Medial Feature Detector
Version 0.5, March 2011

Anonymous ICCV submission

March 8, 2011

1 Purpose, license

This software is confidential and is meant to accompany ICCV 2011 anonymous paper submission #1016 in its supplementary material. Licence for its use is granted only for the purpose of paper review. A public version will be released by the authors if/when the paper is accepted for publication.

2 Environment, installation

This program has been developed in C++ using MS Visual Studio under Windows XP. It uses OpenCV libraries only to open, save or display images. It therefore supports all image file types that OpenCV does, including JPEG, PNG, PBM/PGM/PPM, and TIFF. It has only been tested on Intel processors running Windows XP. All required library files are included within this distribution, in the same folder with MFD executable. No installation is needed—just unzip the distributed archive into a local folder and MFD should run.

3 Contents

Main files:

`mfd.exe`	MFD executable
`mfd.pdf`	this document
`img/`	folder with test images

MS Visual Studio dependencies, also used by OpenCV:

`Microsoft.VC80.CRT.manifest`
`Microsoft.VC80.OpenMP.manifest`
`msvcp80.dll`
`msvcr80.dll`
`vcomp.dll`

OpenCV libraries:

```
cv110.dll
cvaux110.dll
cxcore110.dll
cxts001.dll
highgui110.dll
ml110.dll
```

Test images in folder `img/`:

```
alumgrns.png
b_graf1.png
b_graf2.png
b_frag.png
graf1.png
graf2.png
```

Files starting with `b_` are binary, the rest are gray-scale. MFD automatically converts color images to gray-scale; it uses no color information.

4 Syntax, command list

Use the following syntax to execute MFD:

```
mfd <command(s)> <filename(s)> [options]
```

Alternatively, to see information about MFD, its syntax above and a complete list of commands and options, simply type

```
mfd
```

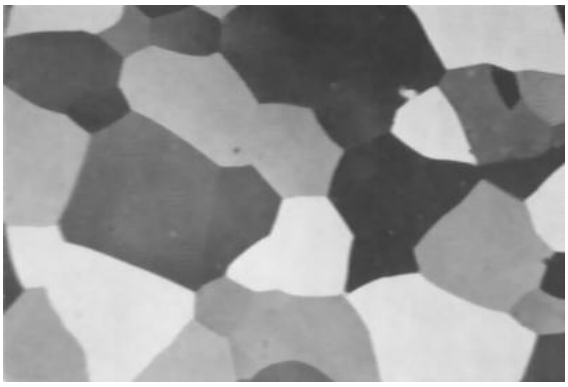
In this document, only a summary of the most important commands and options is given on specific examples. You may experiment with more advanced options, but no further documentation is given.

5 Restrictions

MFD's *view mode* (normally available with option `-v`) is *forced* in the current distributed version. This means one can only view MFD's output on screen but not save files to disk. Also, no feature descriptors (normally available with option `-de`) are supported; only viewing detected features on screen. All remaining commands and options are fully supported.

6 Distance map and medial axis

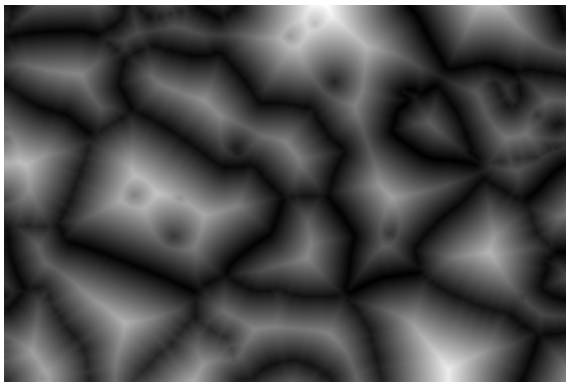
The first examples are on test image `alumgrns.png`:

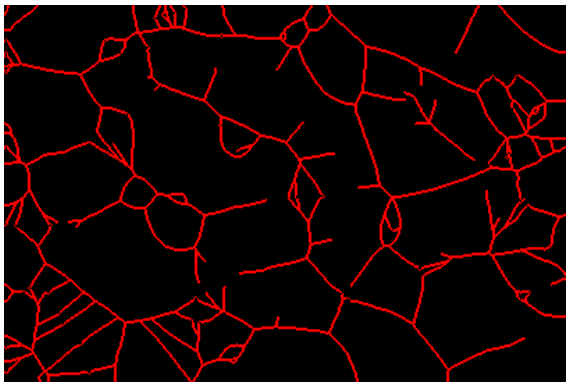


Given a gray-scale image, commands `-d` and `-m` compute the *weighted distance map* and *weighted medial axis*, respectively. They may be used independently, or together:

```
mfd -d -m img\alumgrns.png
```

