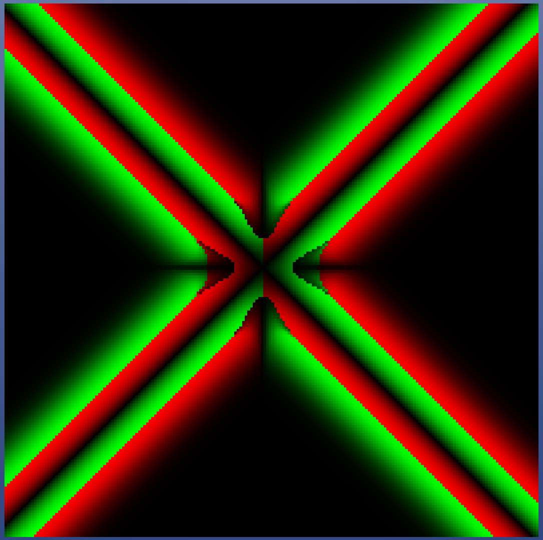
Original image



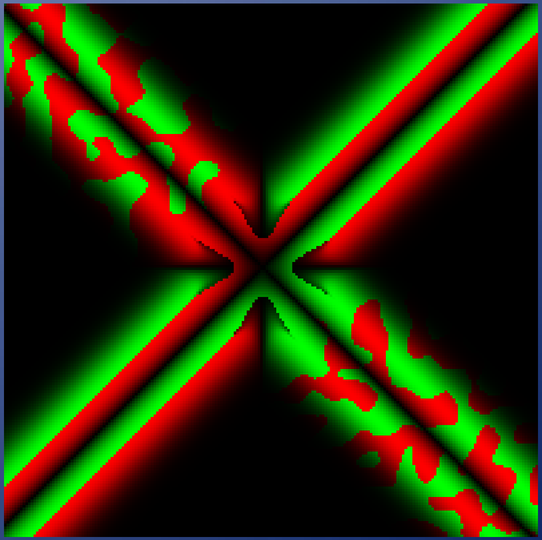
f_1



f_2

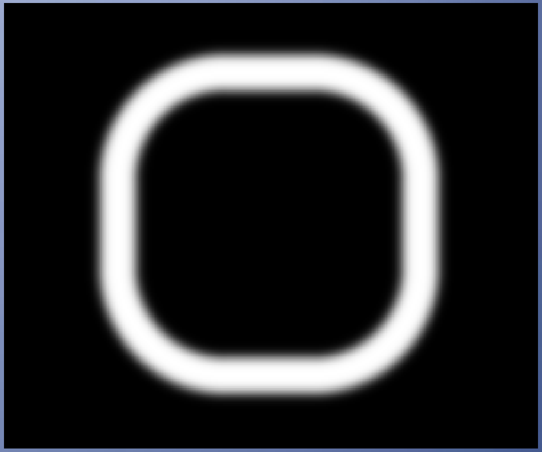


$f_{\max}$

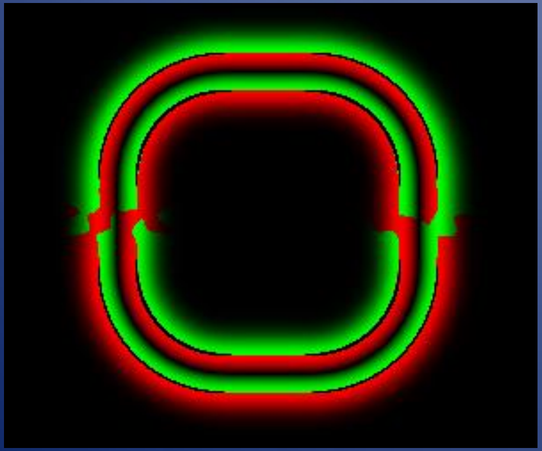


Second try:
…and this?

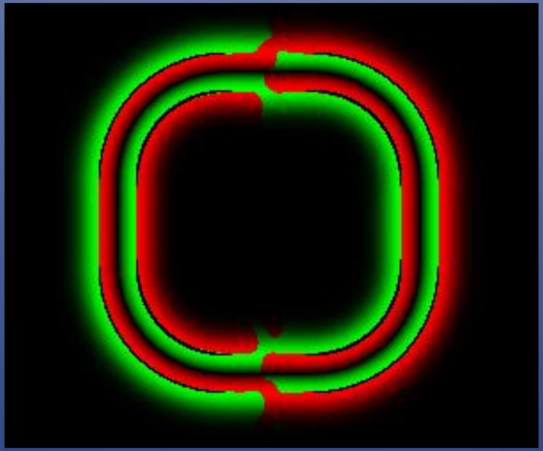
Original image



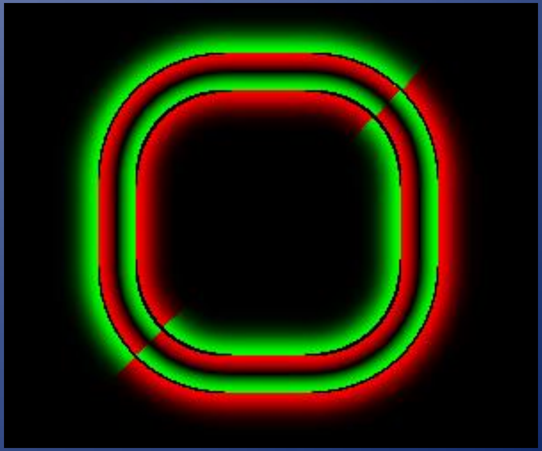
f_1



f_2



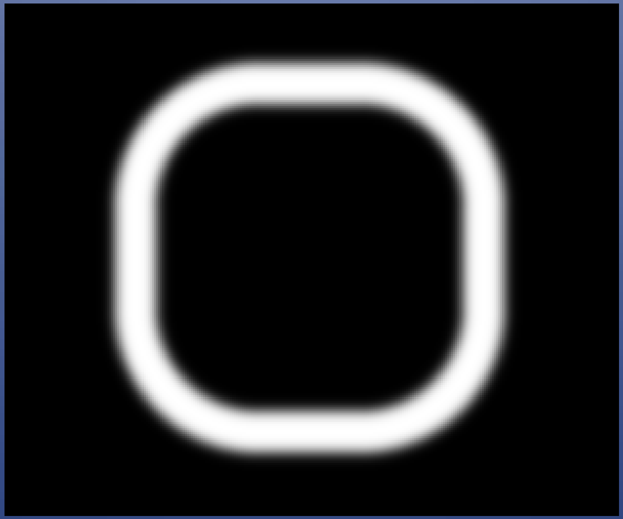
$f_{\max}$



Renen’s proof of nonexistence

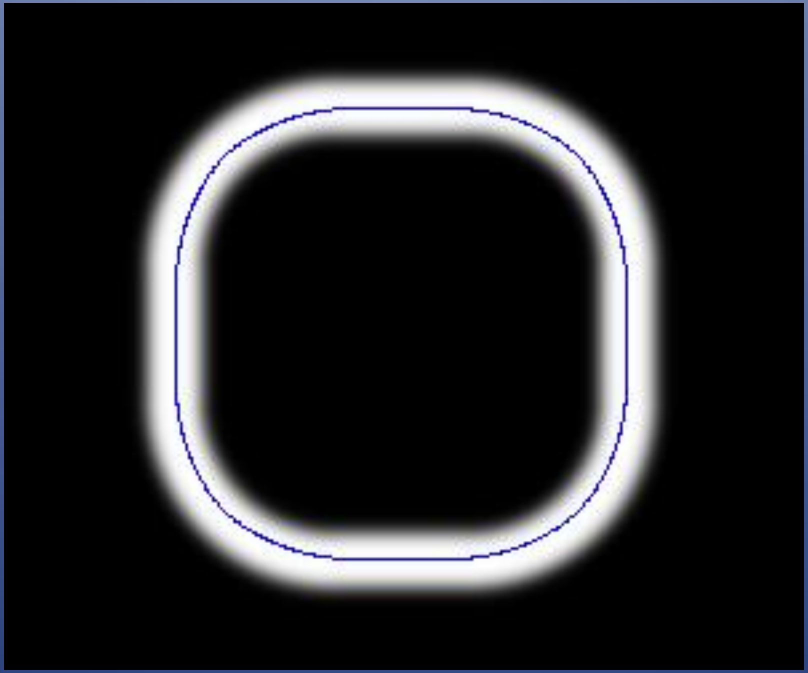
There is a deeper reason that we always get it wrong in at least one of the functions. R. Adar has provided us with an intuitive proof that we cannot base our ridge tracking on the zero-crossings of a single function f. That is, we cannot construct an algorithm that given an arbitrary image I, tracks ridges by searching for zero-crossings of a single function f, that is derived from the image by using local differentiation operators.

Suppose we would like to search this image for ridges:



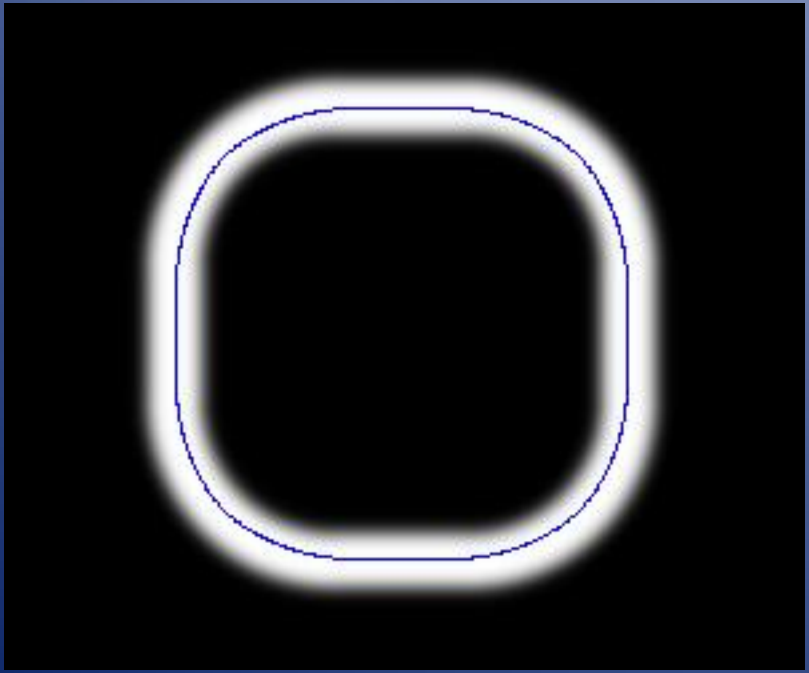
Renen’s proof of nonexistence

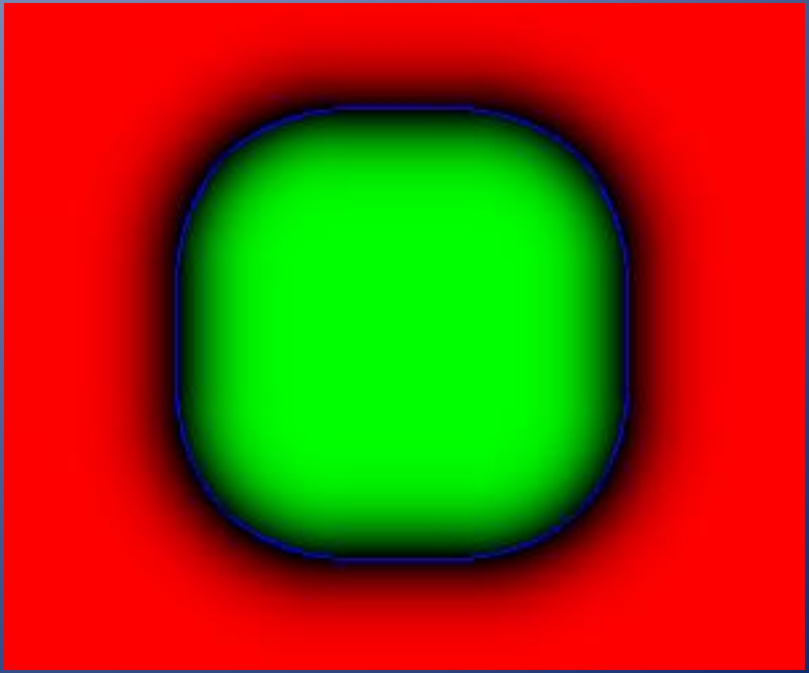
Obviously, we expect the result to look something like this:



Renen’s proof of nonexistence

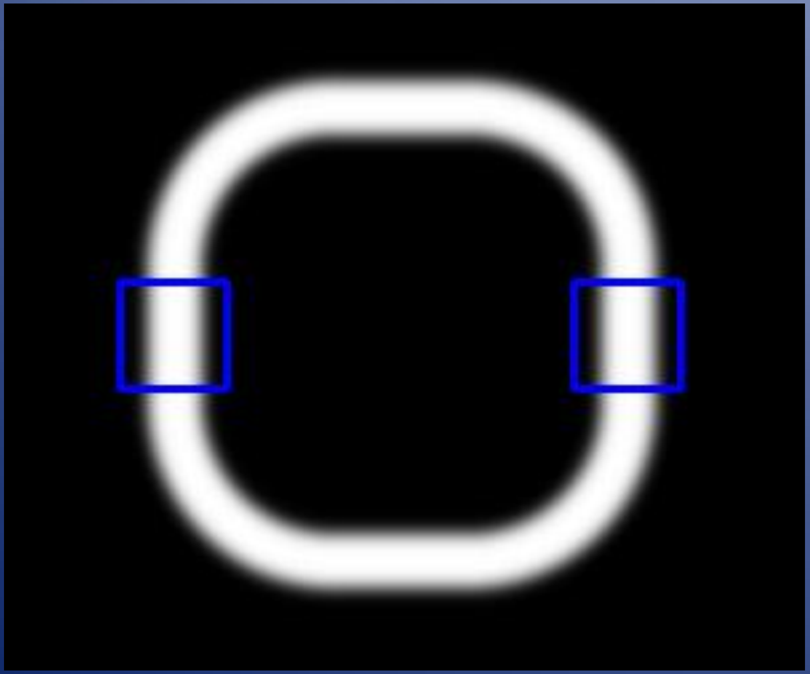
A function which has that curve as its zero-crossing, is w.l.o.g positive in the interior of the curve, and negative outside.

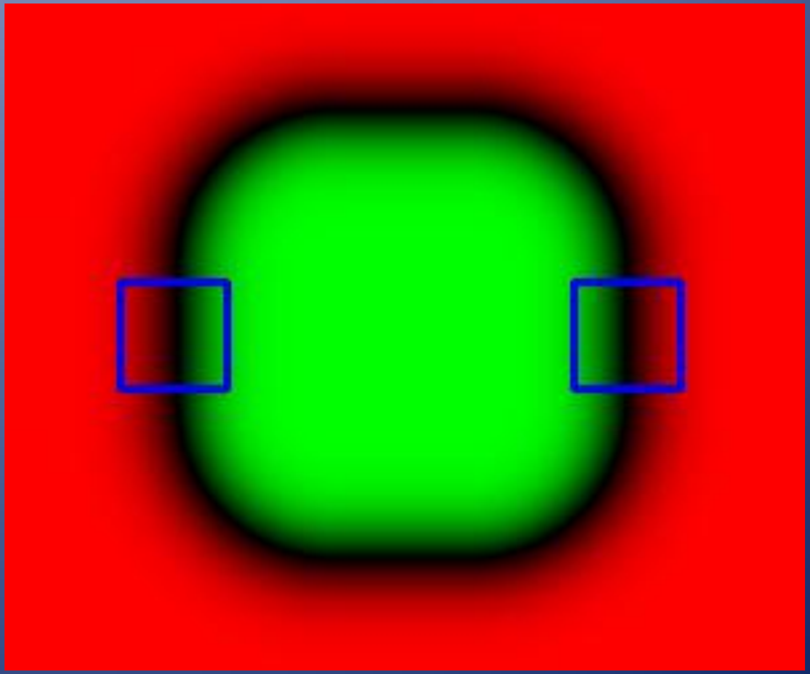




Renen’s proof of nonexistence

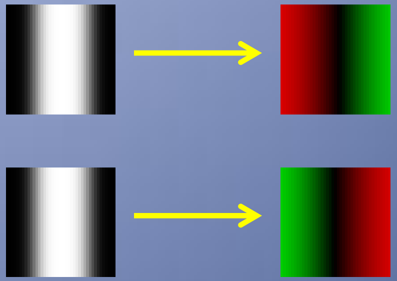
Let us look closely at two neighbourhoods in the original image and the function f.

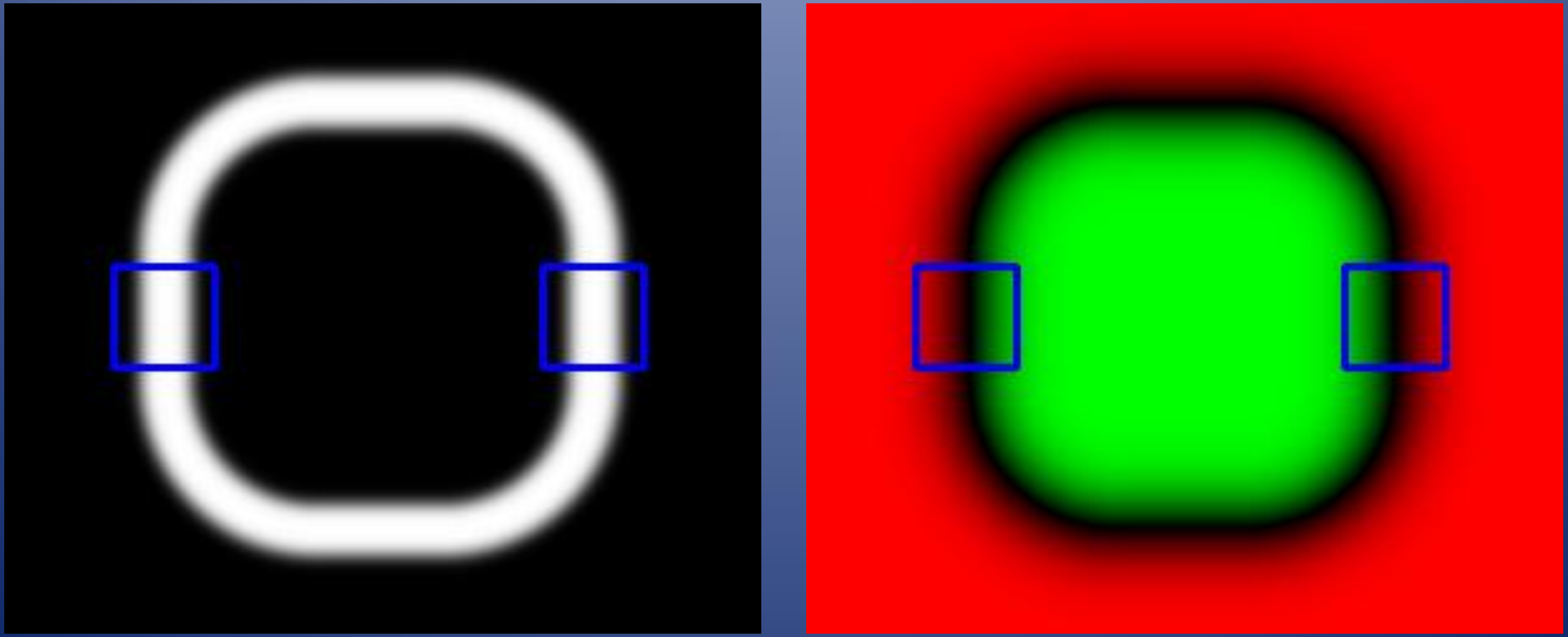




Renen's proof of nonexistence

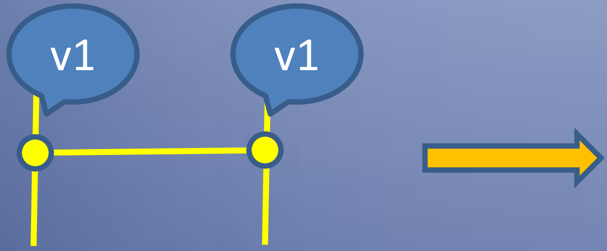
Both neighbourhoods look the same on the original image, but are different in f, even though f was obtained only by local differentiation, and that is impossible.



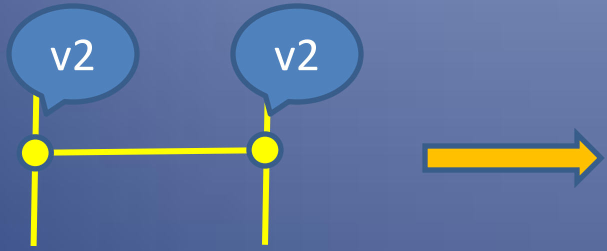


The solution – choose the function locally

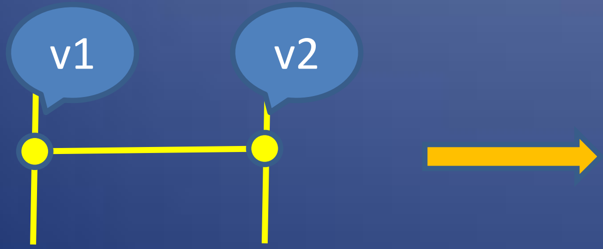
How we will choose the right vector ?



choose v1



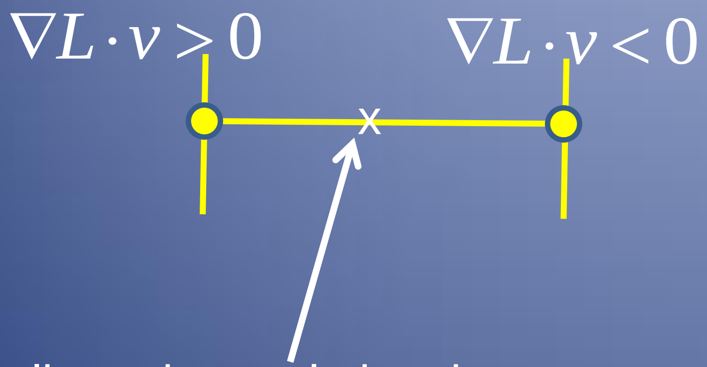
choose v2



choose $\arg\max_{\{v_1,v_2\}} \|v\|$

The solution – choose the function locally

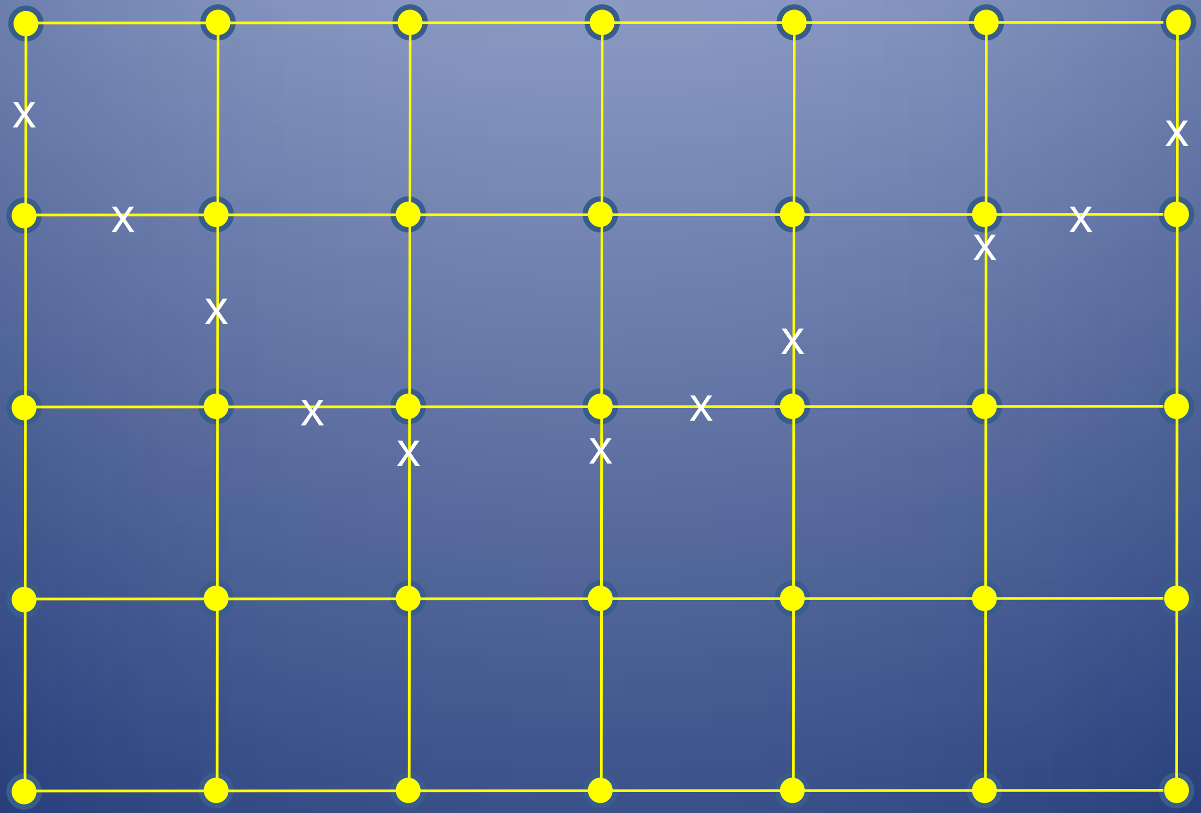
Where is the zero-crossing located ?



via linear interpolation the zero crossing is here

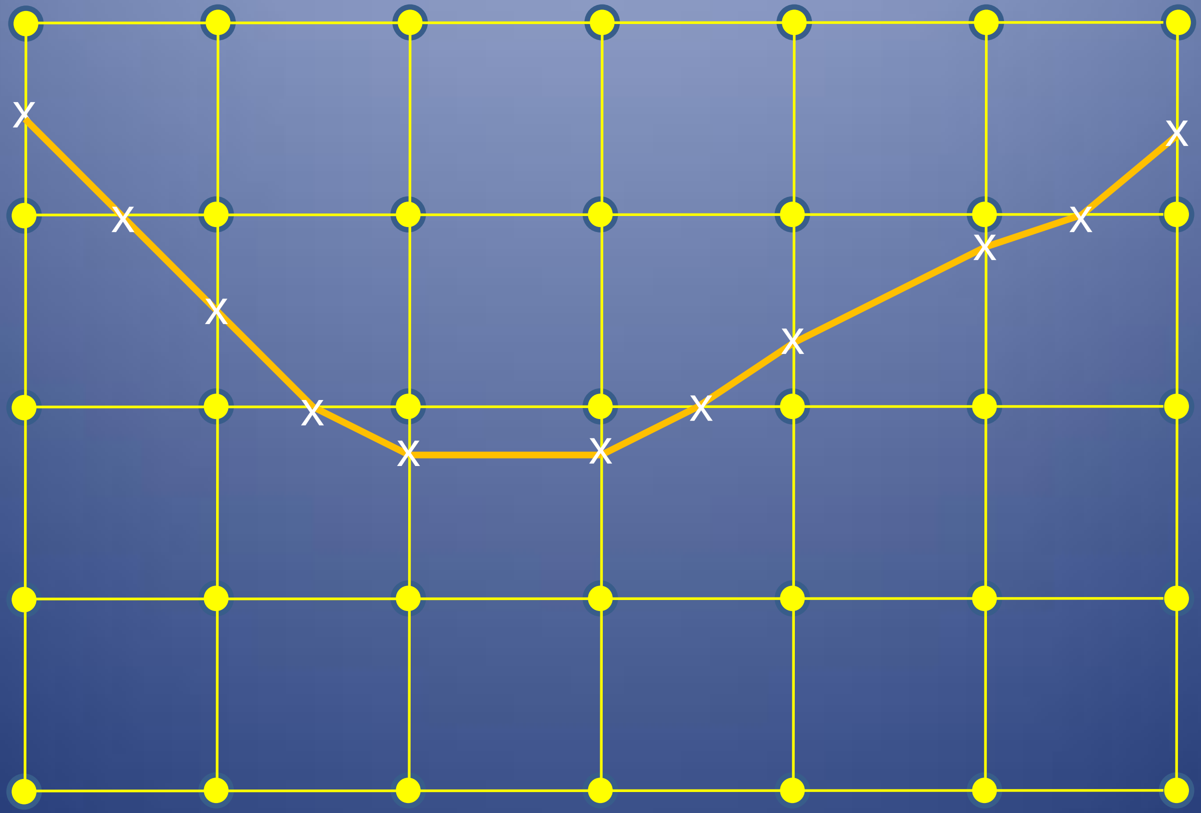
The solution – choose the function locally

By using the previous rule on a grid we get:



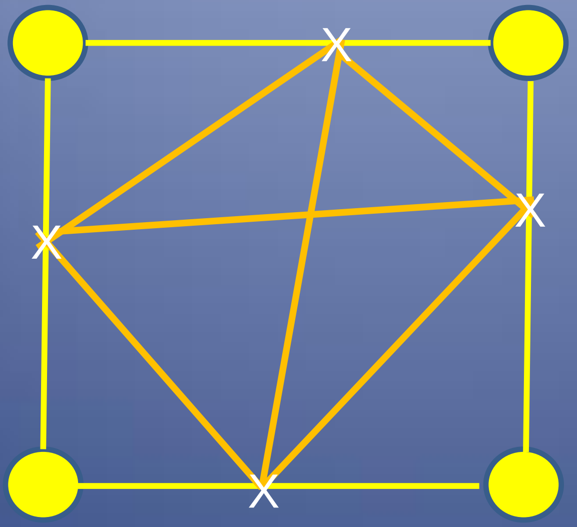
The solution – choose the function locally

We connect the zero-crossing points by edges



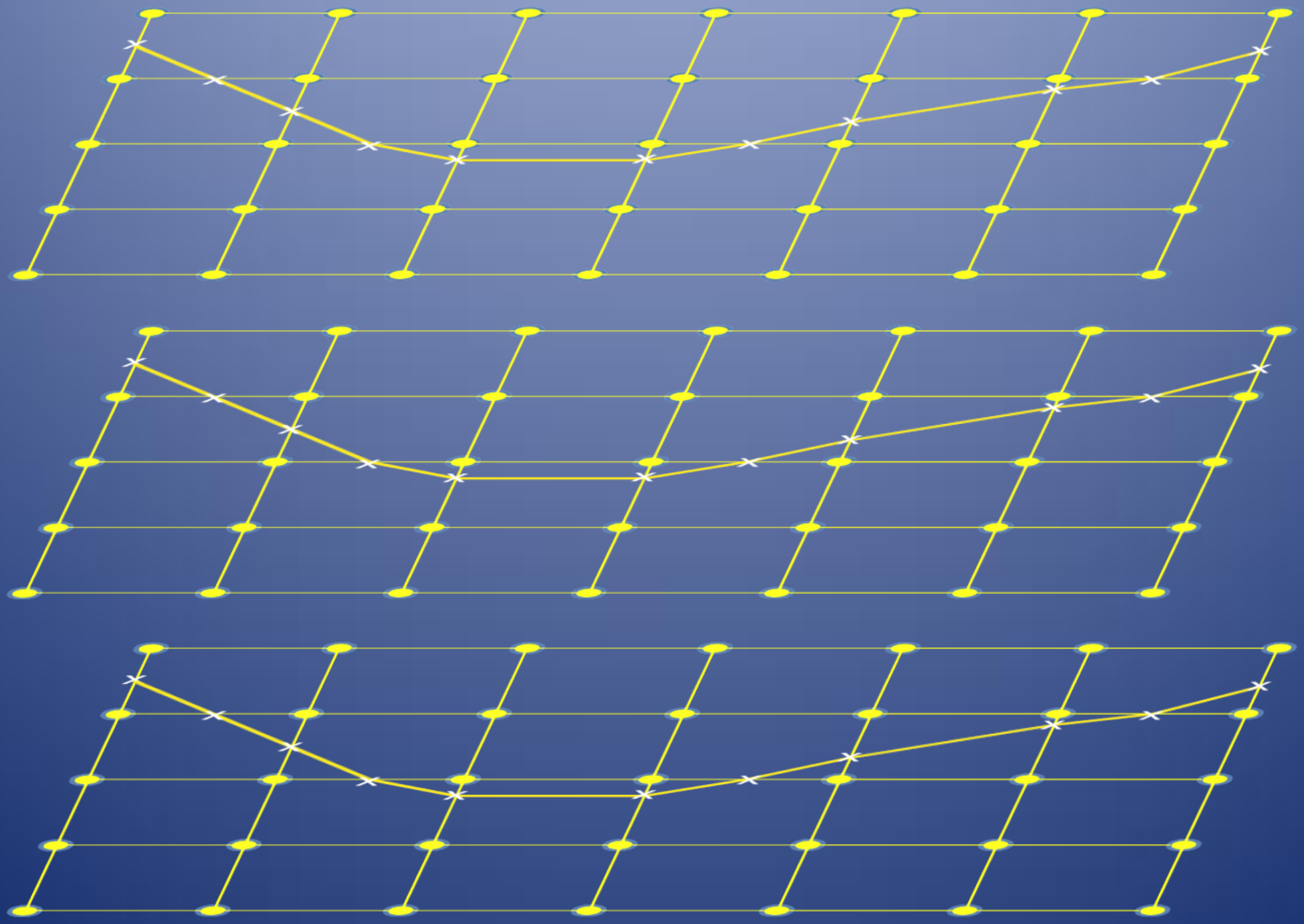
The solution – choose the function locally

In case of ambiguity we create all the possible edges



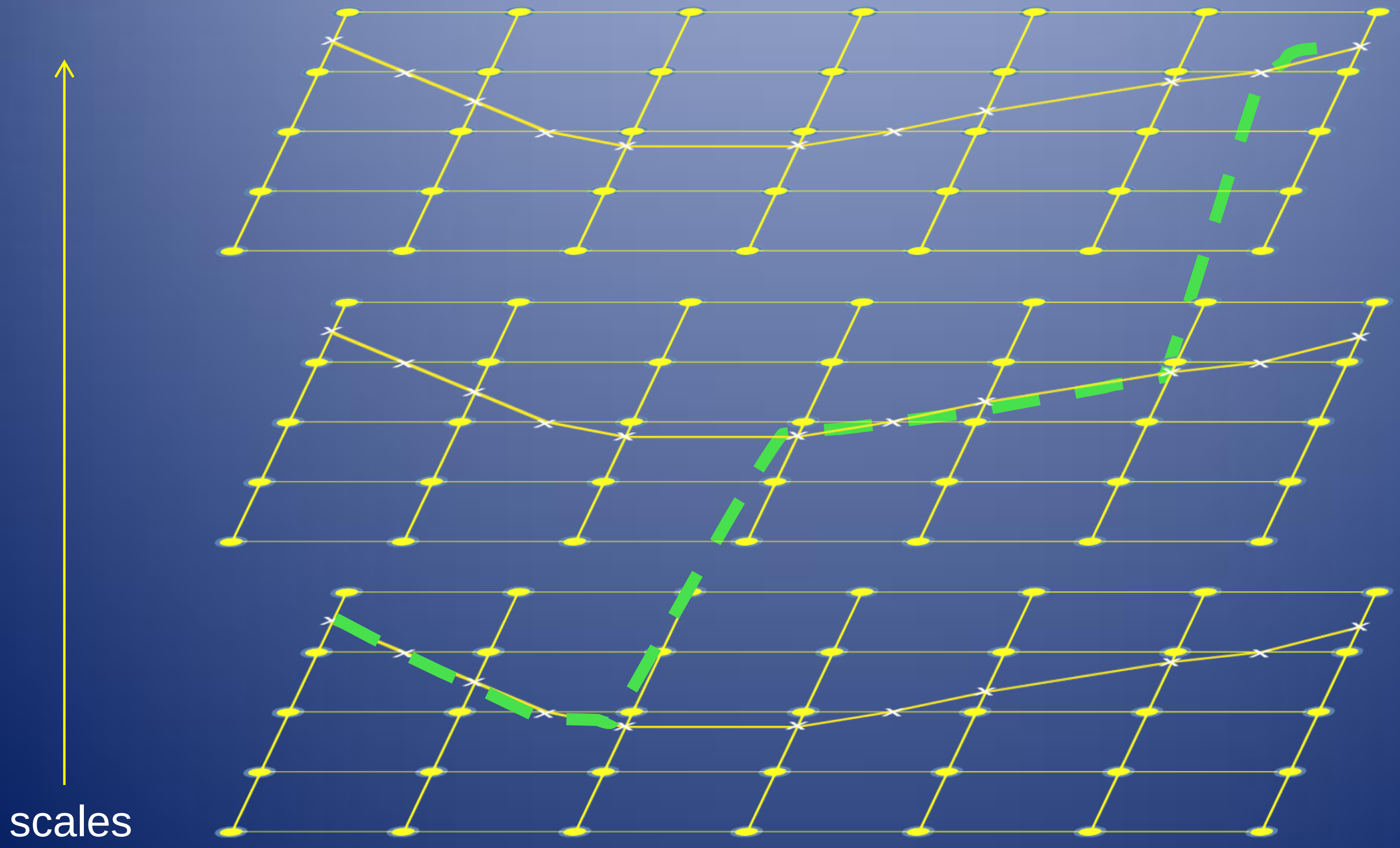
The solution – choose the function locally

We do it separately for every scale



The solution – choose the function locally

We do it separately for every scale



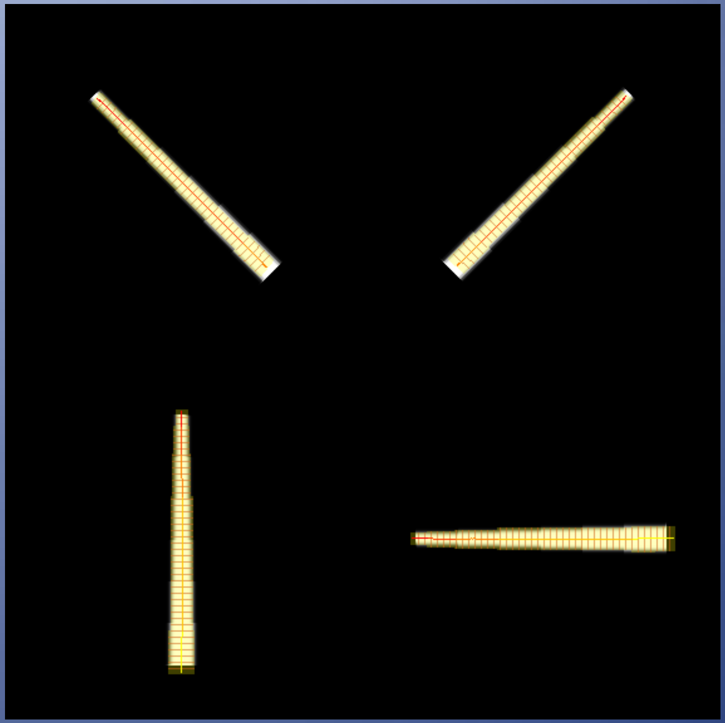
Tracking algorithm highlights

- Accumulating data about connected components by using Union-Find data structure during the graph construction.
- Integrating the strength function over each connected component in order to evaluate its quality.
- Prioritizing scale-space points by their component strength and their strength.
- Seeking the scale-space neighborhood for scale jump opportunities.
- The scale is proportional* to the ridge width, so we can determine the width from the scale.
- Keeping a record of ridge ownership over voxels in their domain.
- Preprocessing is performed independently at each scale, so it can be parallelized.

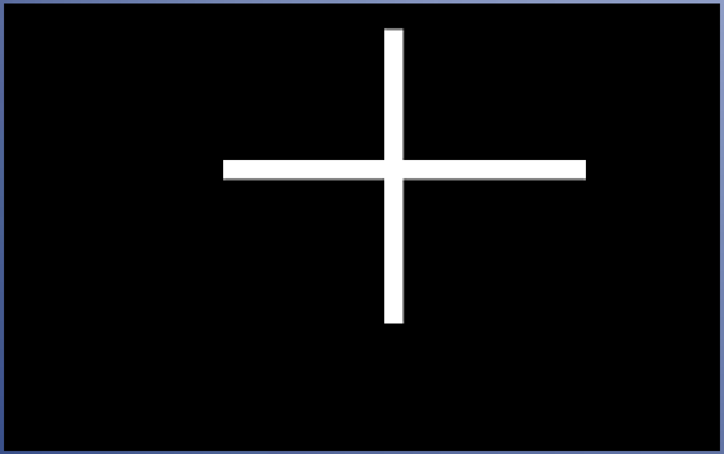
* Up to a function that depends on the angle, due to non-rotational-invariance.