In this case, the two output images are displayed in sequence:

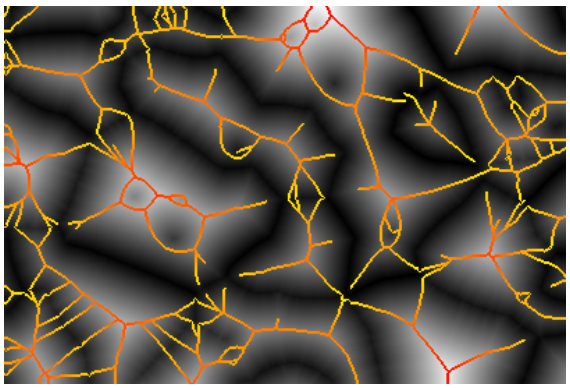
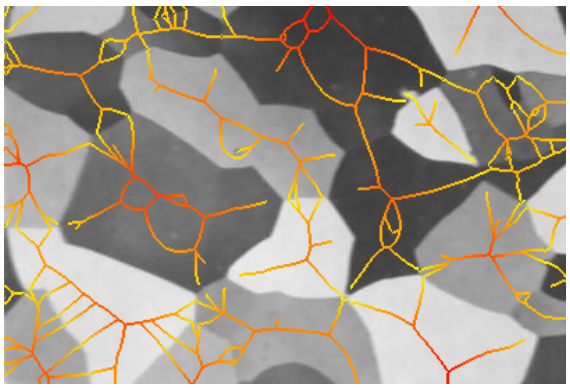




It is more useful to display the medial axis *overlayed* on top of the distance map or the input image, with options `-od` or `-o`, respectively:

```
mfd -m img\alumgrns.png -od -a
mfd -m img\alumgrns.png -o -a
```

Here we have also used option `-a` to visualize *height* on the medial axis using a yellow-red color map (yellow/red being low/high, respectively):

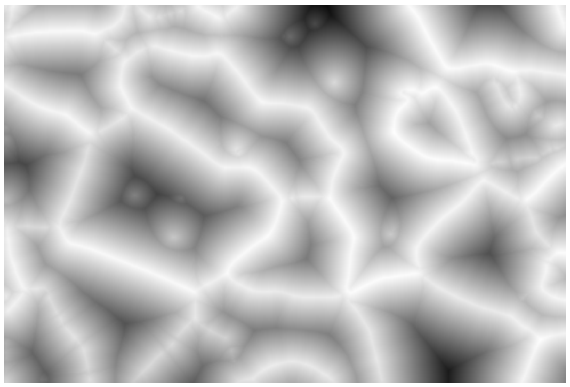
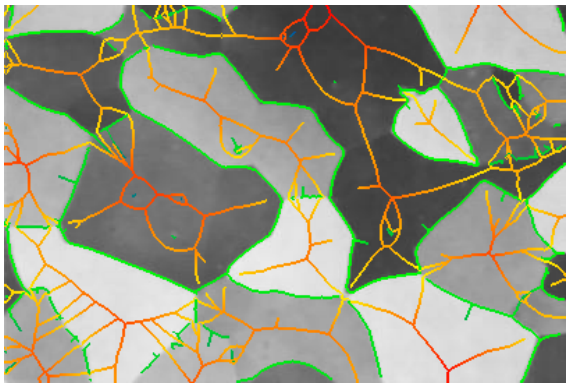



7 Duality and decomposition

Duality can be demonstrated by commands `-ud` and `-u`, computing the *dual distance map* and *dual medial axis*, respectively:

```
mfd -ud img\alumgrns.png
mfd -u img\alumgrns.png -o -a
```