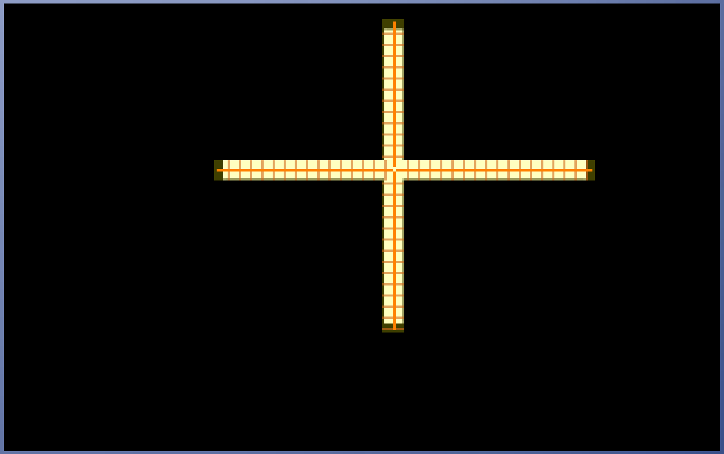
Now it’s time for some examples

Synthetic data

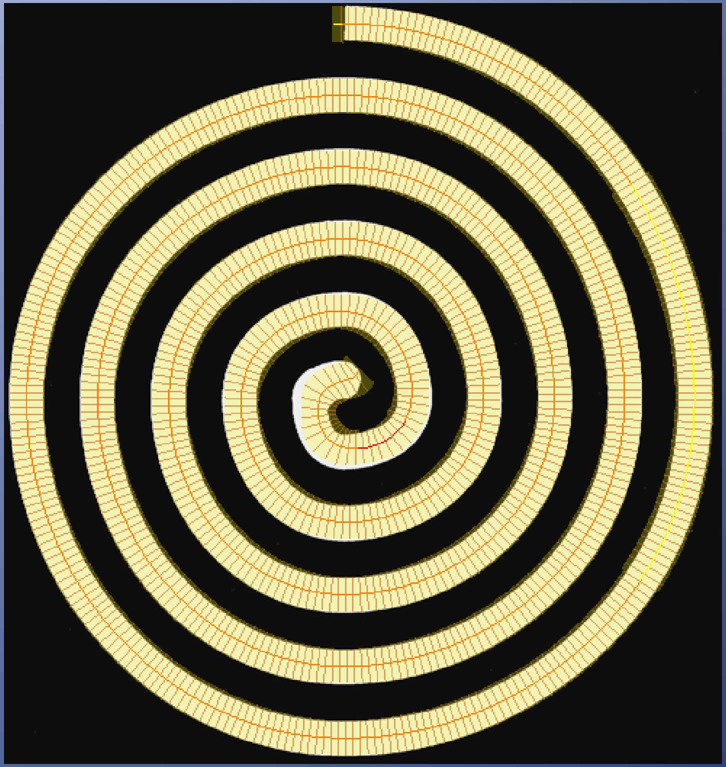


Synthetic data

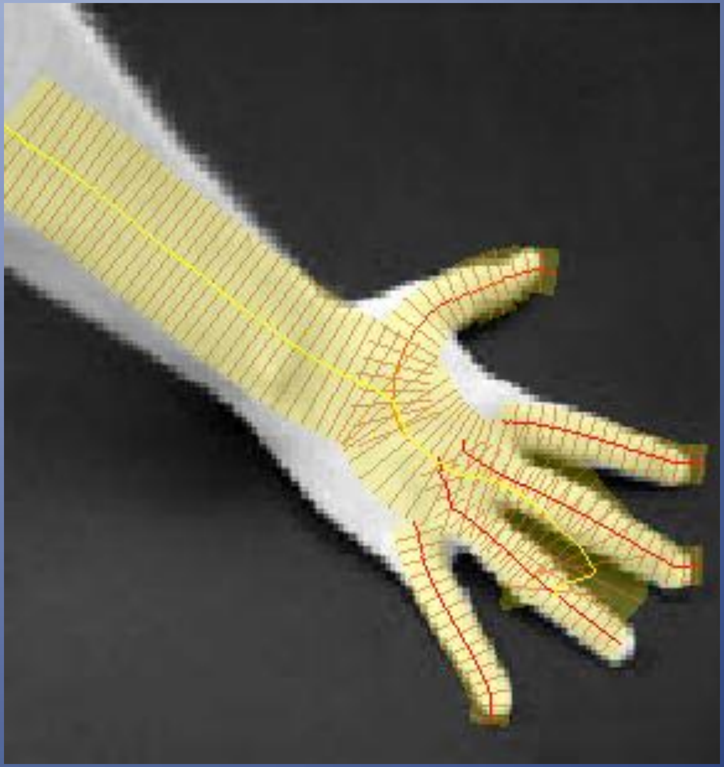




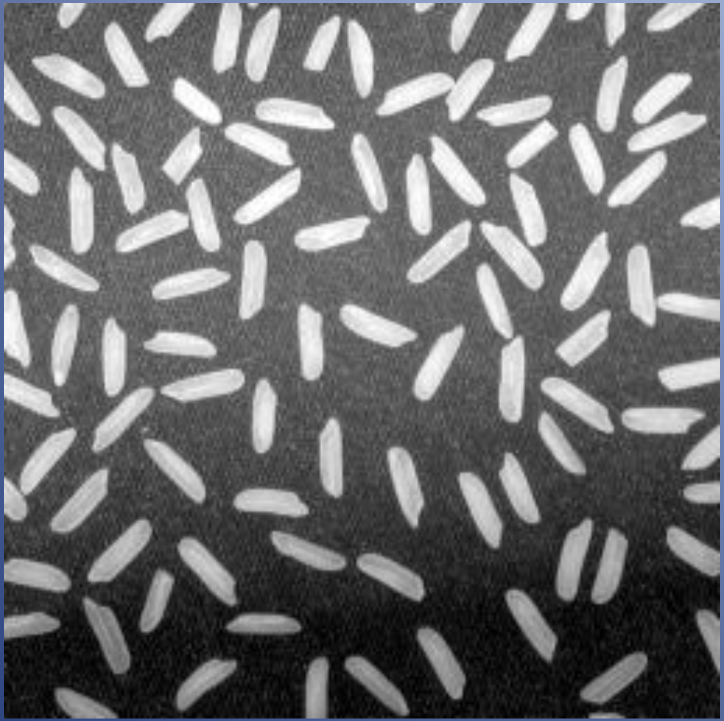
Synthetic data

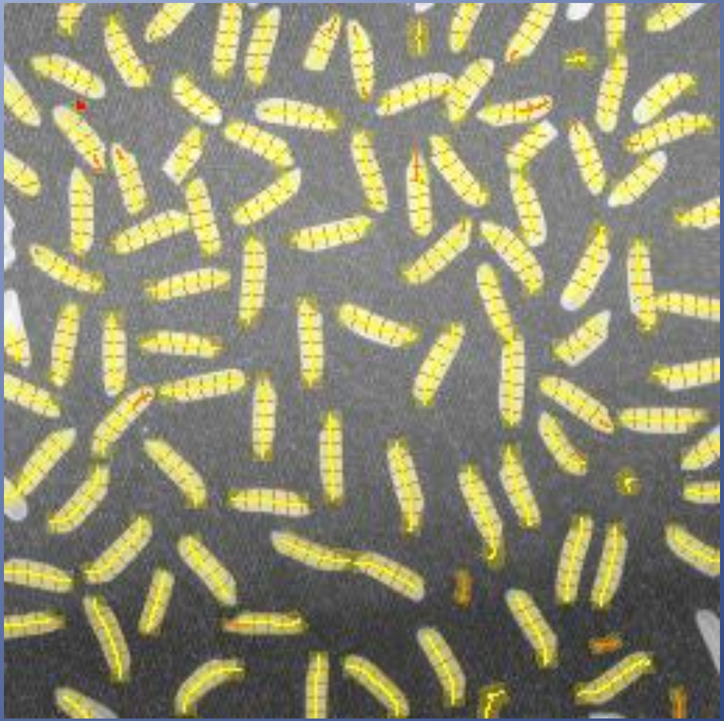


Synthetic data

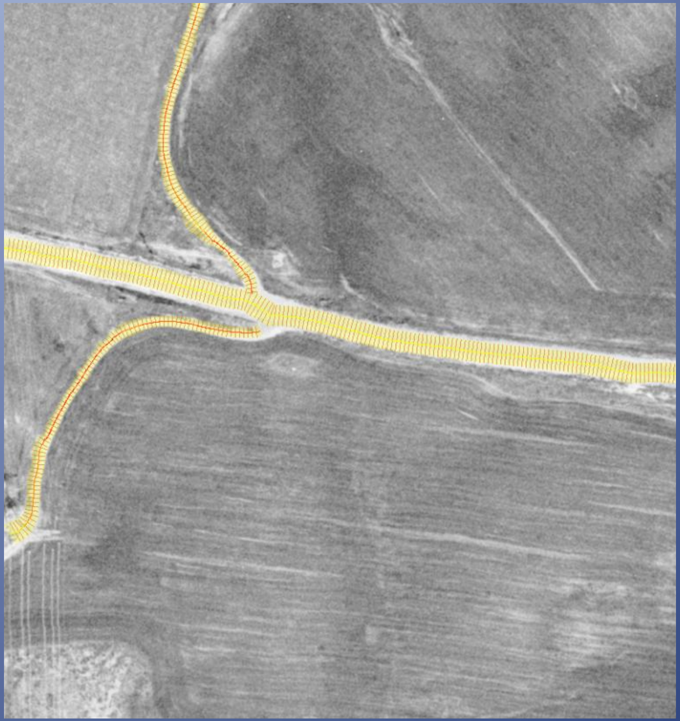


Synthetic data





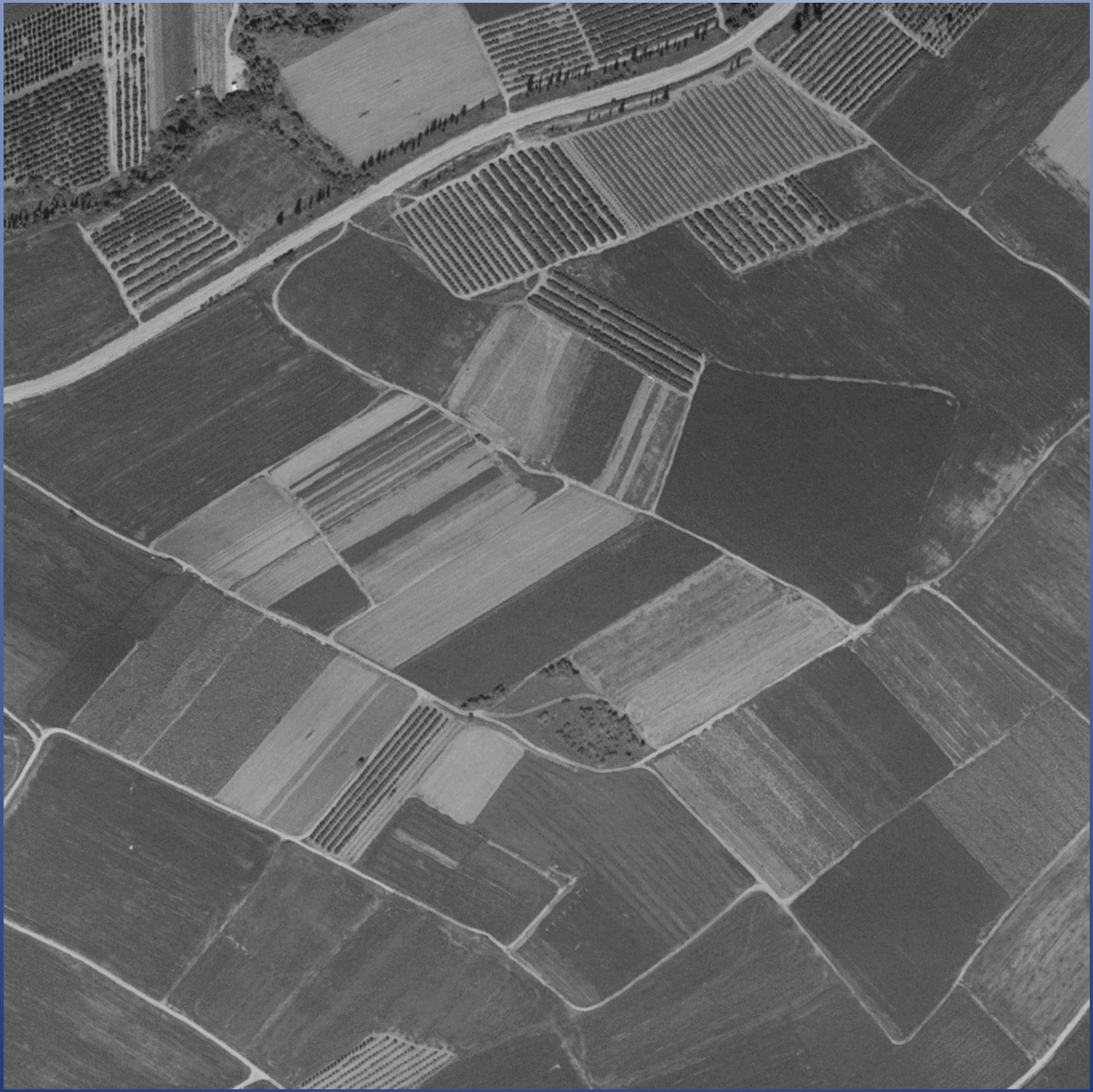
Aerial photo



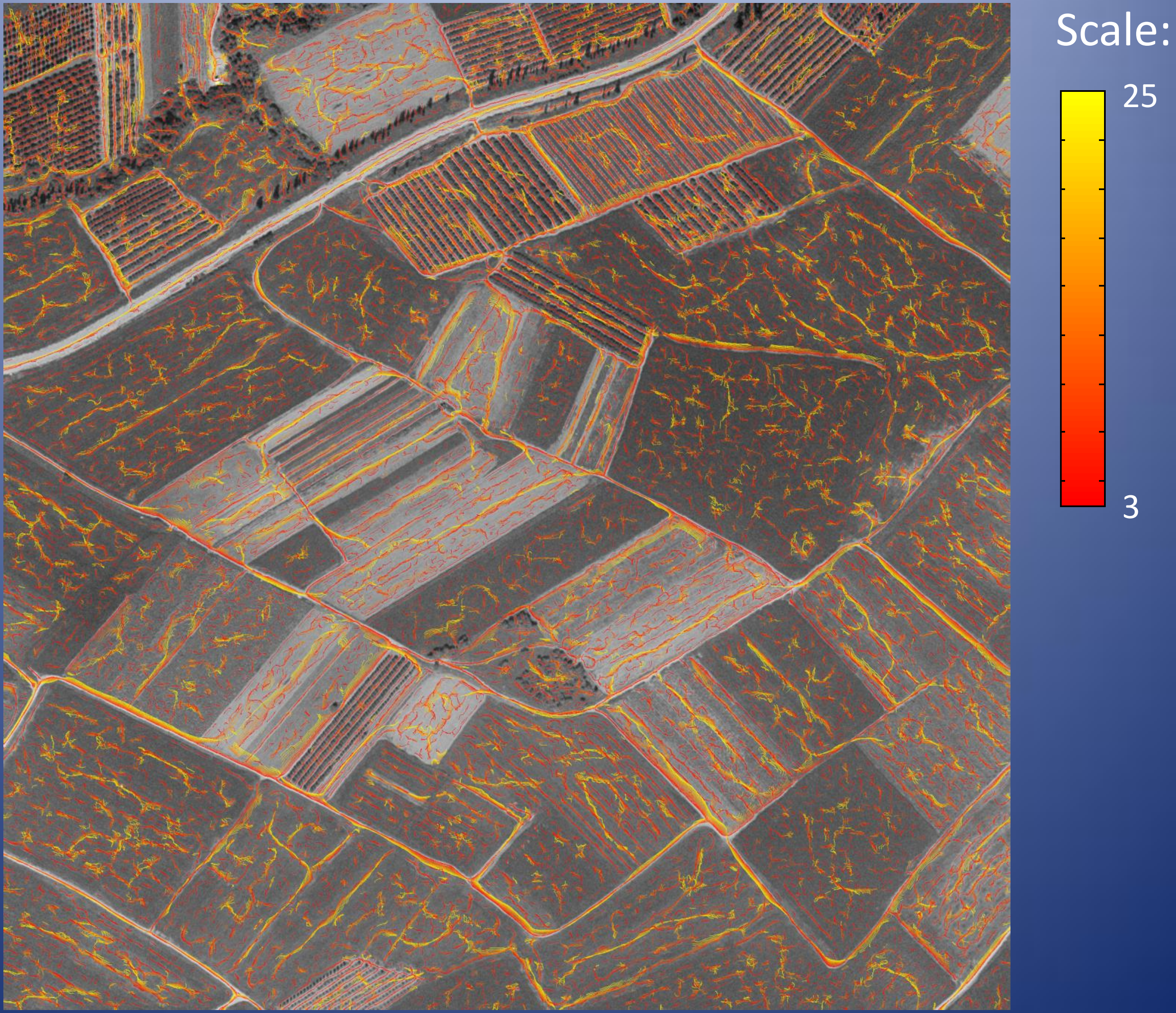
Different phases of the algorithm

Here is an example of a typical aerial photo:

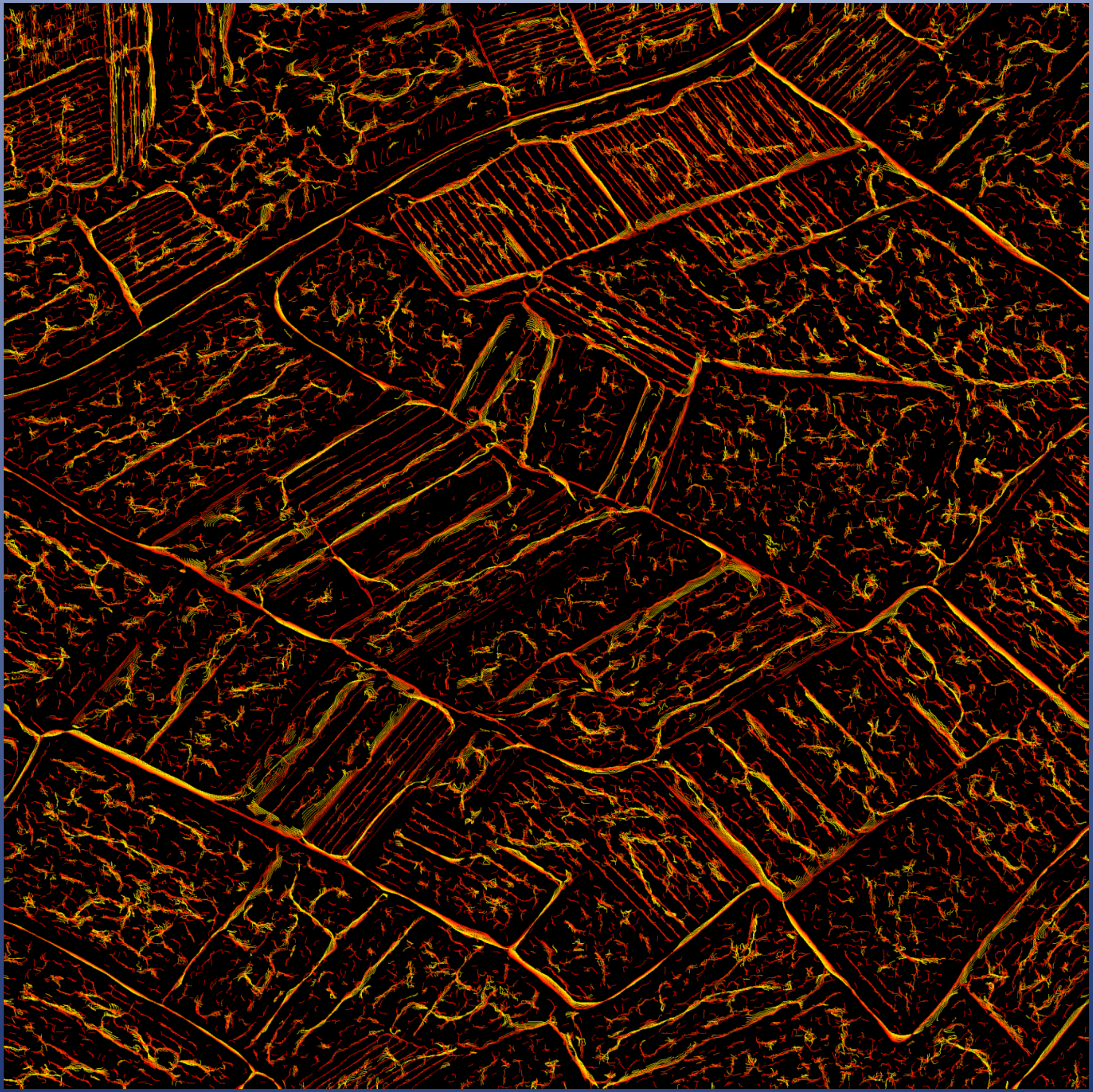
2048x2048
pixels

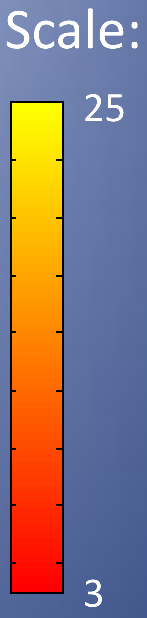


All the ridge points from all scales together:

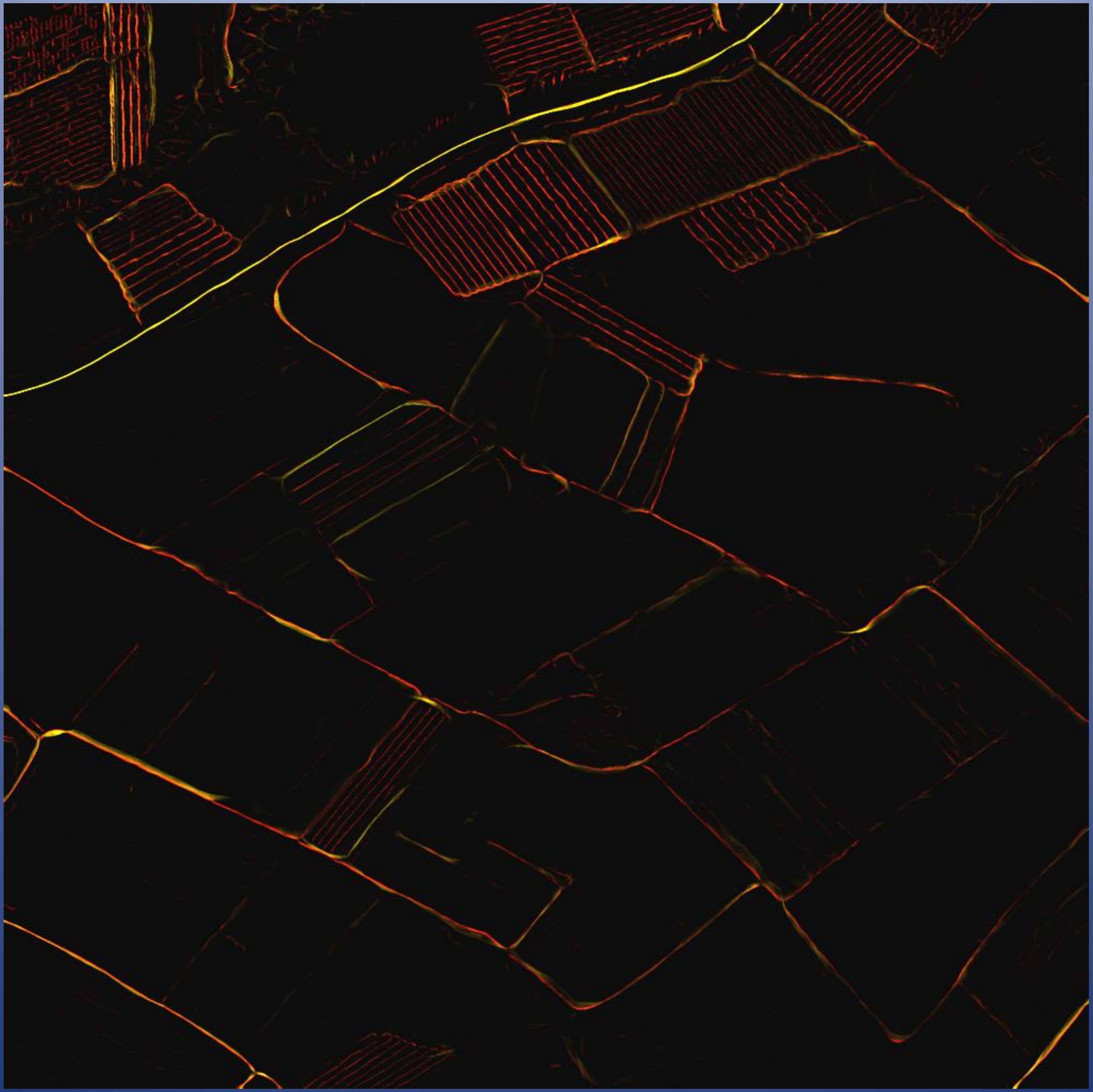


All the ridge points from all scales together:

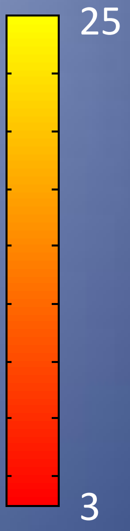




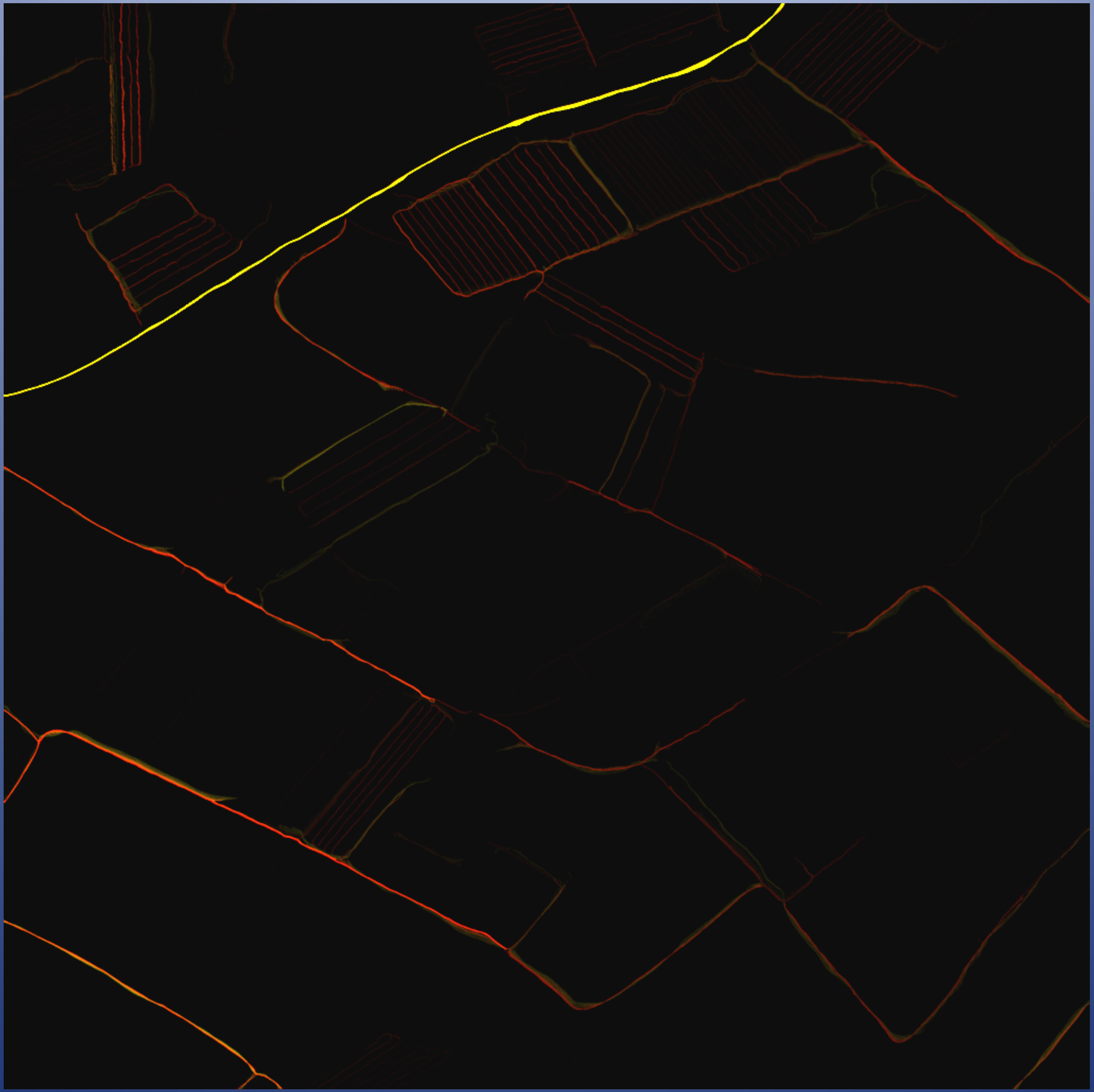
Candidate points (brightness by point strength)

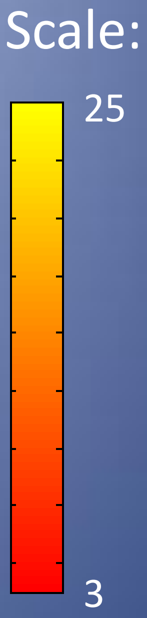


Scale:

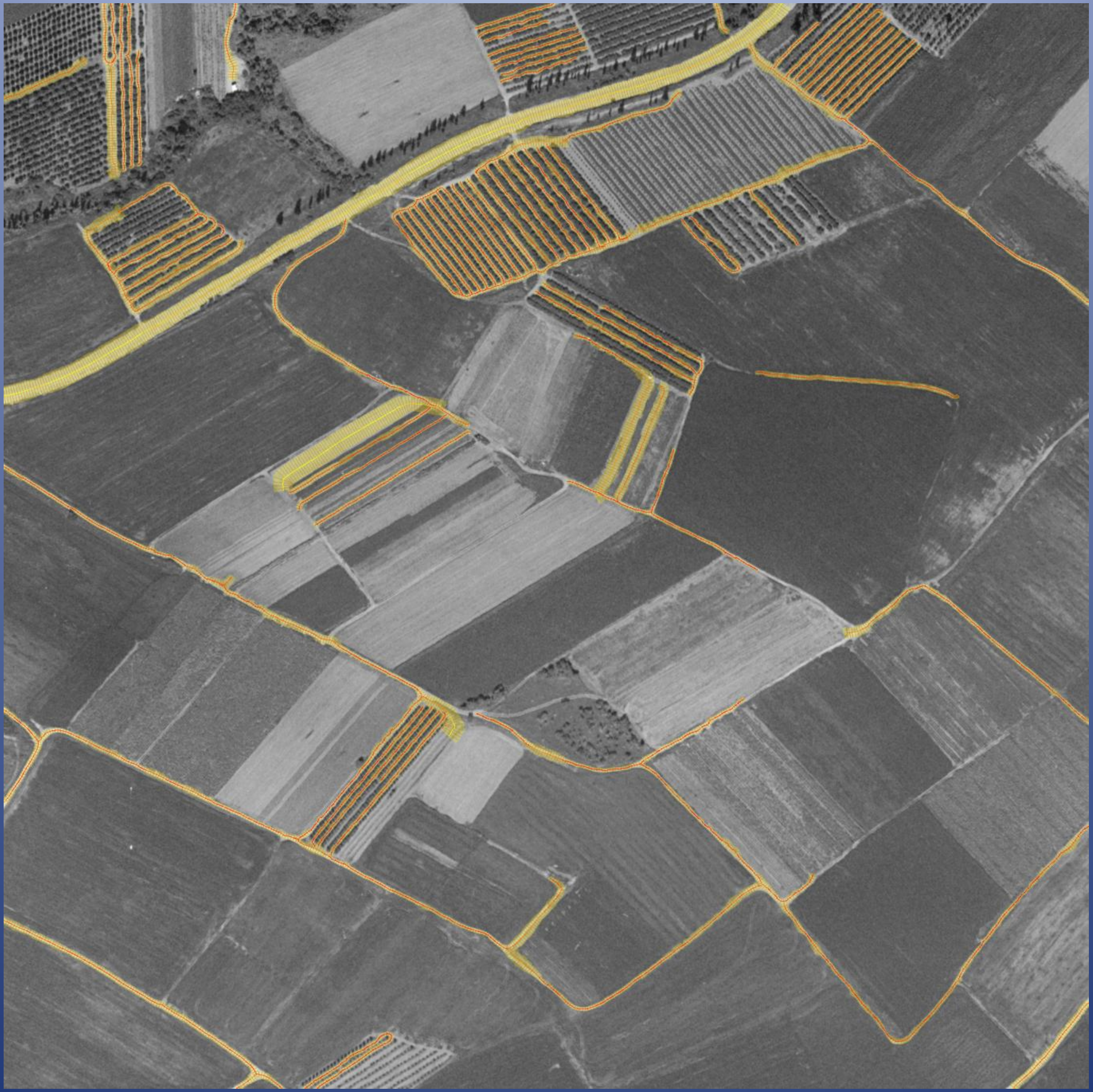


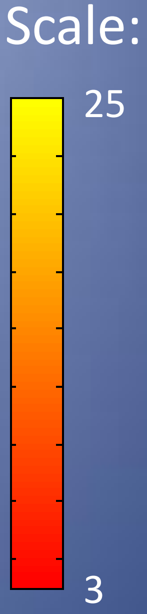
Candidate points (brightness by connected component grade)



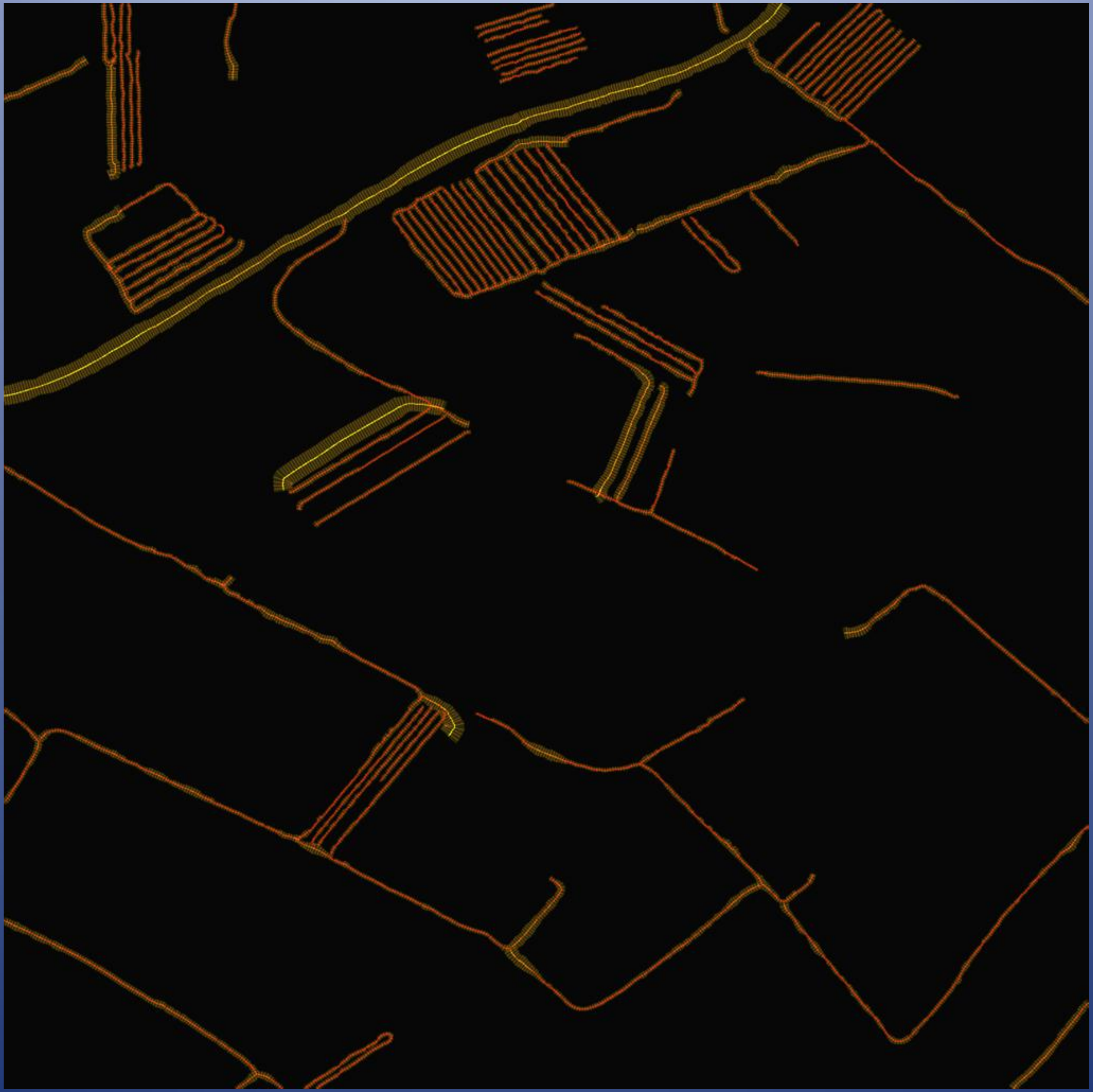


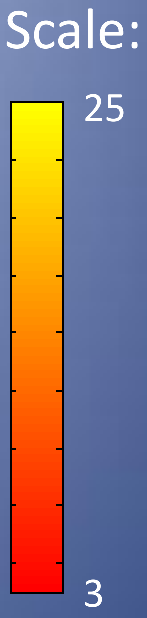
100 best ridges



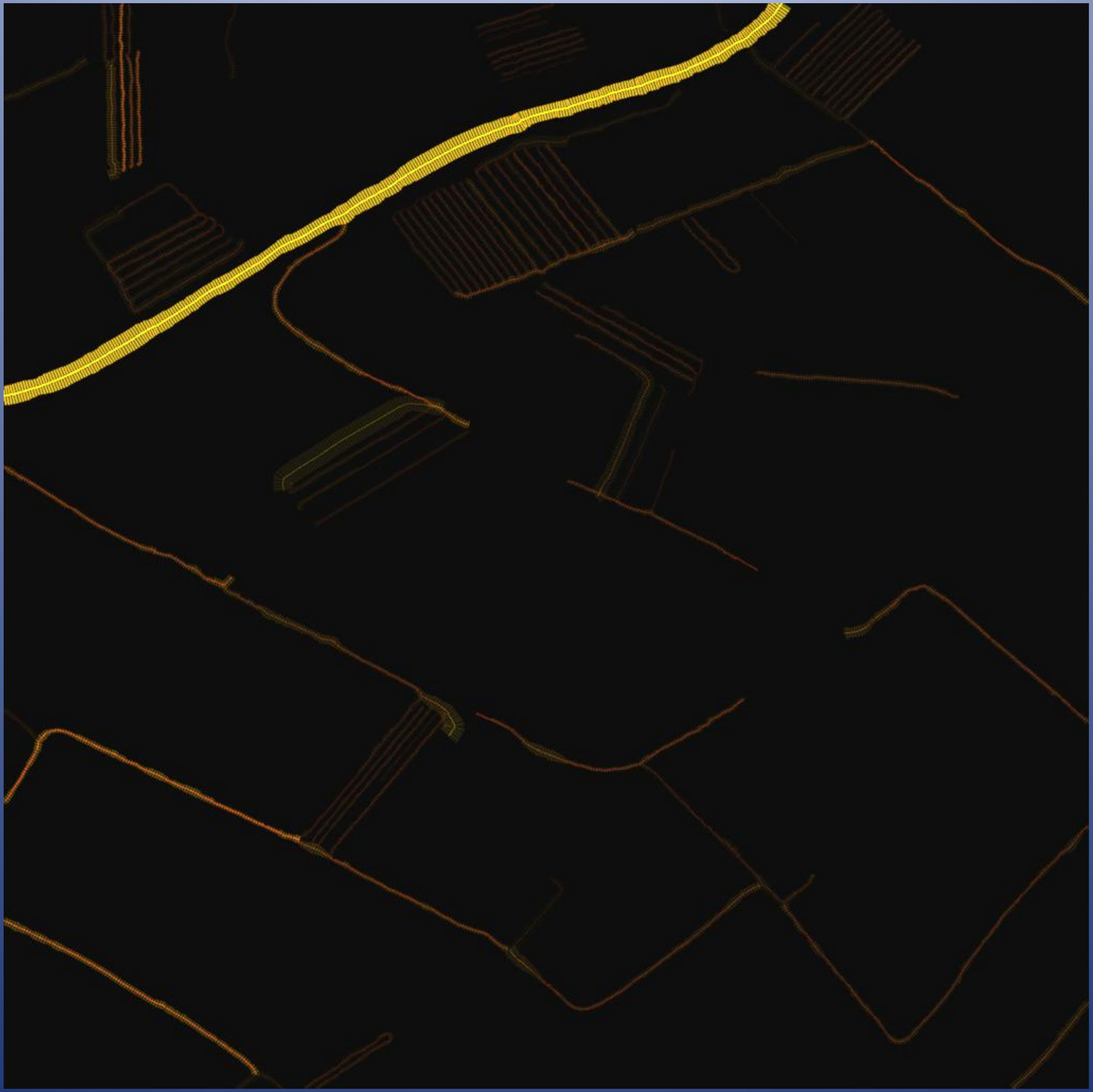


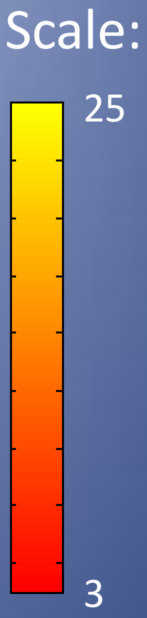
100 best ridges



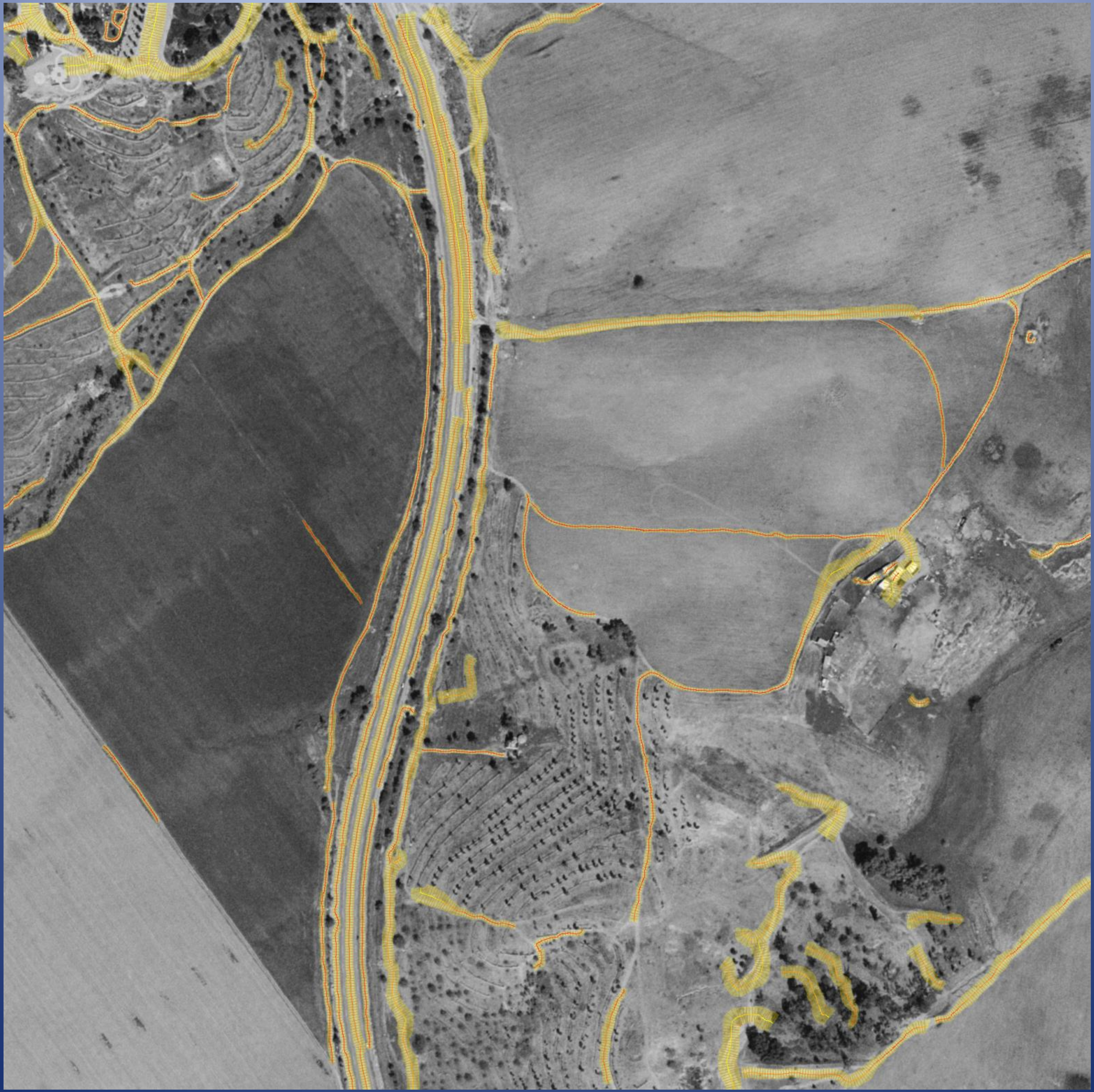


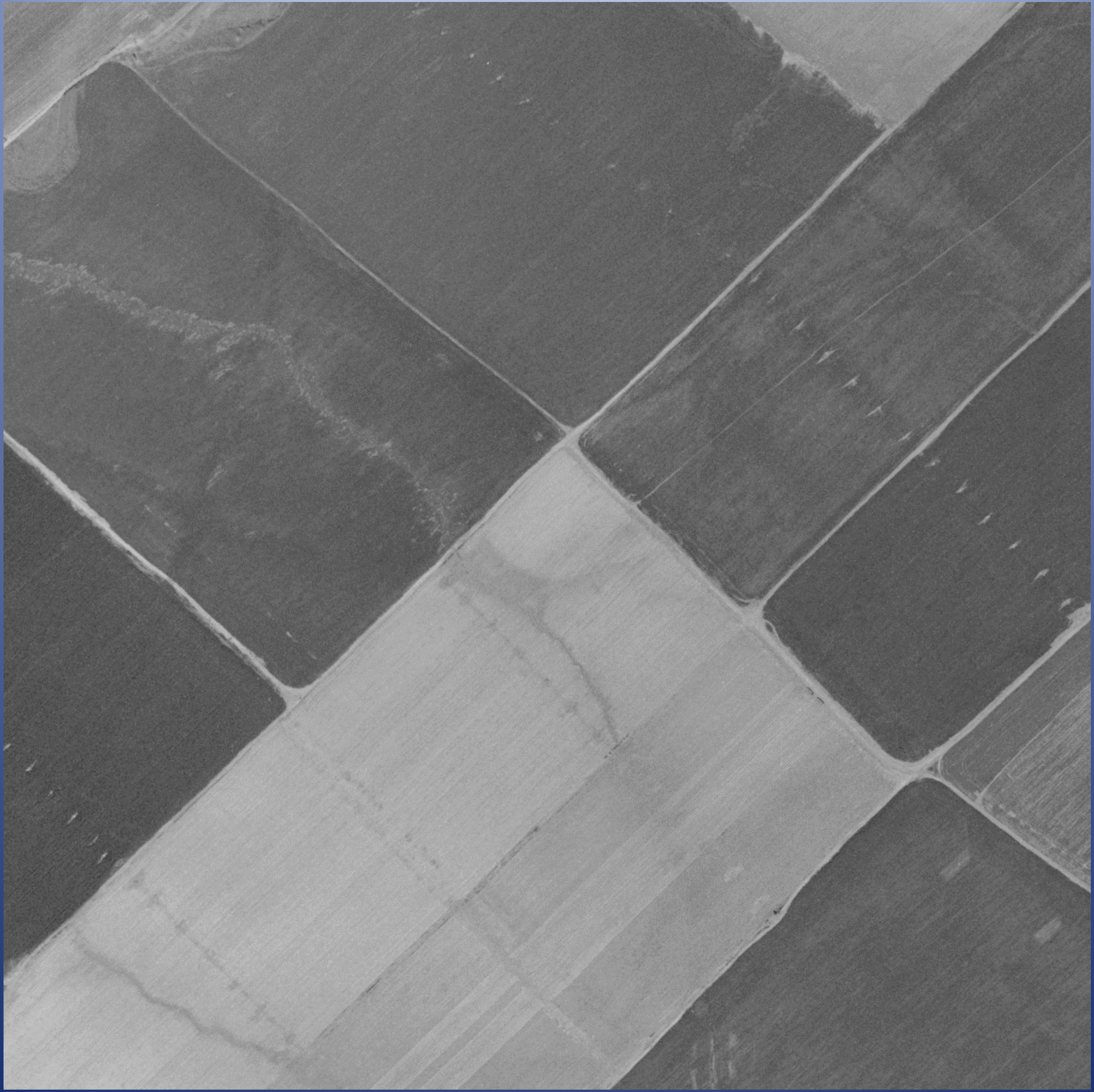
100 best ridges. Brightness by ridge grade.