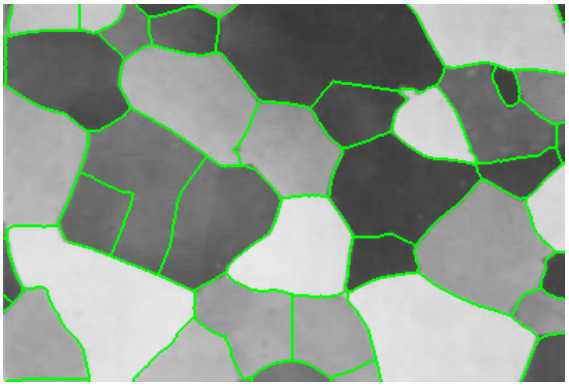
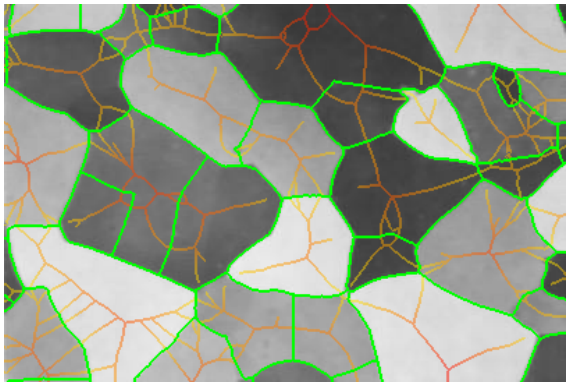
Here, the dual medial axis is displayed on top of the image and together with the primal medial axis. It is clear that the dual medial axis lies close to image boundaries, yet there are undesirable effects near crossings of the two types of curves:

Though the latter may be explored with option `-sd`, let's proceed to the *medial axis decomposition* and image *partitioning* with command `-p`:

```
mfd -p img\alumgrns.png -o
mfd -p img\alumgrns.png -o -a -pm
```

Here, option `-pm` displays the medial axis underlying the partition boundaries:

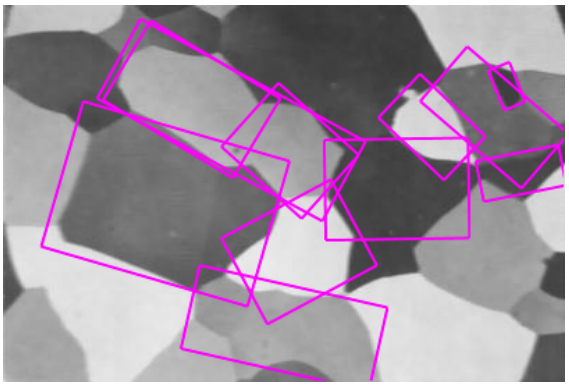
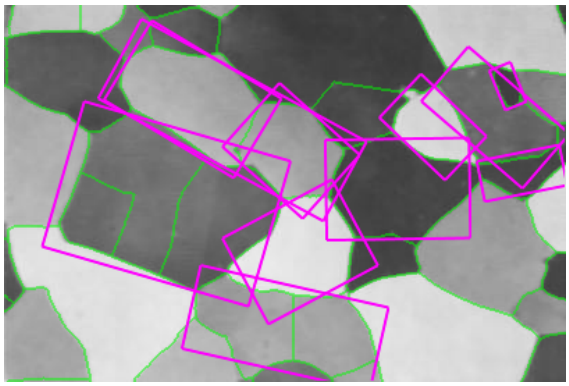



8 Medial features

Finally, let's detect *medial features* with command `-f`:

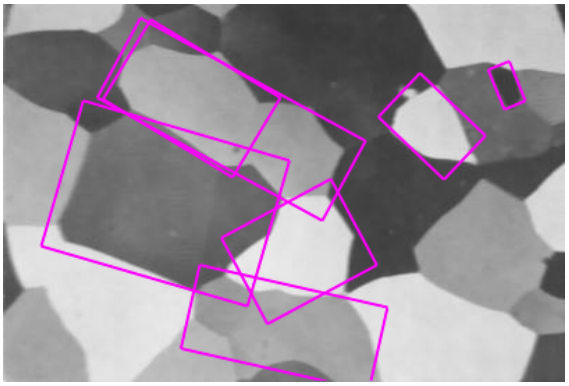
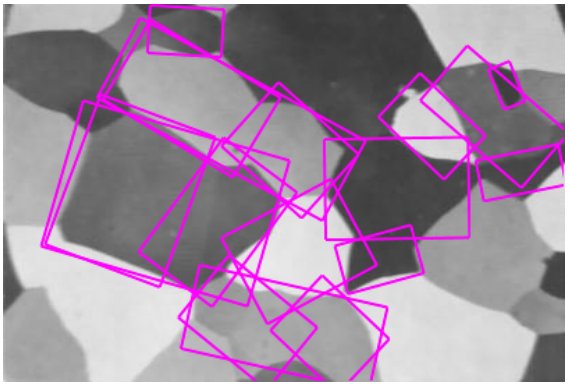
```
mfd -f img\alumgrns.png -o
mfd -f img\alumgrns.png -o -fp
```

where, similarly to `-pm`, option `-fp` displays the partition underlying the features:

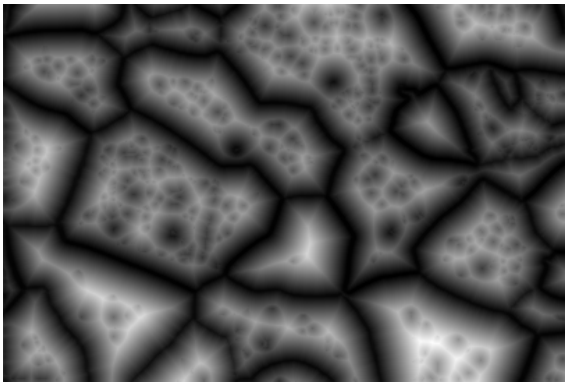
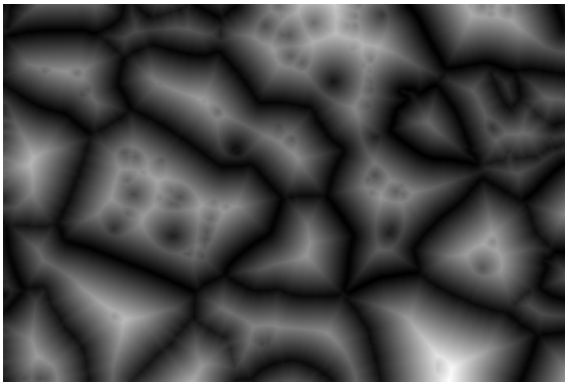
The *shape fragmentation* threshold τ here corresponds to parameter `-ff`. A lower value is more selective, while the default is .6:

```
mfd -f img\alumgrns.png -o -ff .4
mfd -f img\alumgrns.png -o -ff .8
```

The effect of *scale parameter* σ, corresponding here to parameter `-hf` with default value 4, is better seen on the distance map:

```
mfd -d img\alumgrns.png -hf 1
mfd -d img\alumgrns.png -hf 2
```