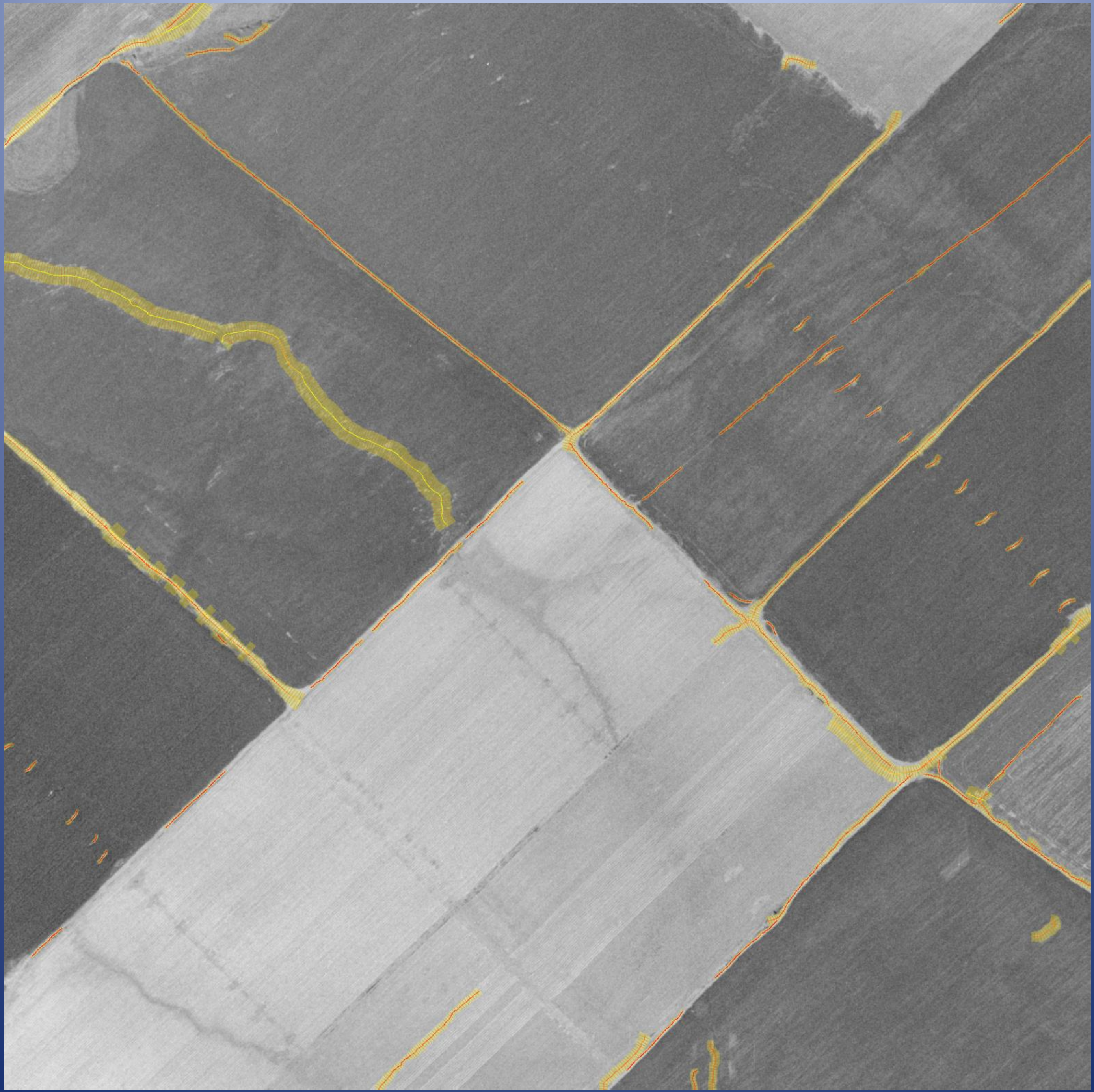


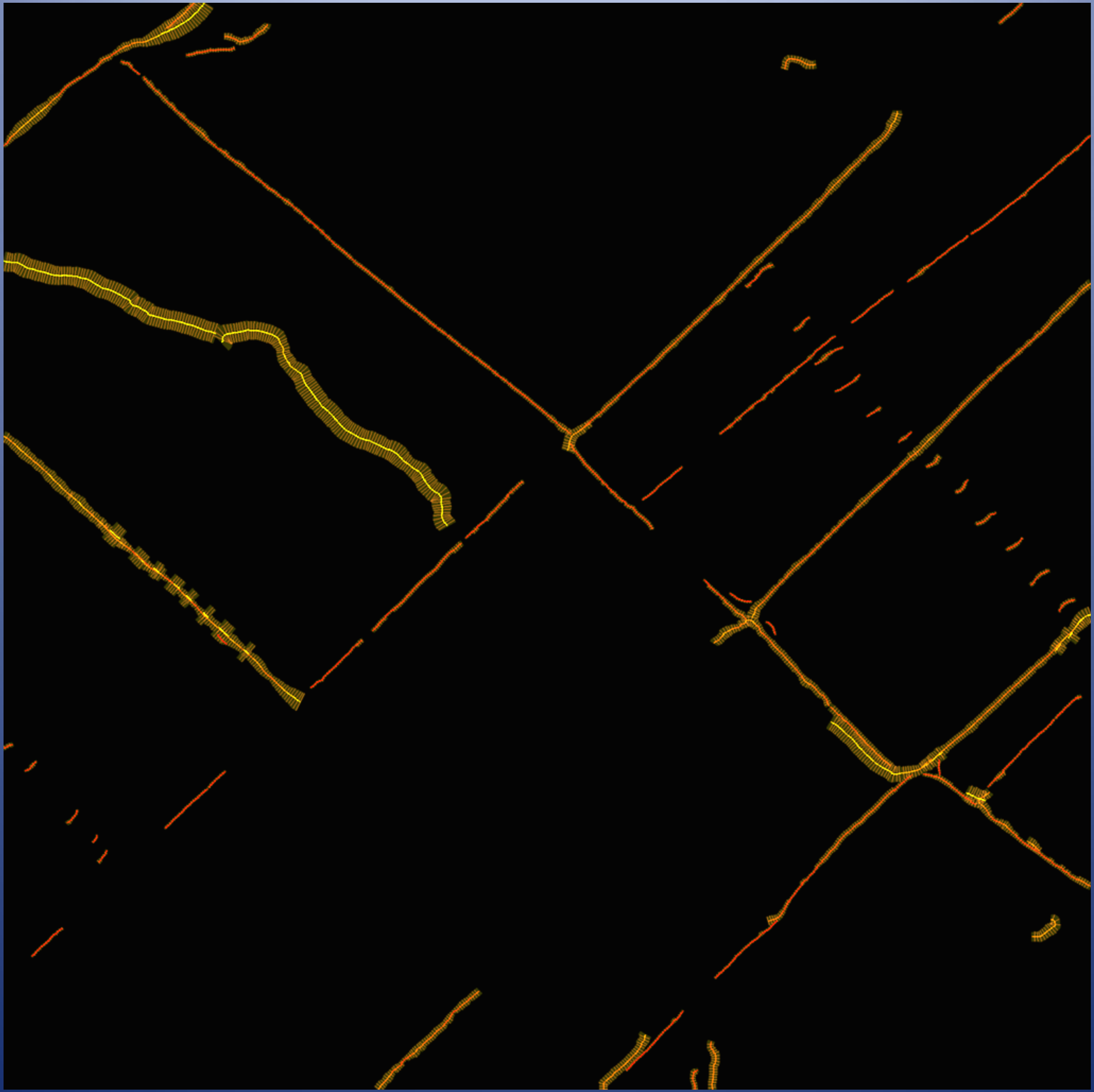


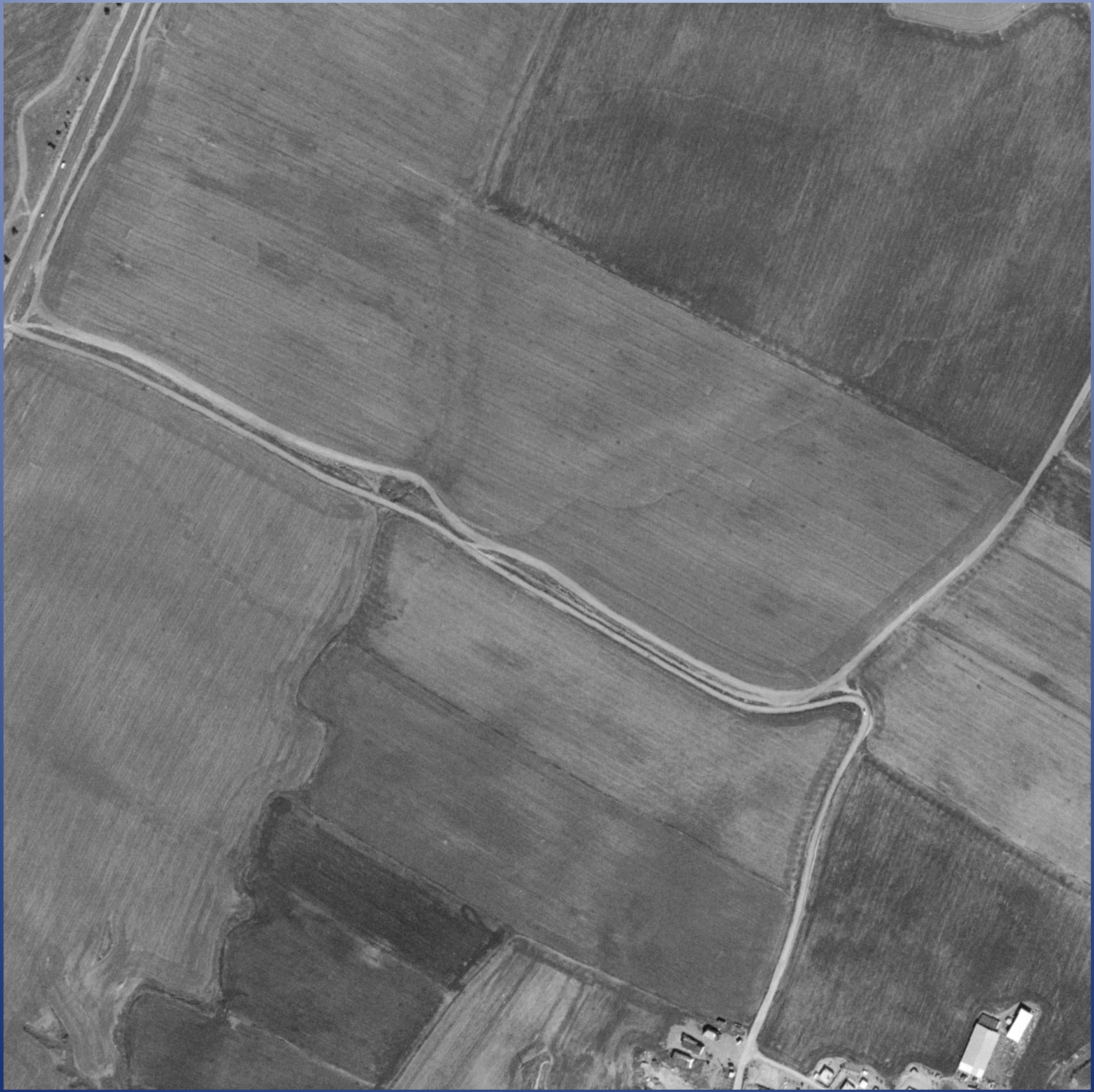
More examples

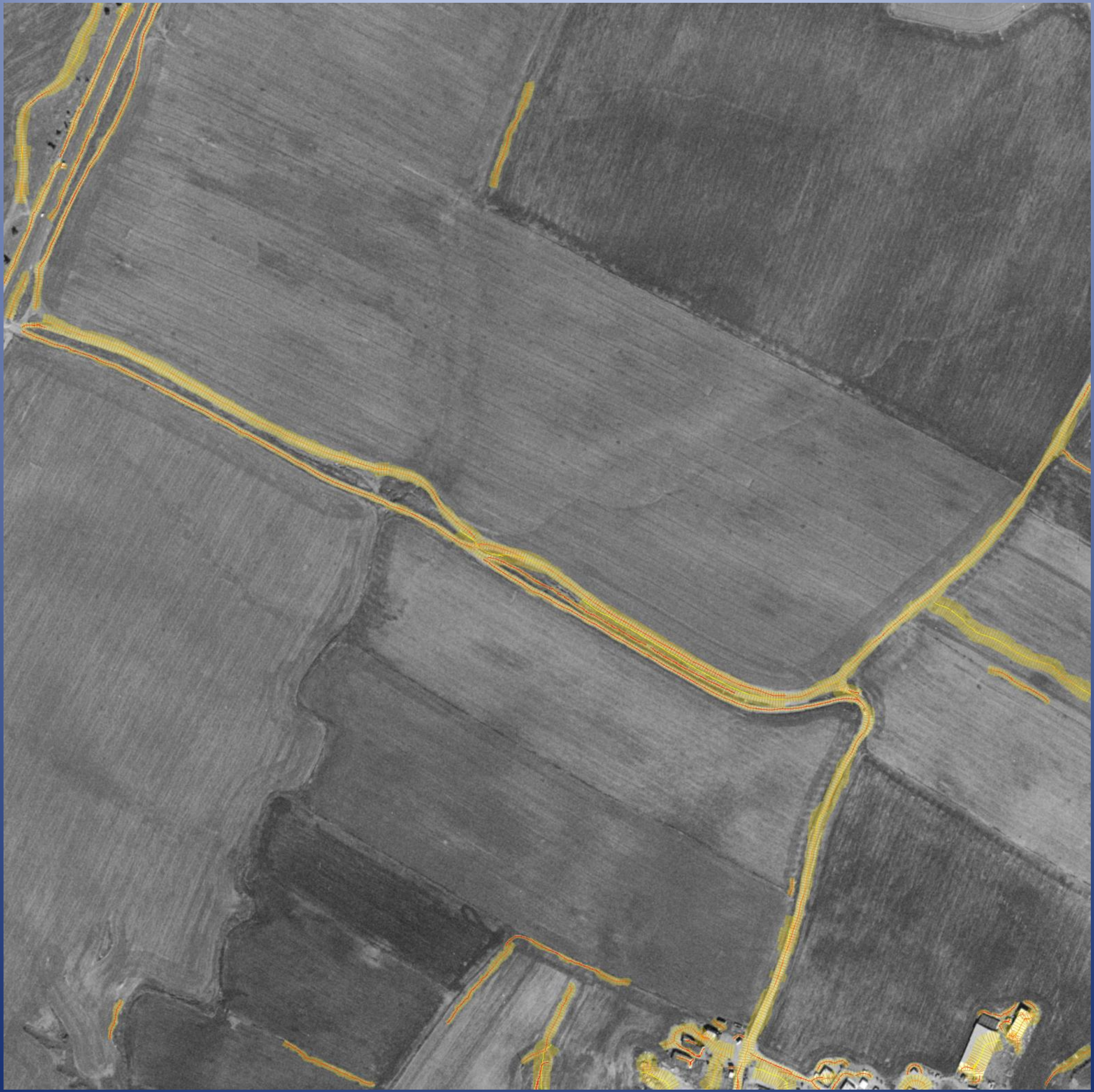


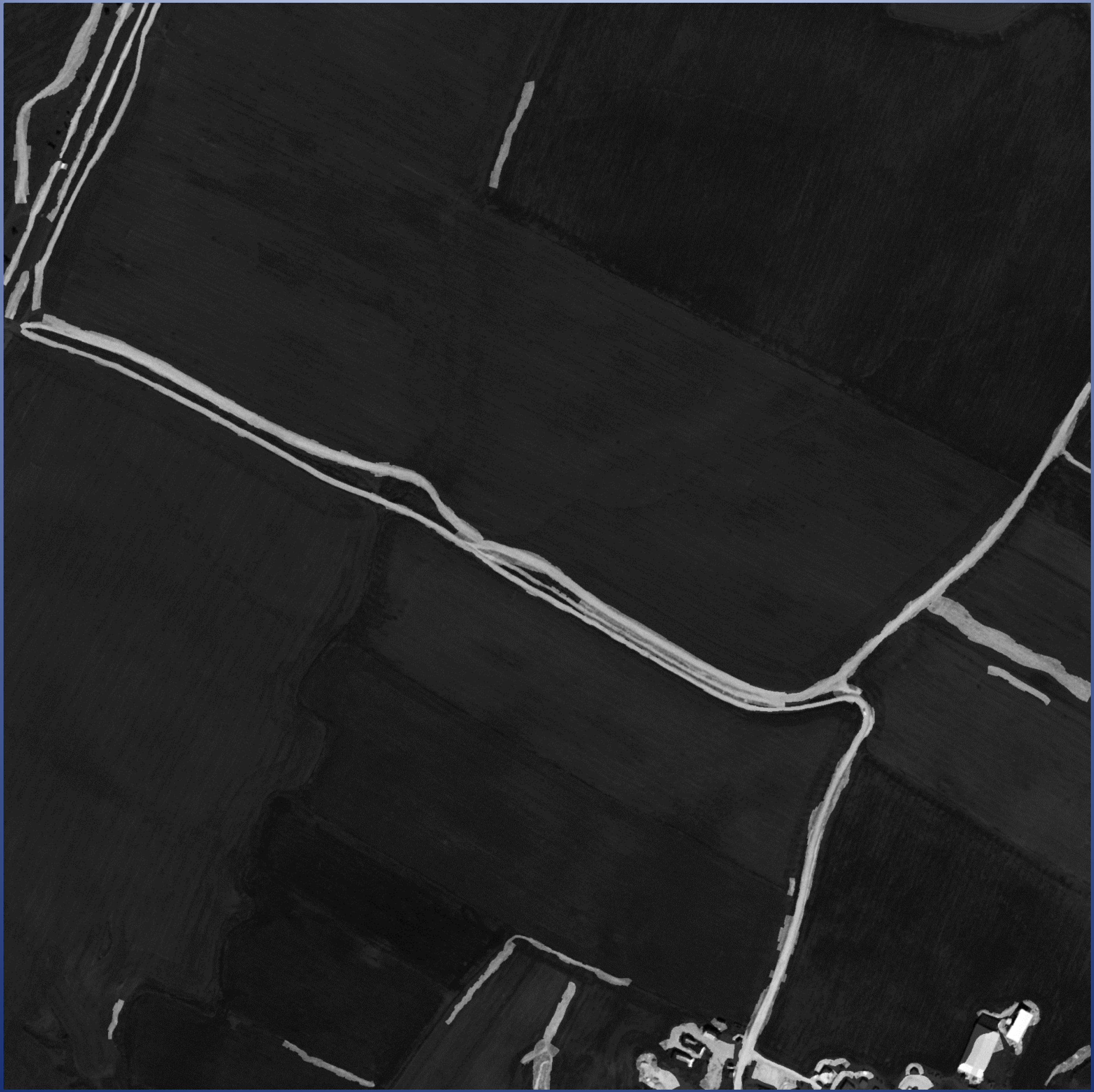


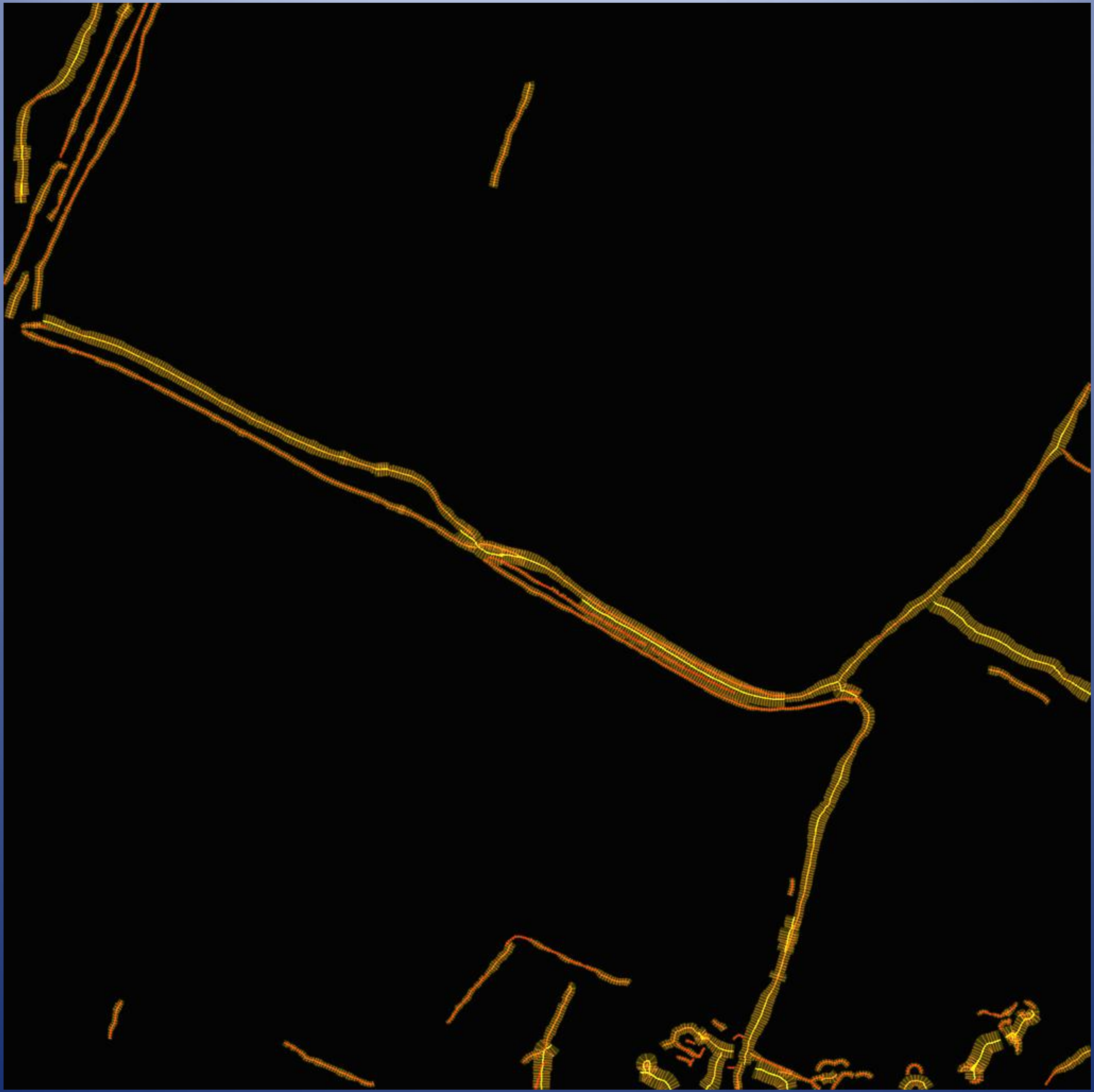




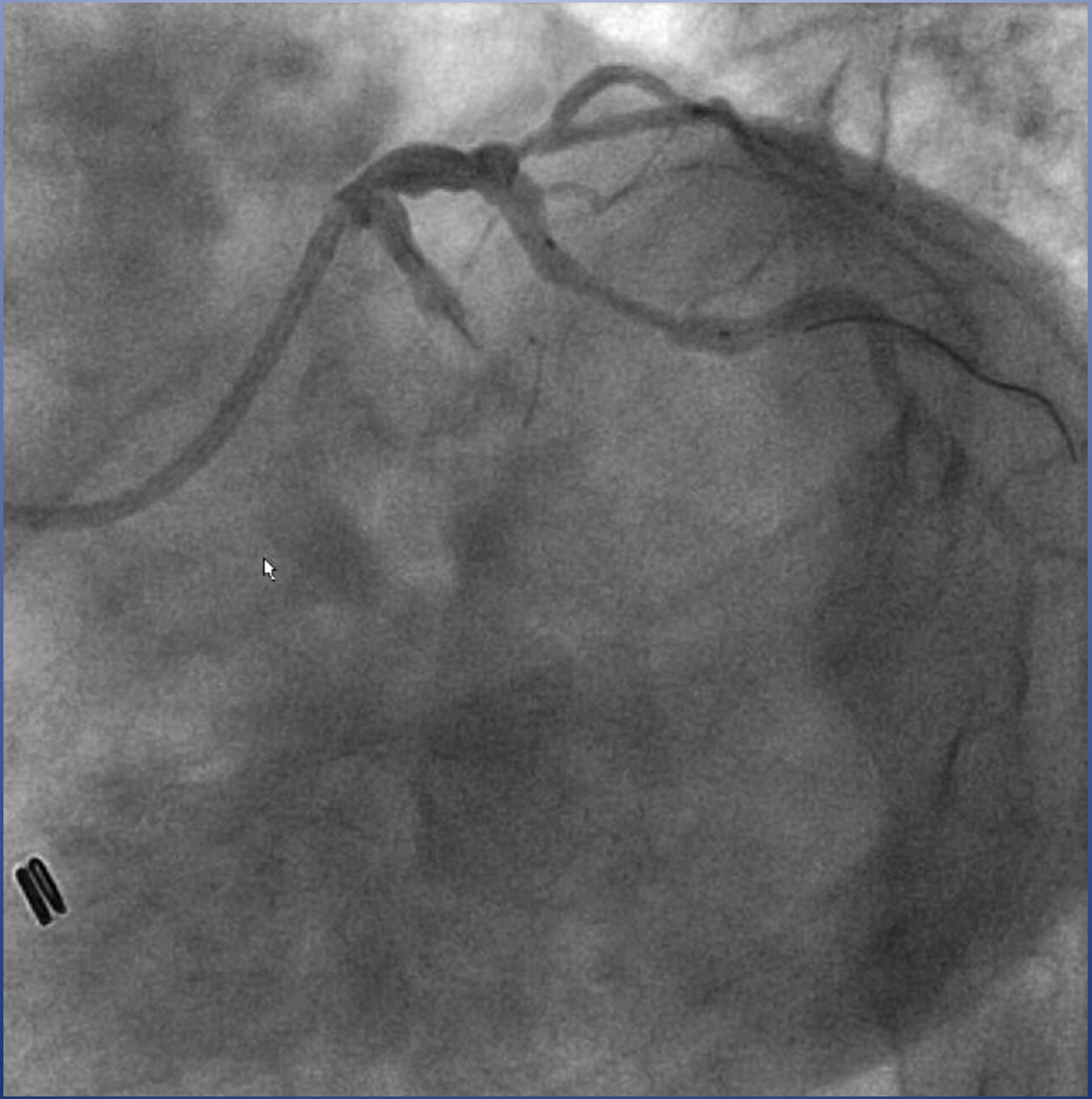




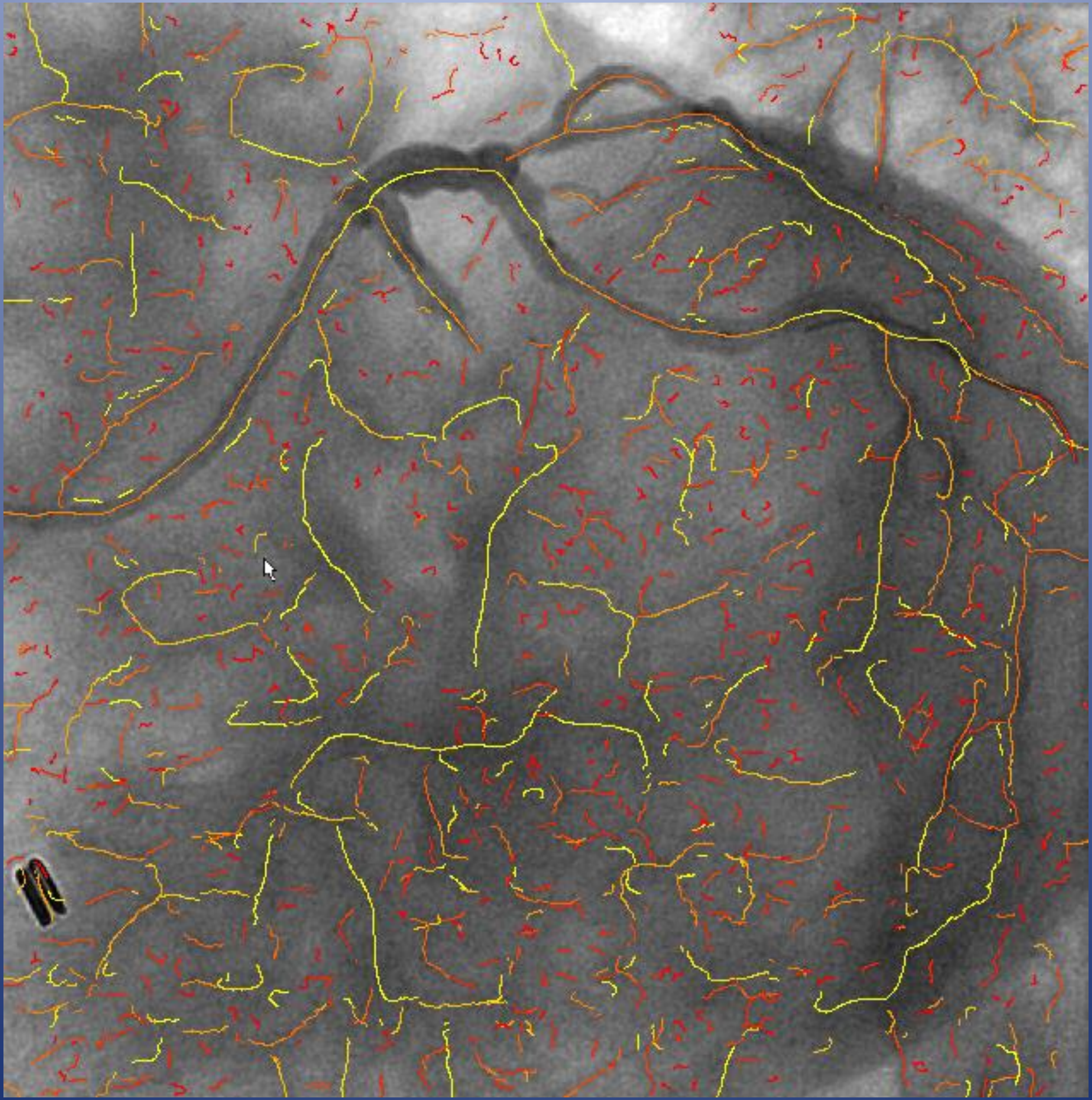




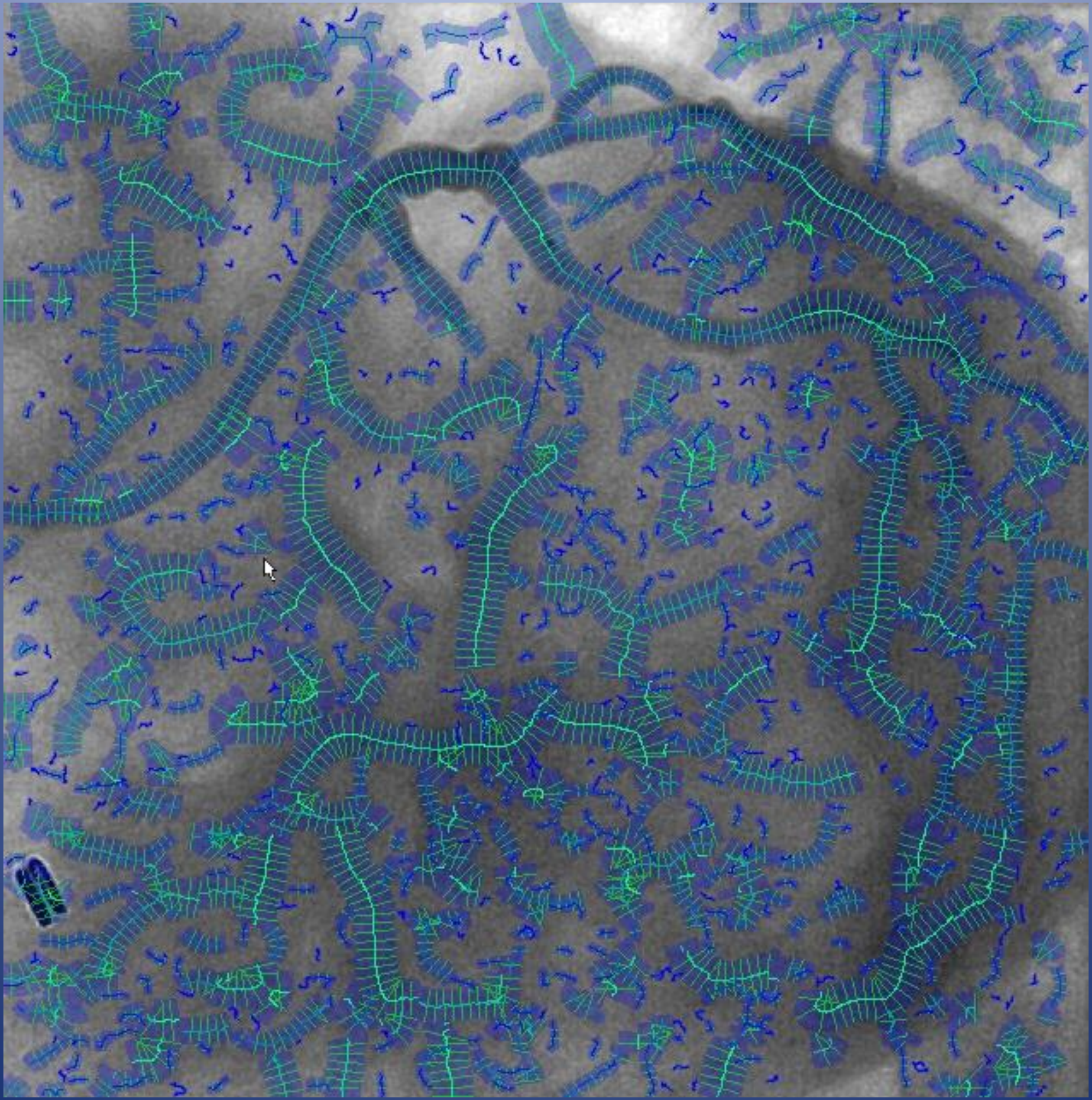
Blood vessels



Blood vessels

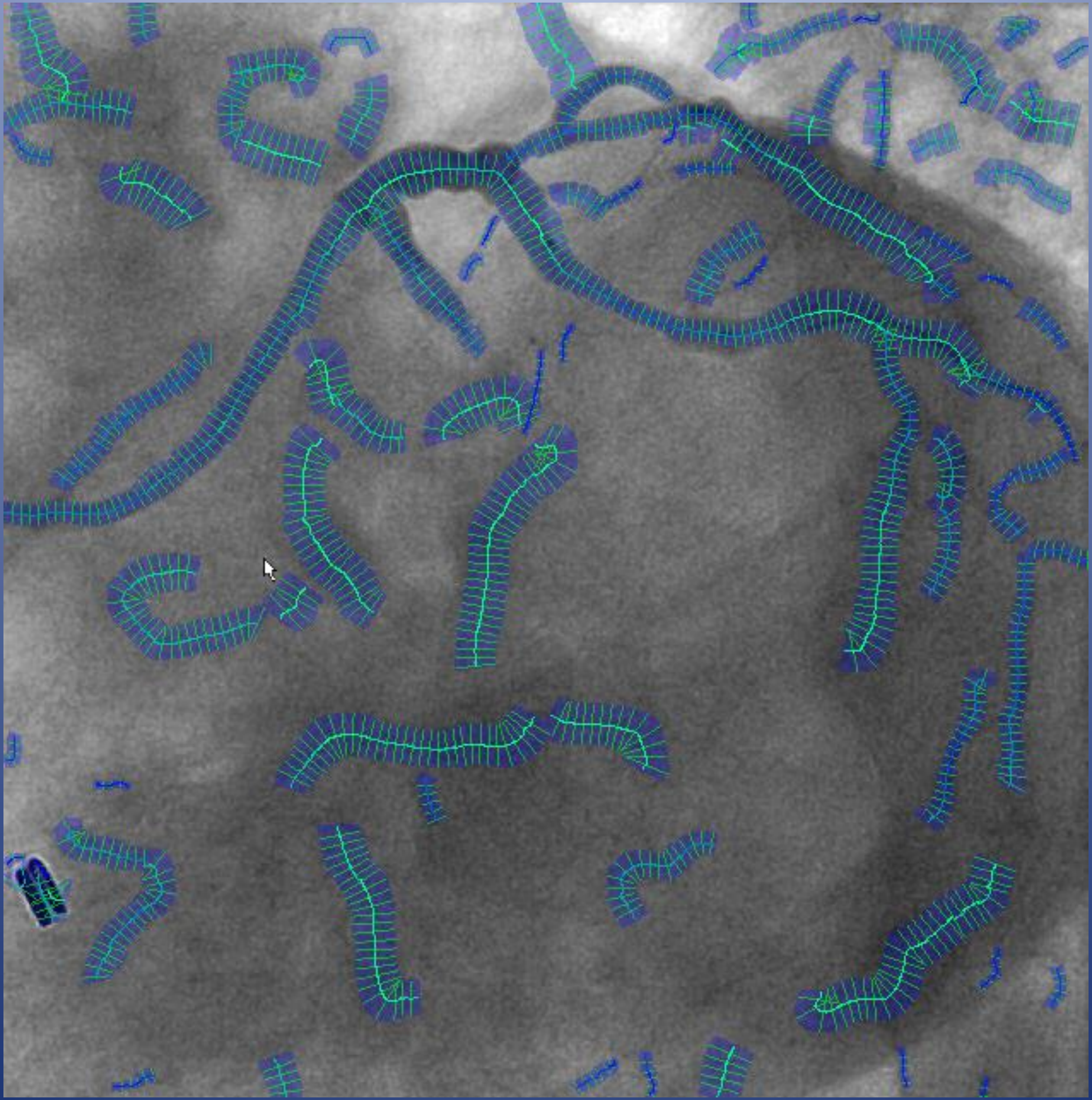


Blood vessels



Blood vessels

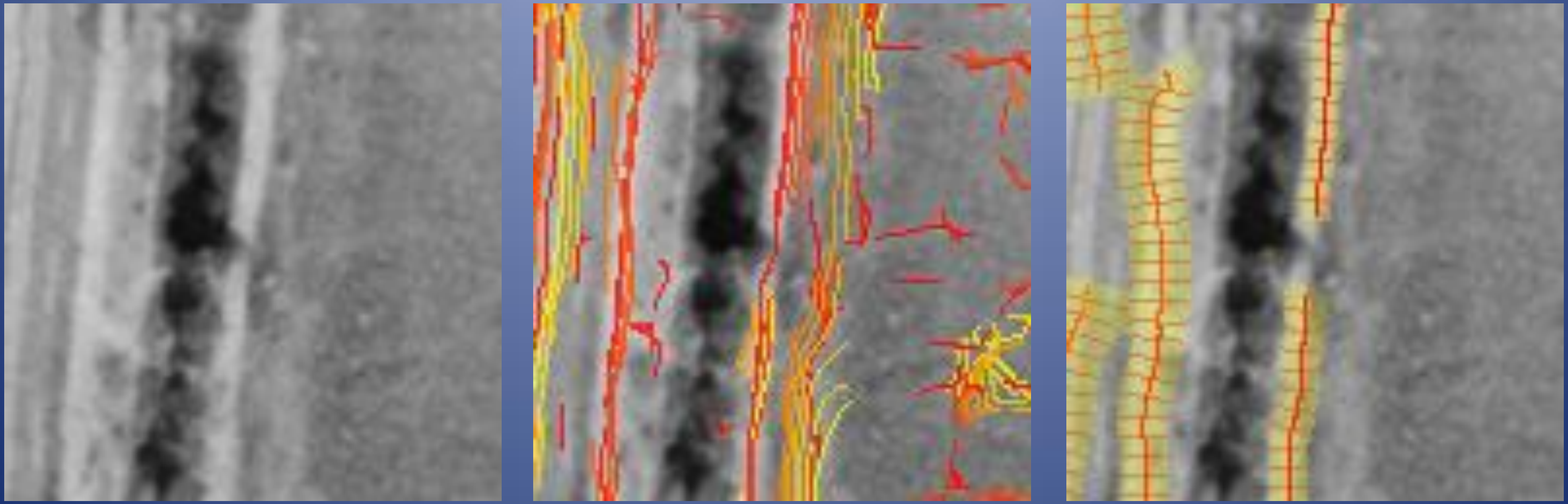
100
best
ridges



Algorithm pitfalls

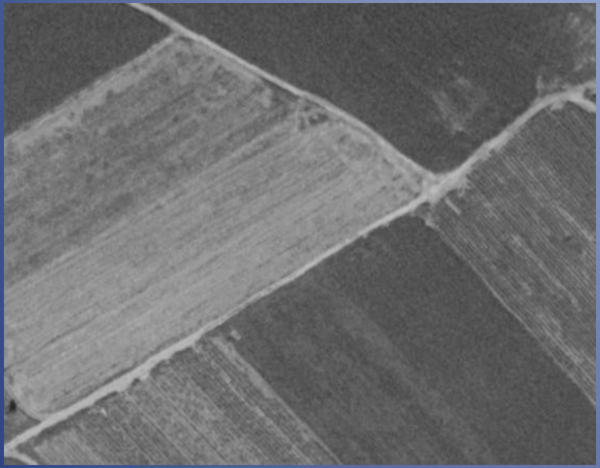
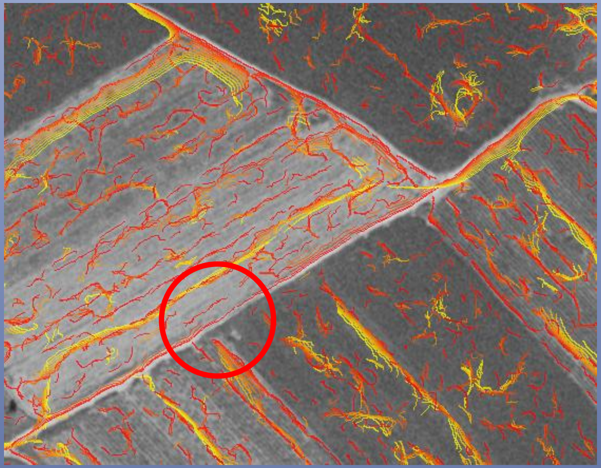
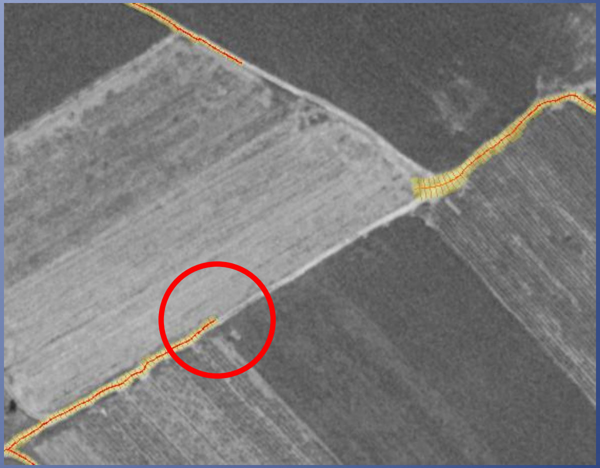
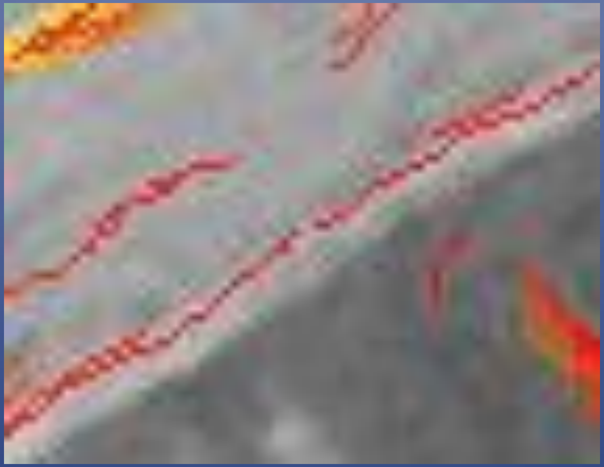
Discontinuities

Small objects that lie within the road essentially prevent it from being a ridge.



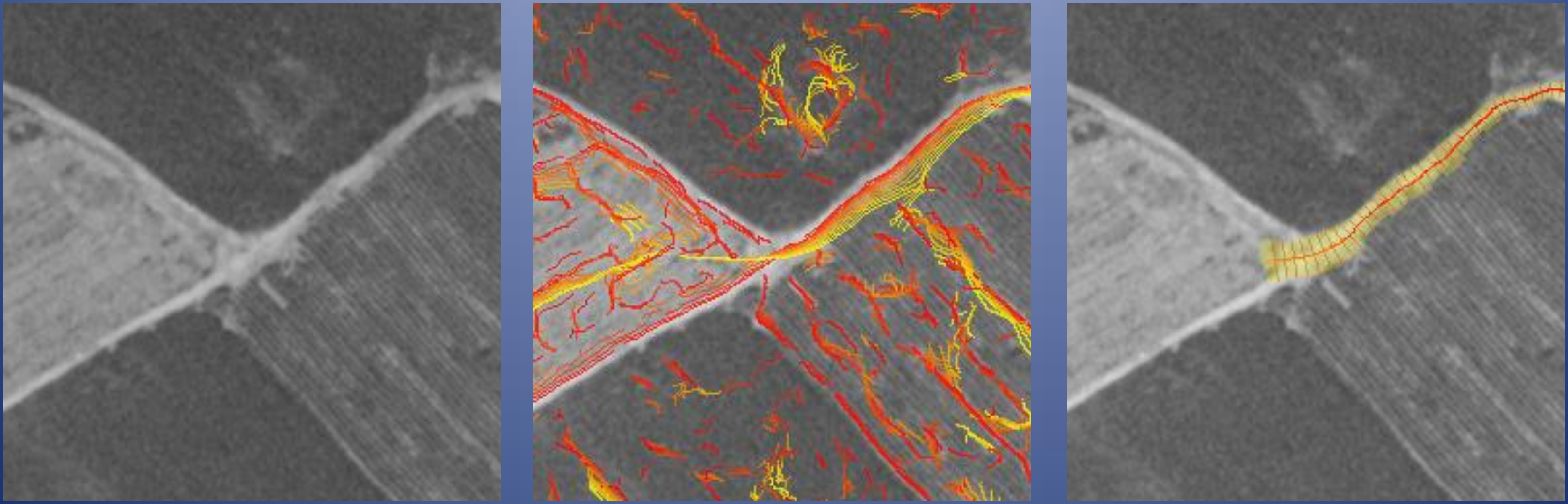
Discontinuities

Bright ridges outside of the road also interfere with the ridge

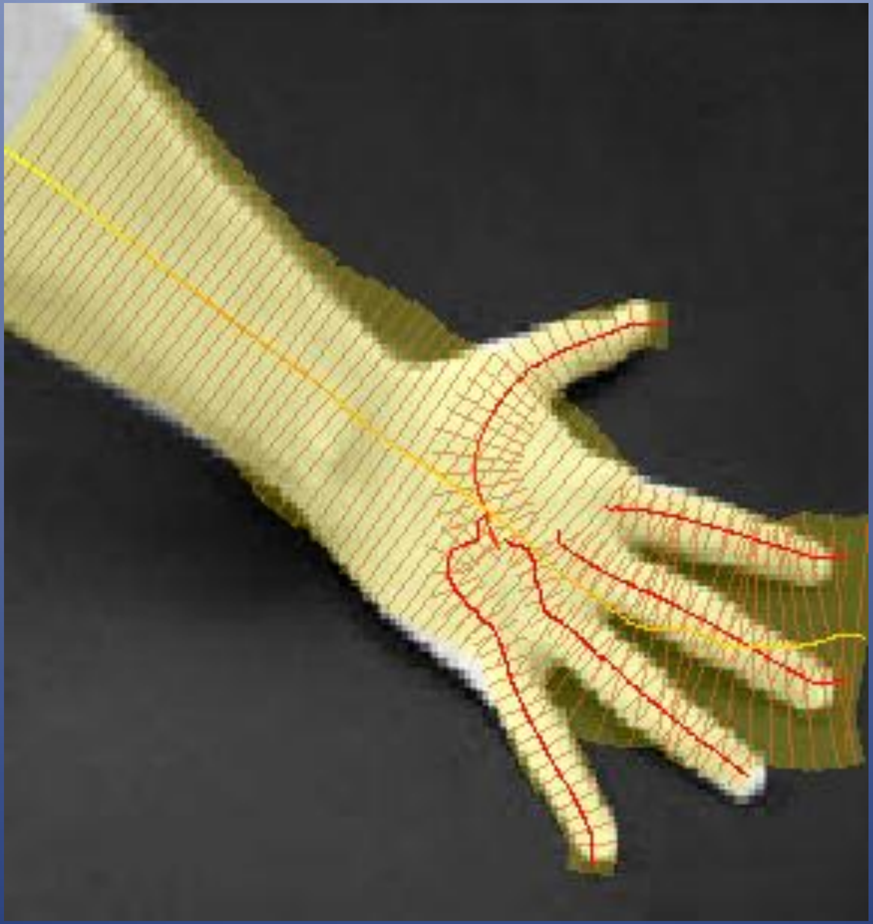
Drifting in scale space

Ridges which are related tend to drift when changing scale. When changing scale, we choose heuristically which ridge to continue tracking from. In this particular example, it caused us to continue tracking along the wrong ridge, which has lead us to a dead end.



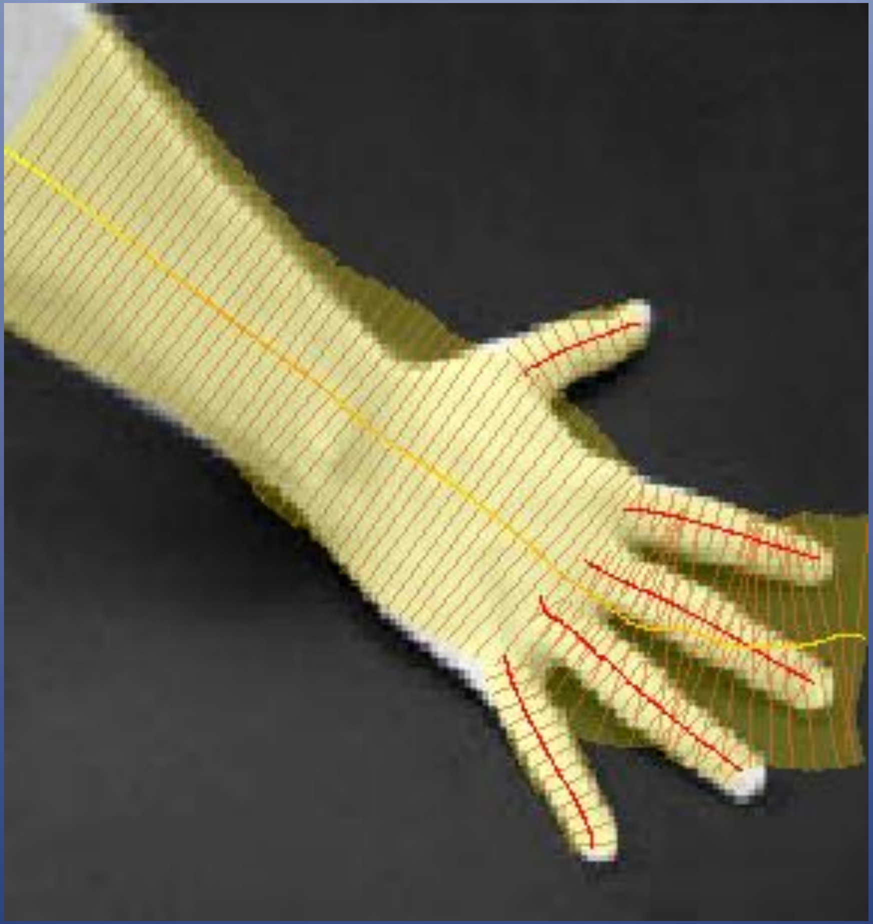
Fake ridge continuations

While tracking the ridge, elongated regions with small height difference relative to their surrounding pretend to be a ridge. Even though these points get a low grade, they might be attached to real ridges with a high grade. The algorithm starts from the good region and continues to the bad one.



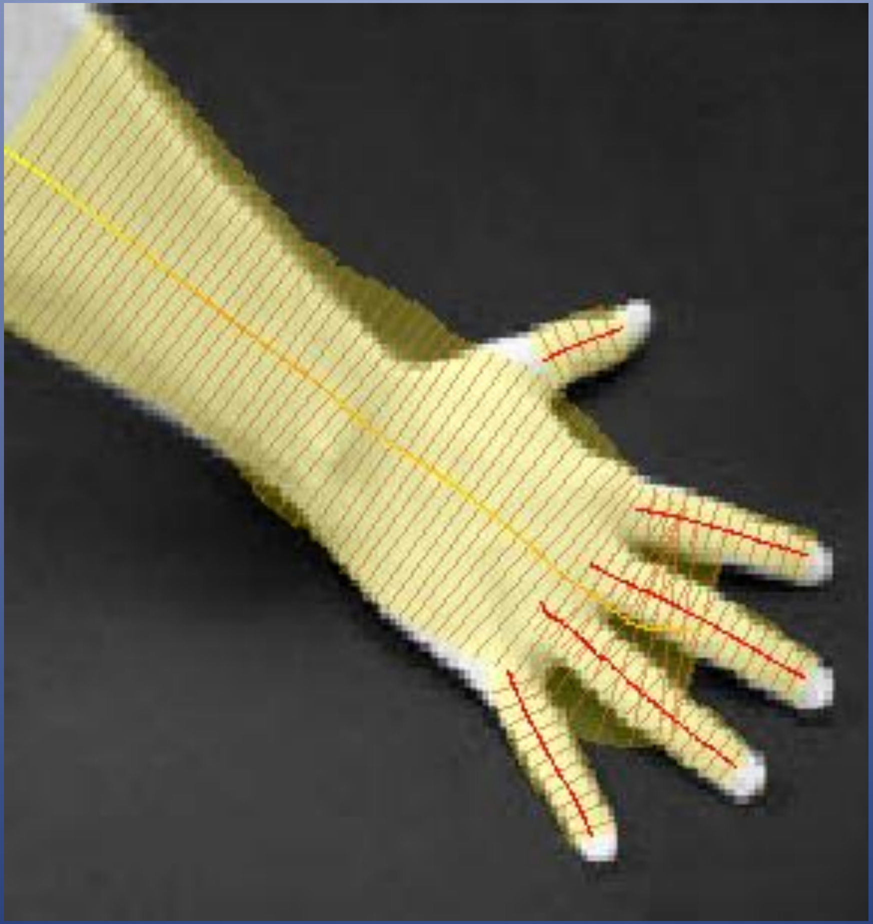
Fake ridge continuations

Solution: We have a criteria that compares the height resulting in stepping forward and stepping aside, which is scale-invariant, and can be applied with different levels of strictness.



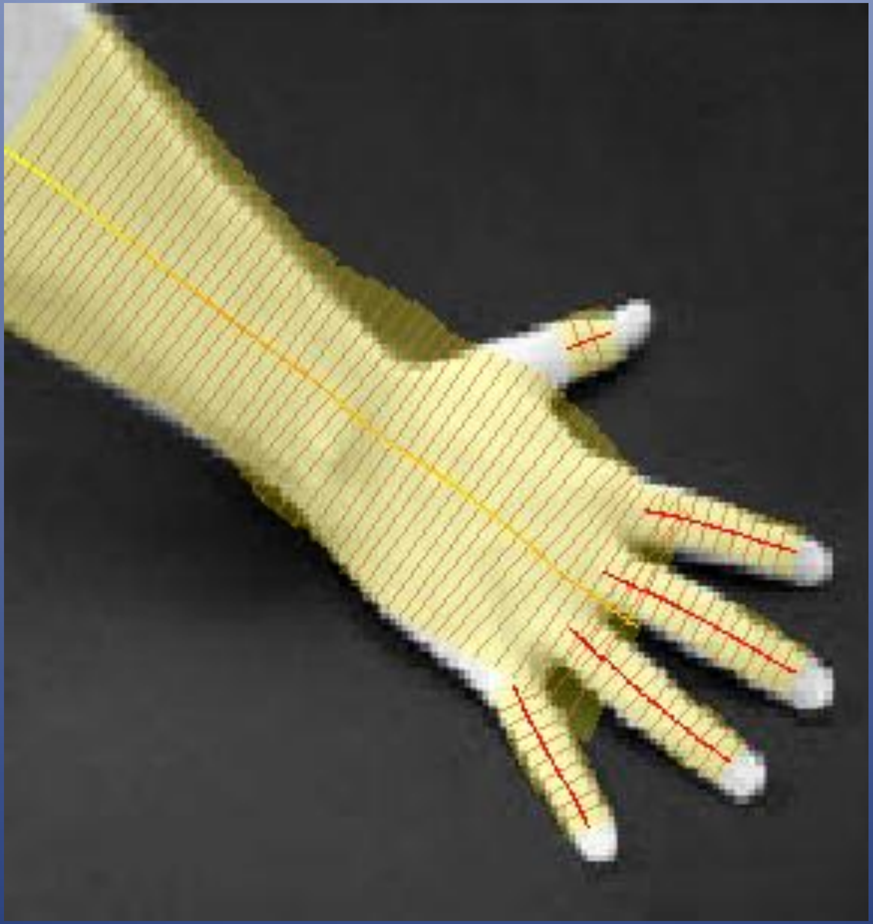
Fake ridge continuations

However, no level if strictness is perfect for the entire image.



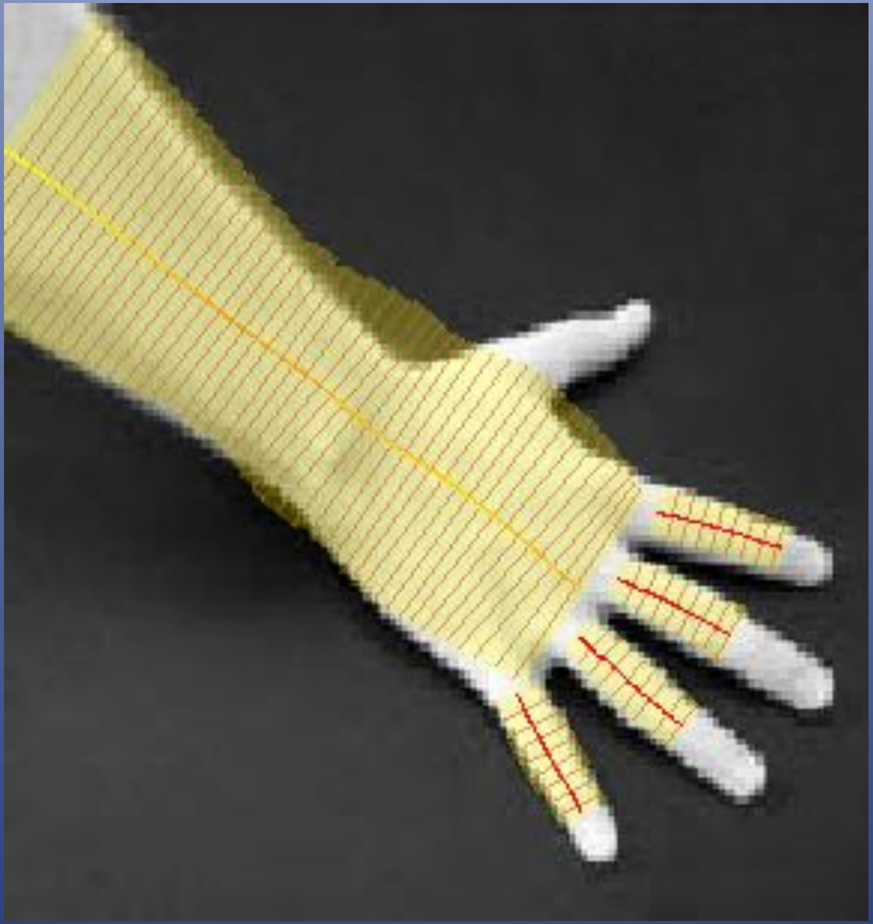
Fake ridge continuations

However, no level if strictness is perfect for the entire image.