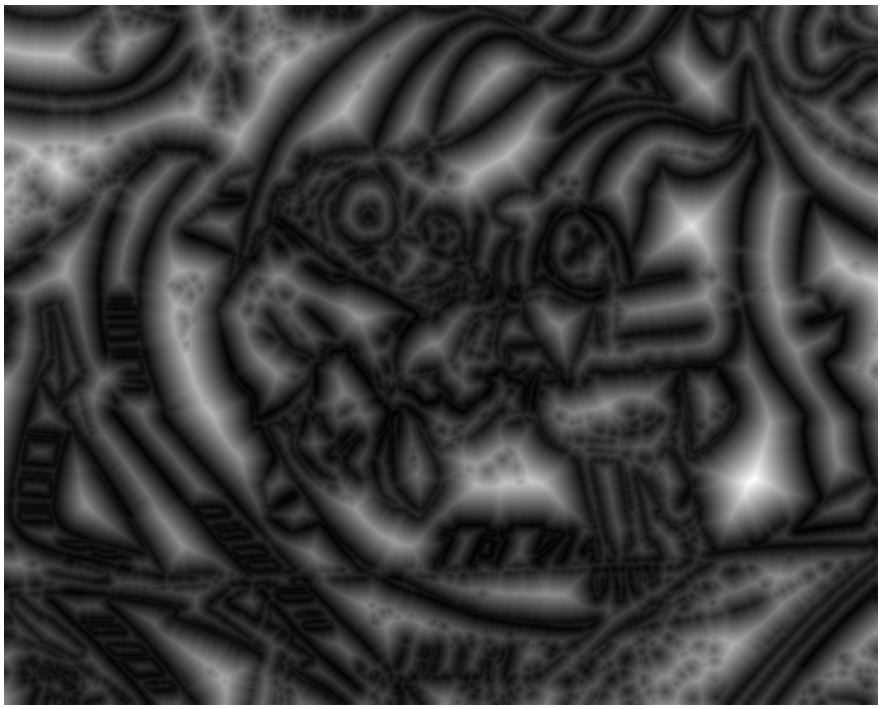



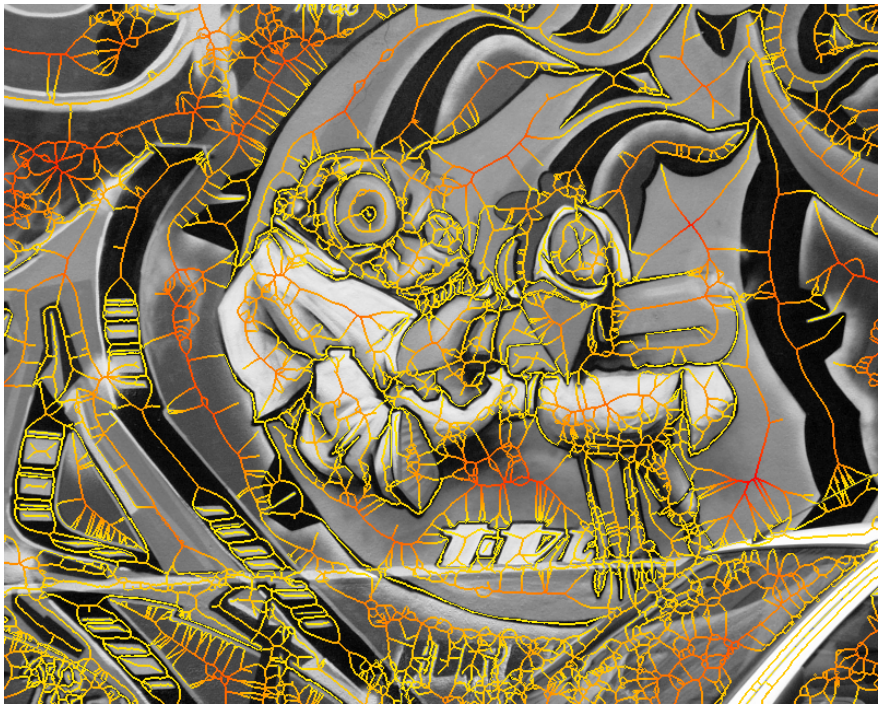
9 More examples

Let's look at the more complex image 1 of the GRAFFITI scene, `graf1.png`:

Here is the *weighted distance map* and *medial axis*:

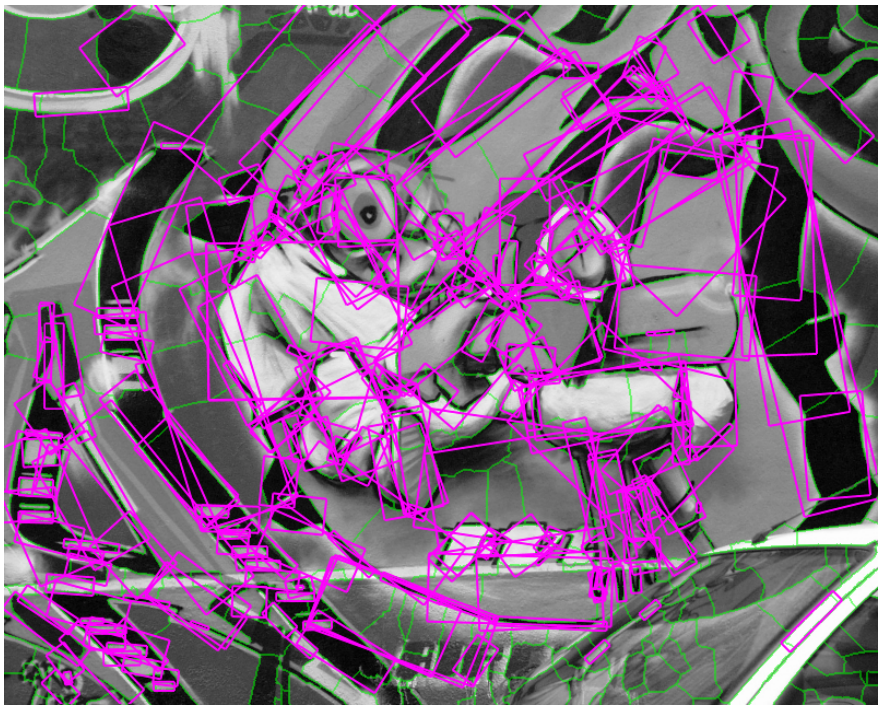
```
mfd -d img\graf1.png
mfd -m img\graf1.png -o -a
```