Fake ridge continuations

However, no level if strictness is perfect for the entire image.



Algorithm running time

Algorithm running time

Theoretically the time complexity is:

$$O\left(\text{number of voxels} + \log^*\left(\text{number of ridge points}\right)\right)$$

The algorithm performs a constant number of passes on each voxel during the preprocessing stage (except for using the Union-Find) and a constant number of passes on each zero crossing point during the tracking stage.

Practically (in seconds):

Resolution\ nScales	256x256	641x635	1024x1024	2048x2048
9	1	5.8	17.5	93.3
12	1.35	7.2	21.4	143.4
17	1.85	9.7	30.5	195.1

These results were obtained using this hardware: Intel Core2 Duo 2.20Ghz, 4Gb RAM, running on Linux (Ubuntu 11.04).

Algorithm running time

Let us compare the scale-space derivative calculation speed with our method vs. Lindeberg's method (with Gaussian derivatives and convolution).

In the following table we calculated the 5 derivatives in different scales with both methods and compared their time in seconds :
Black – our result .
Red – Lindebergs.

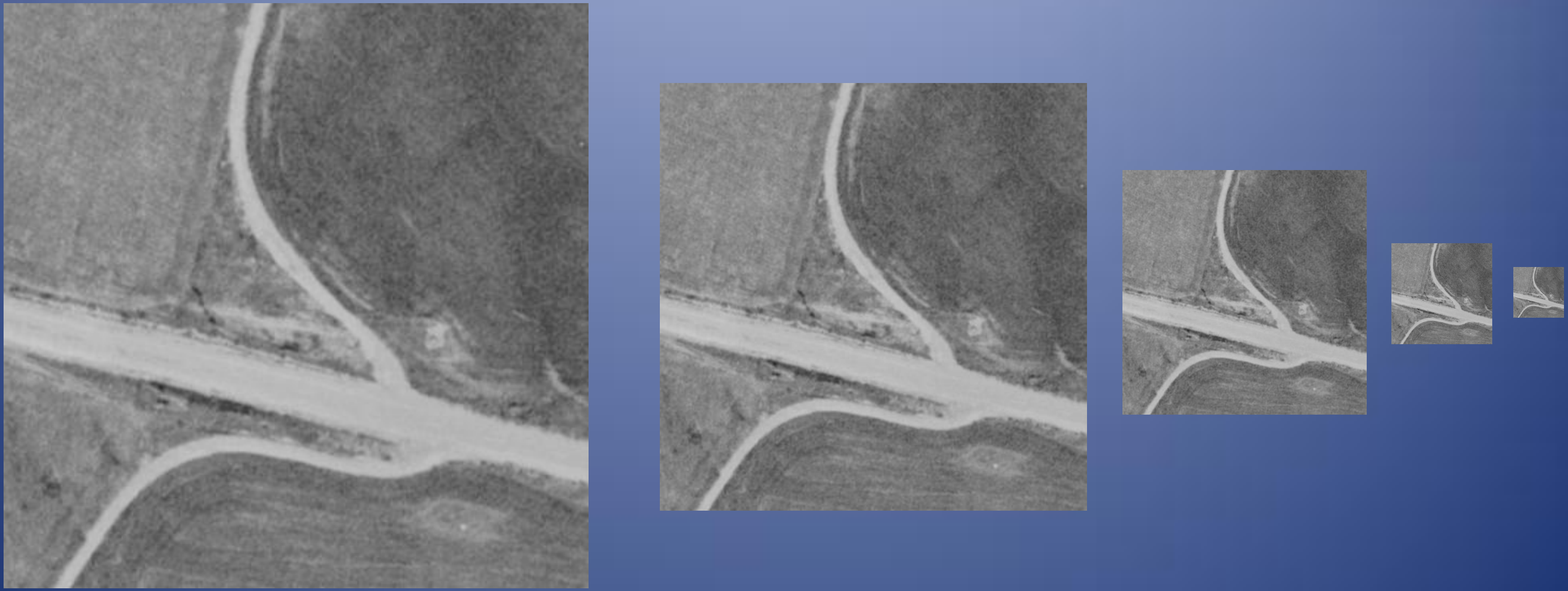
Resolution\nScales	256x256	512x512	1024x1024	2048x2048
9	0.2 1	1 9	4 28	18 107
12	0.25 2	1.3 9.5	6 38	25 155
17	0.7 4	2 17	8 59	36 243

These results were obtained using this notebook: Intel Core2 Duo 2.50Ghz, 4Gb RAM, running on WindowsXP 32 bit.

Future Work

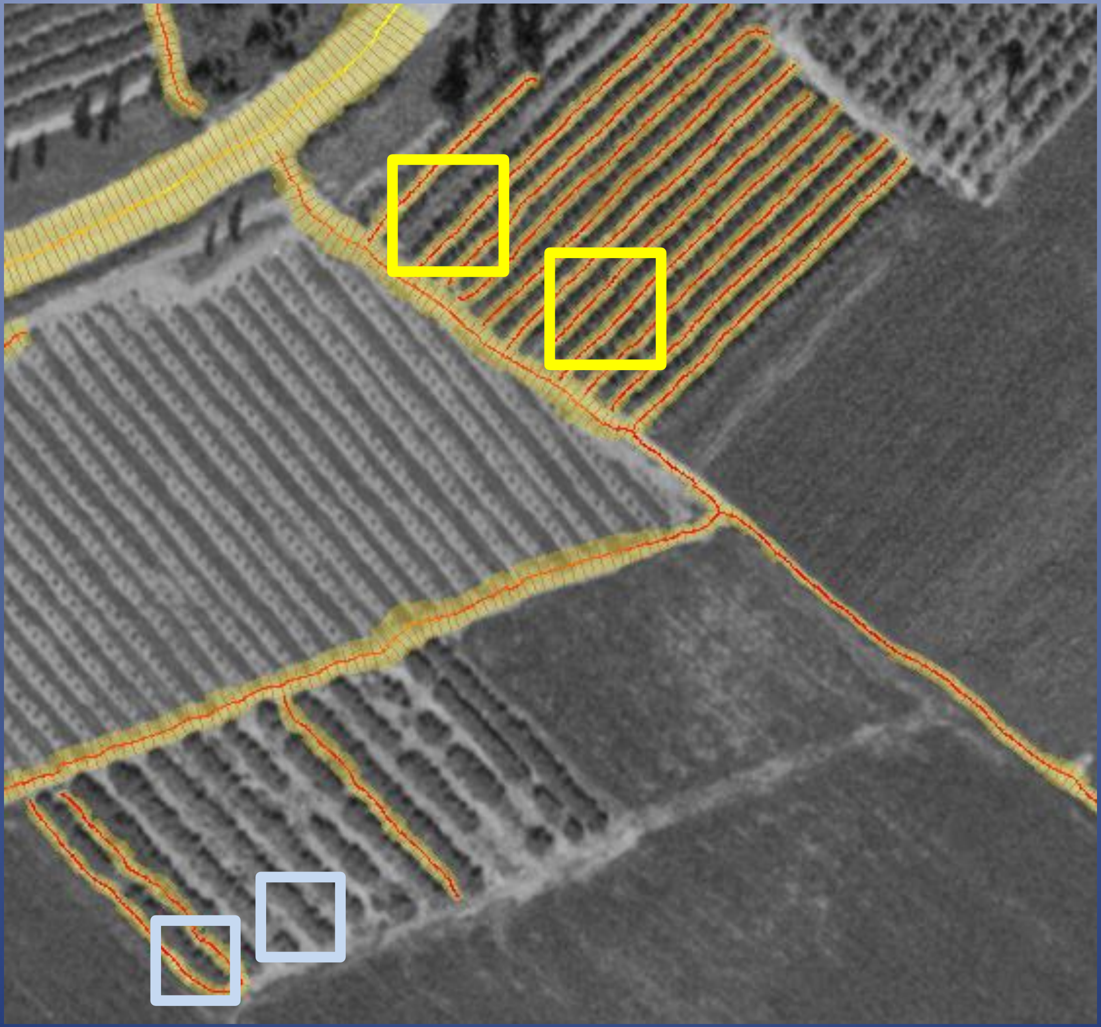
Future Work

1. Using a pyramidal image representation in order to perform calculations in different scales:



Future Work

2. Propagating discoveries of ridges by local feature matching similarity.

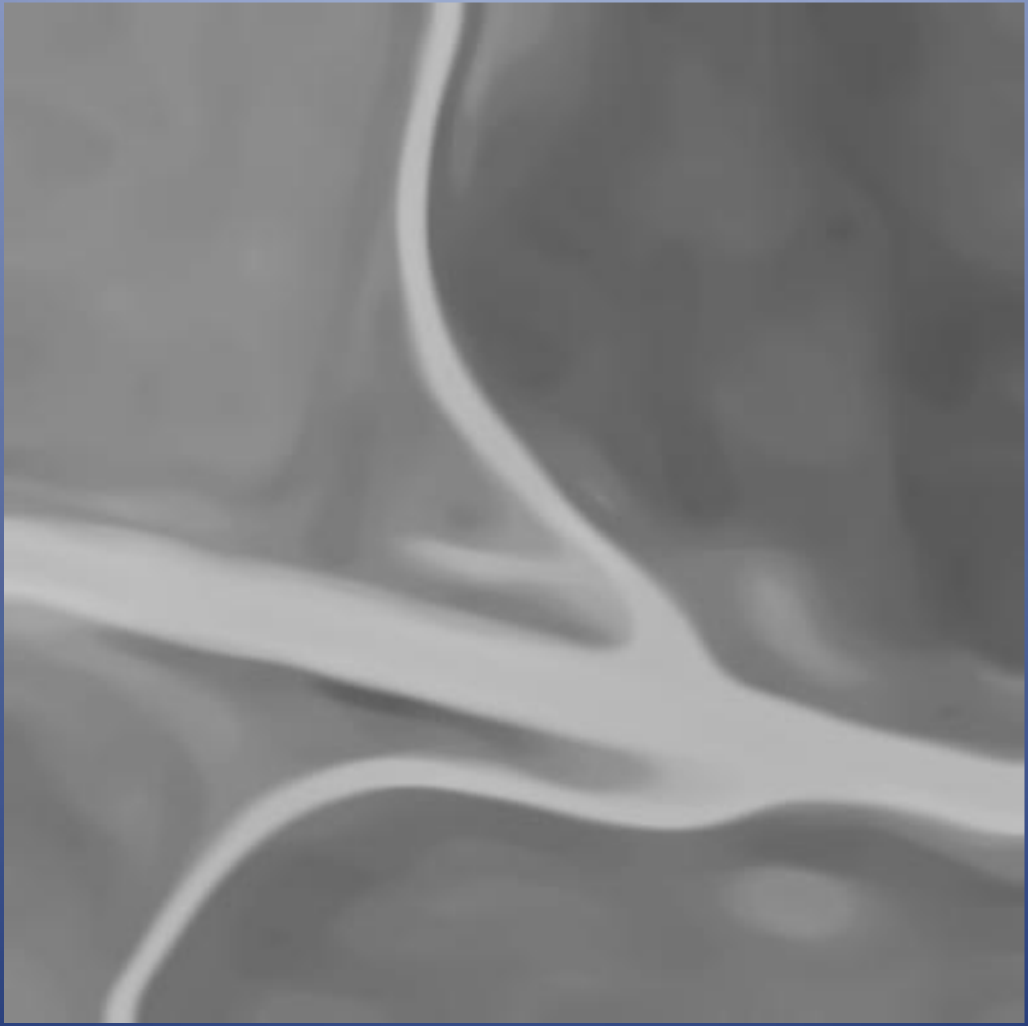


Future Work

3. Enhancing ridges prior to detection using Beltrami flow or another ridge-enhancing flow.

Future Work

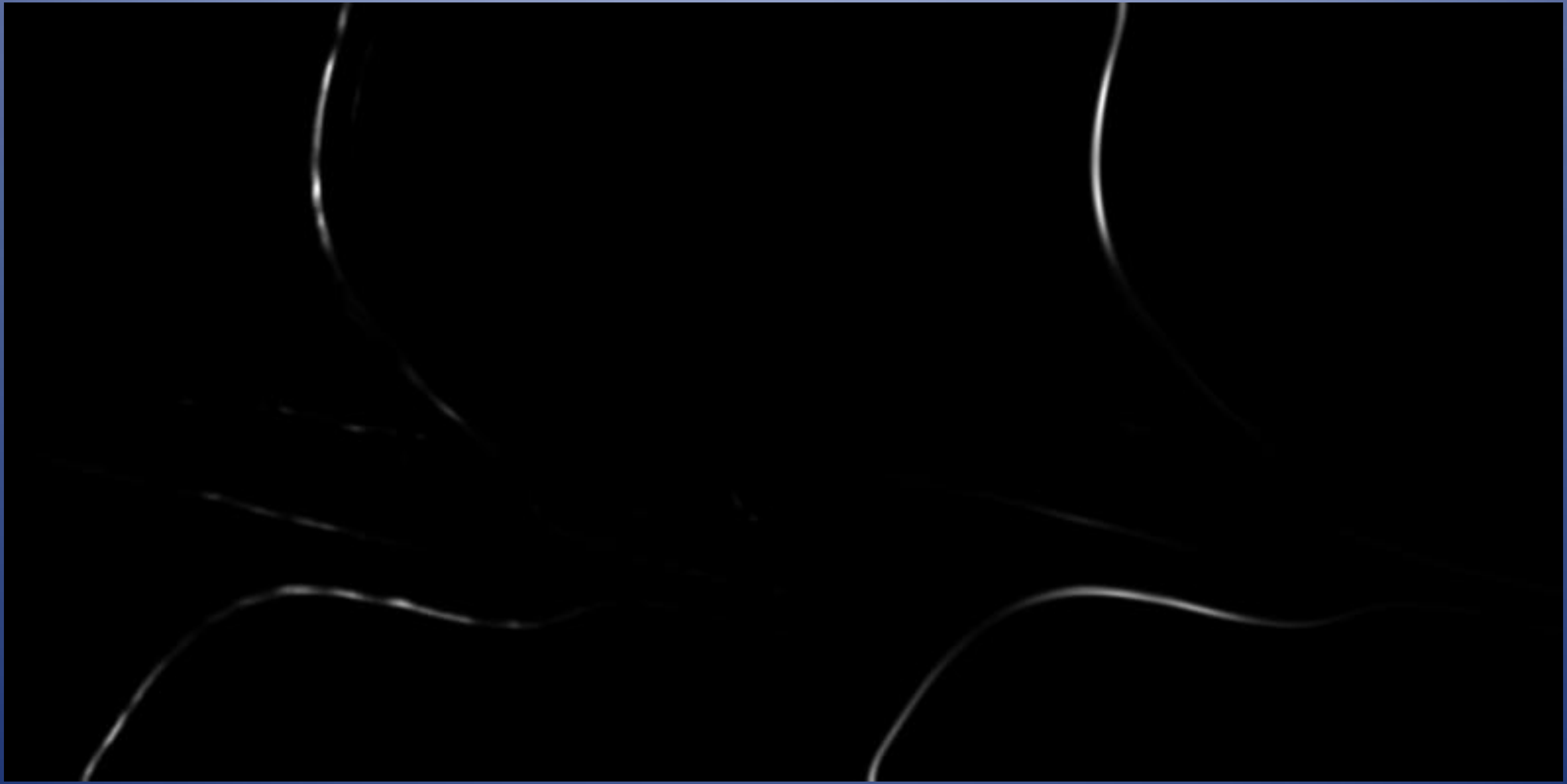
3. Enhancing ridges prior to detection using Beltrami flow or another ridge-enhancing flow.



Future Work

3. Enhancing ridges prior to detection using Beltrami flow or another ridge-enhancing flow.

Ridge strength function for small scale:

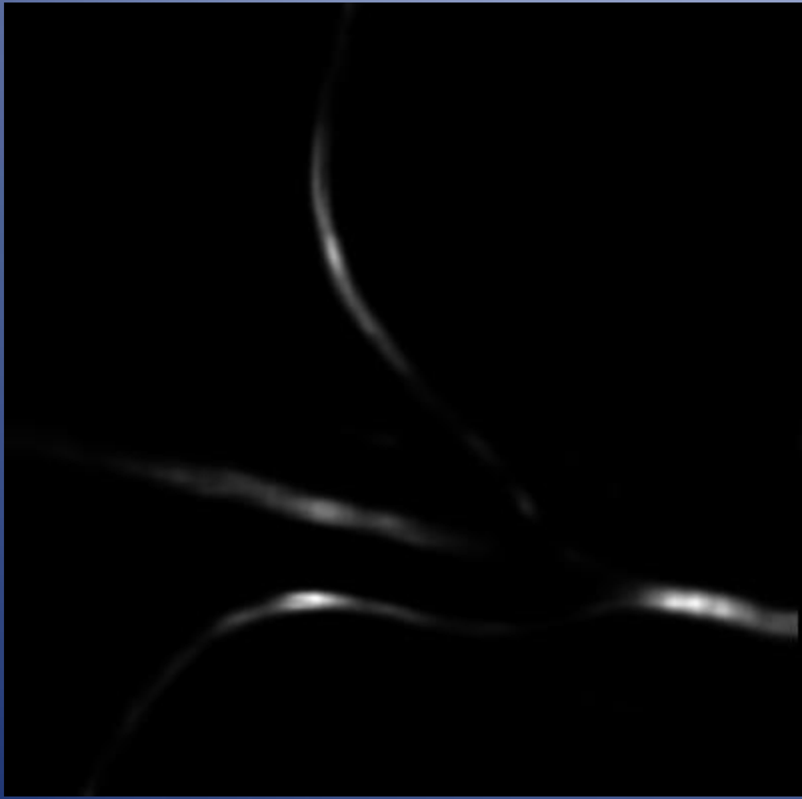


Without flow With flow

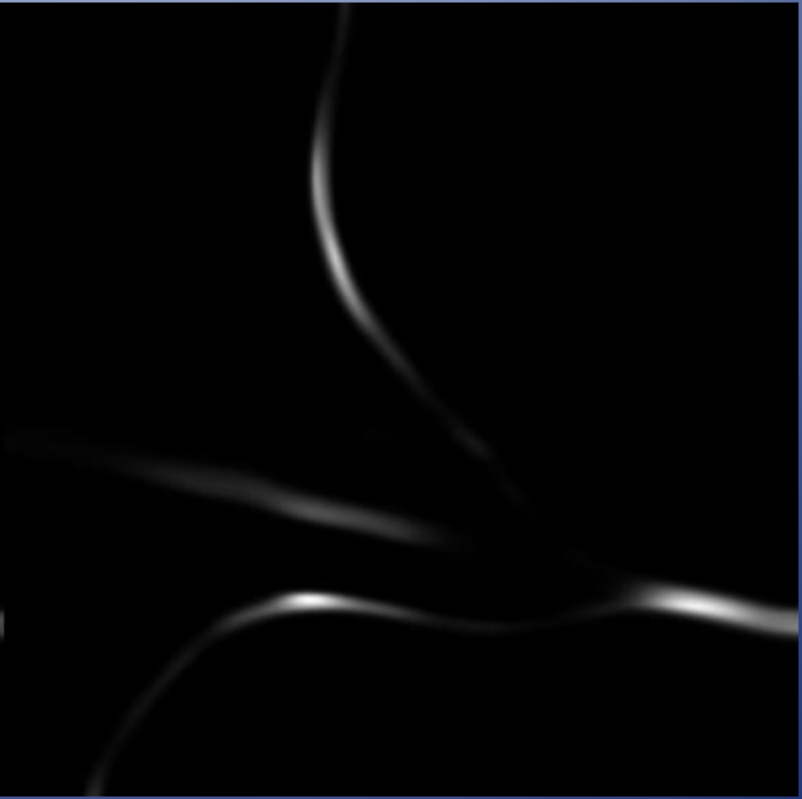
Future Work

3. Enhancing ridges prior to detection using Beltrami flow or another ridge-enhancing flow.

Ridge strength function for large scale:



Without flow



With flow

References

1. Lindeberg,T. “Edge detection and ridge detection with automatic scale selection” , International journal of Computer Vision, 30 ,2 ,pp 117-154 , 1998.
2. Witkin, A. P. "Scale-space filtering", Proc. 8th Int. Joint Conf. Art. Intell., Karlsruhe, Germany,1019–1022, 1983.
3. Koenderink, Jan "The structure of images", Biological Cybernetics, 50:363–370, 1984.

Acknowledgements

Renen (our supervisor) – thank you for your support and useful advice during all this time and hard work.

Also a special thanks for our lab manager, Aram Movsisian, for providing us all things we needed along the way.

Mika (Tal’s girlfriend) – thank you for your support during the hard times.

Acknowledgements

Renen (our supervisor) – thank you for your support and useful advice during all this time and hard work.

Also a special thanks for our lab manager, Aram Movsisian, for providing us all things we needed along the way.

Mika (Tal’s girlfriend) – thank you for your support during the hard times.

Fin