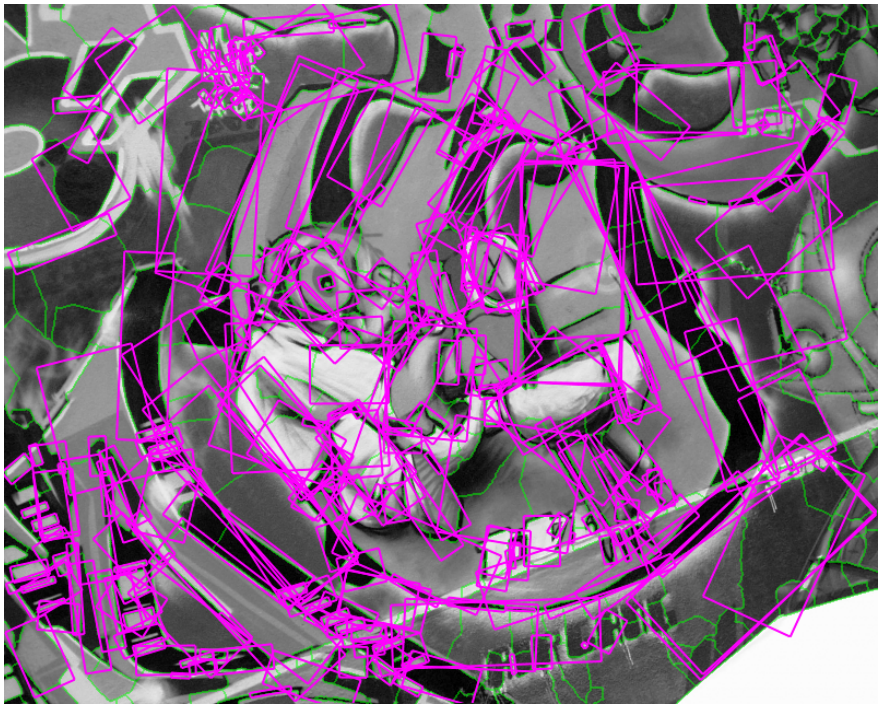




And here are the *medial features*, along with the underlying *partition*, of images 1 and 2:

```
mfd -f img\graf1.png -o -fp
mfd -f img\graf2.png -o -fp
```



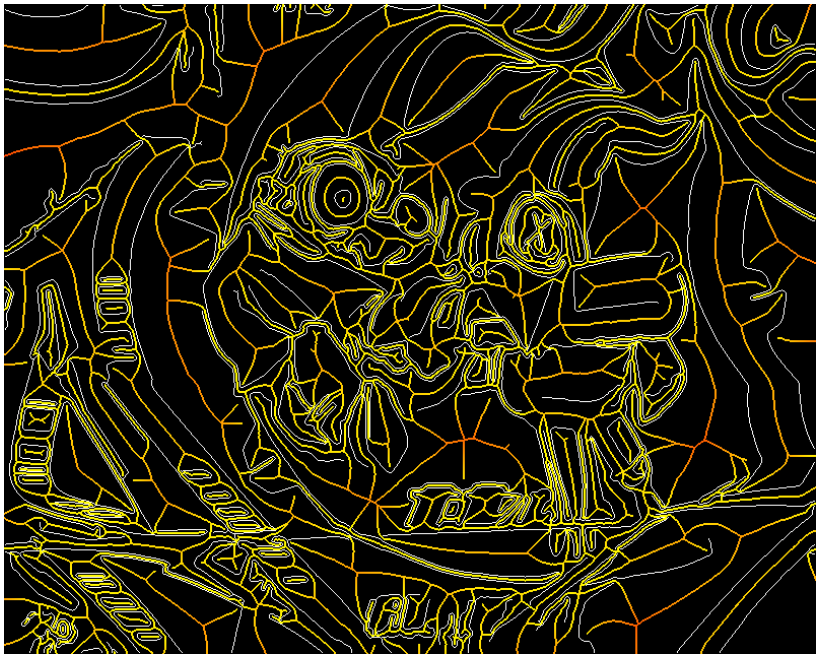


10 Binary input

Let's try the binary input b_graf1.png, obtained via Canny edge detection from graf1.png:

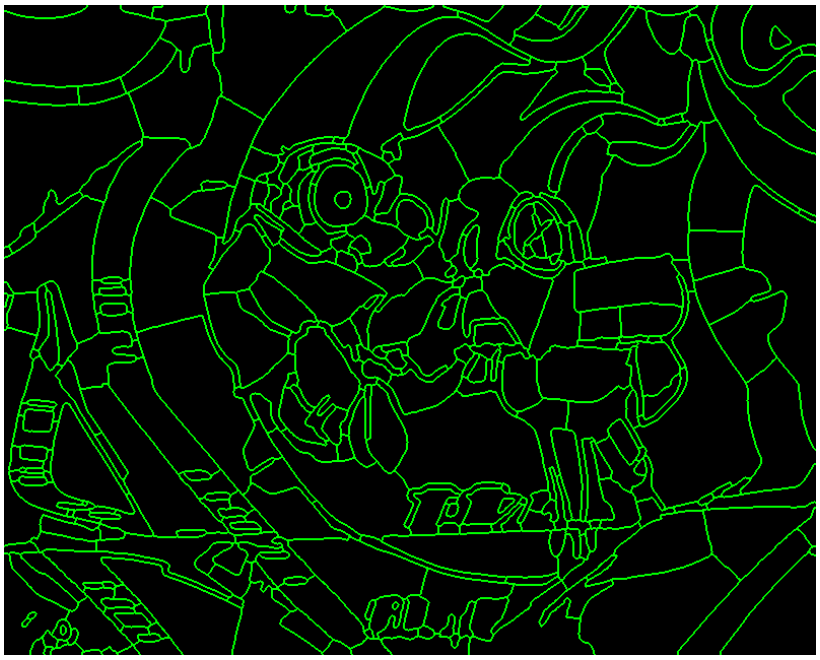
Binary images are processed in MFD with option -b. Everything works the same way as for grayscale images, with the main difference being a faster underlying implementation and that no gradient is computed. For instance, the medial axis:

```
mfd -m img\b_graf1.png -b -o -a
```



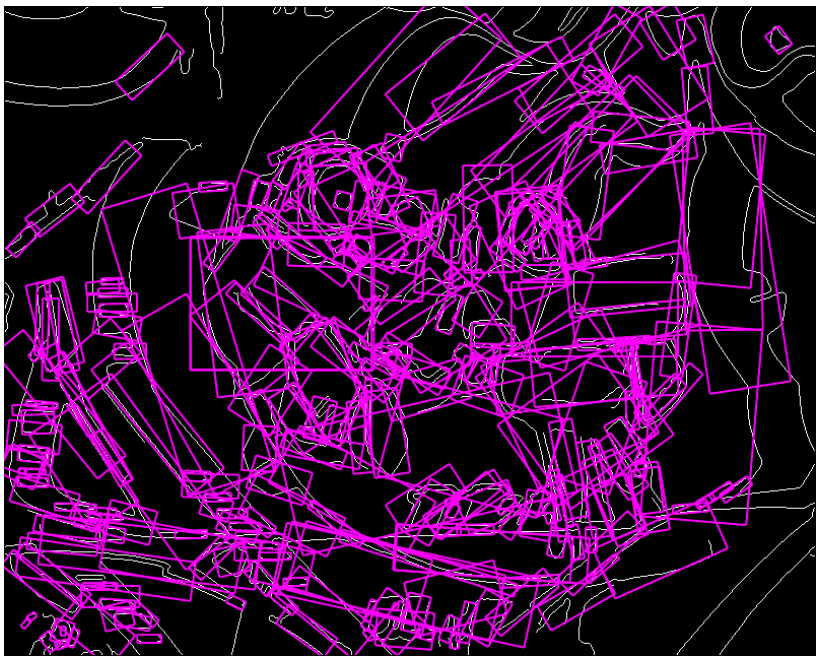
It is interesting to look at the *image partition* obtained from the edge map and similarities with the grayscale case:

```
mfd -p img\b_graf1.png -b
```



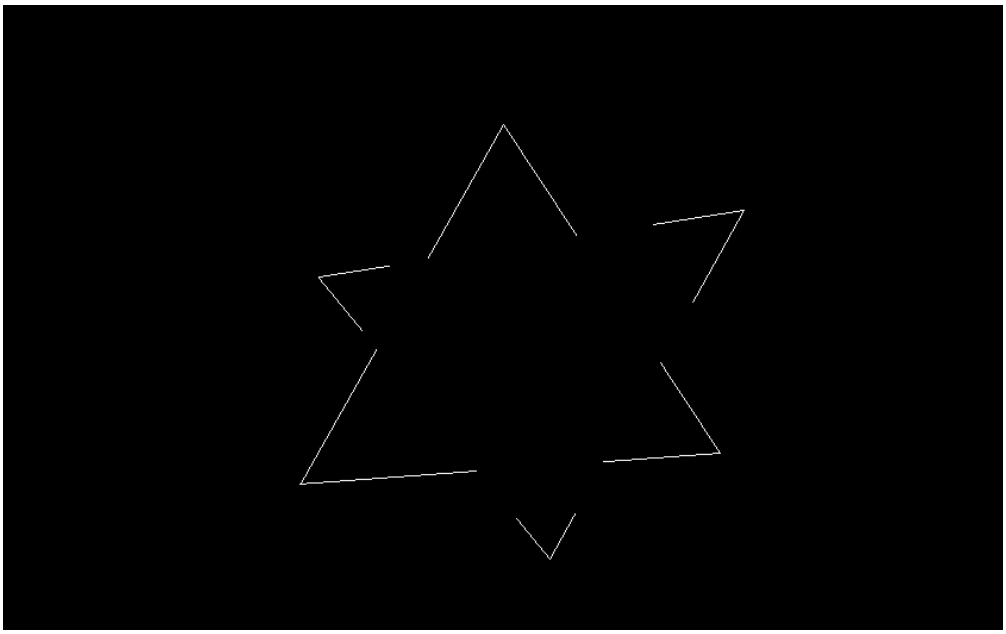
Similarly for the *medial features*:

```
mfd -f img\b_graf1.png -b -o
```



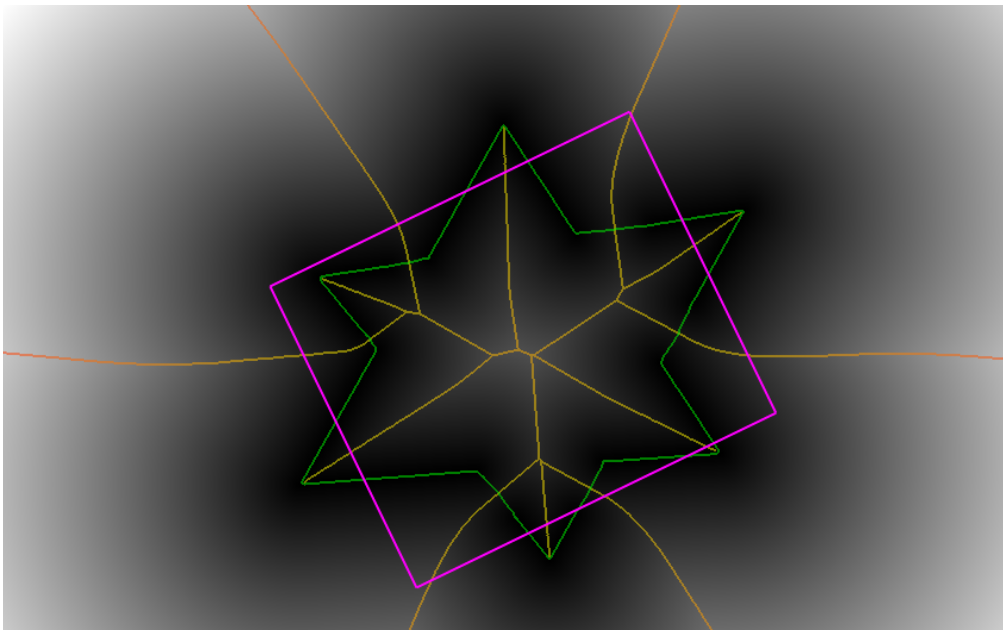
11 Fragmented shapes

Finally, a simple binary image of a fragmented shape, b_frag.png:



and the corresponding distance map, medial axis, partition, and features:

```
mfd -f img\b_frag.png -b -a -od -fp -pm
```



12 Further options

Further options worth exploring are the following:

`-t`	display detailed timing
`-vs`	display detailed statistics
`-i`	display distance isocontours
`-re`	display chord residue
`-vc`	display labels after medial axis decomposition
`-fc`	show underlying feature cover
`-df`	simulate grassfire in distance propagation, *e.g.* `-df 5`