Medial Feature Detector
Version 0.6

Yannis Avrithis

October 22, 2011

1 License

This software is meant to accompany [1], and may be used freely for research purposes only. Please cite [1] if you use this software.

2 Environment, installation

MFD uses OpenCV library to open, save or display images. It supports all image file types that OpenCV does, including JPEG, PNG, PBM/PGM/PPM, and TIFF. Two platforms are supported, Linux 64-bit and Windows 32-bit. For the Windows version, 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. For the Linux version, please install OpenCV first from your Linux distribution (package `libcv`) or from `http://opencv.willowgarage.com`.

3 Contents

Main files:

`mfd`	MFD Linux 64-bit executable
`mfd.exe`	MFD Windows 32-bit executable
`mfd.pdf`	this document
`img/`	folder with test images

MS Visual Studio dependencies, also used by OpenCV (Windows only):

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

OpenCV library files (Windows only):

```
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 grayscale. MFD automatically converts color images to grayscale; it uses no color information.

4 Syntax, command list

Use the following syntax to execute MFD:

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

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

```
mfd
```

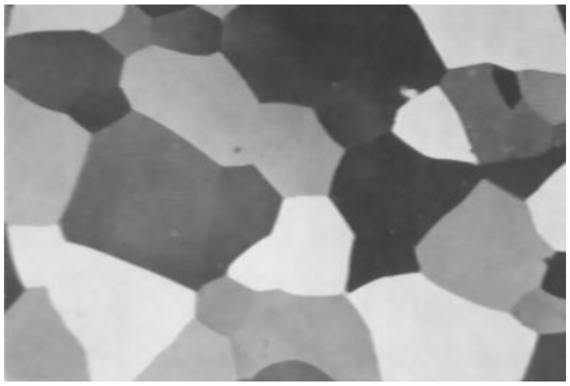
This document is a "quick start guide" rather than a manual. Only a summary of the most important commands and options is given on specific examples. You may experiment with more advanced options, but no further documentation is given.

5 View mode

By default, MFD operates in *file mode*, saving its output in one or more files with appropriate filename extensions in the same folder as the input file(s), unless a different `<output folder>` is specified. Alternatively, option `-v` causes MFD to operate in *view mode*, displaying output sequentially on screen instead of saving files to disk. In the following, option `-v` is assumed.

6 Distance map and medial axis

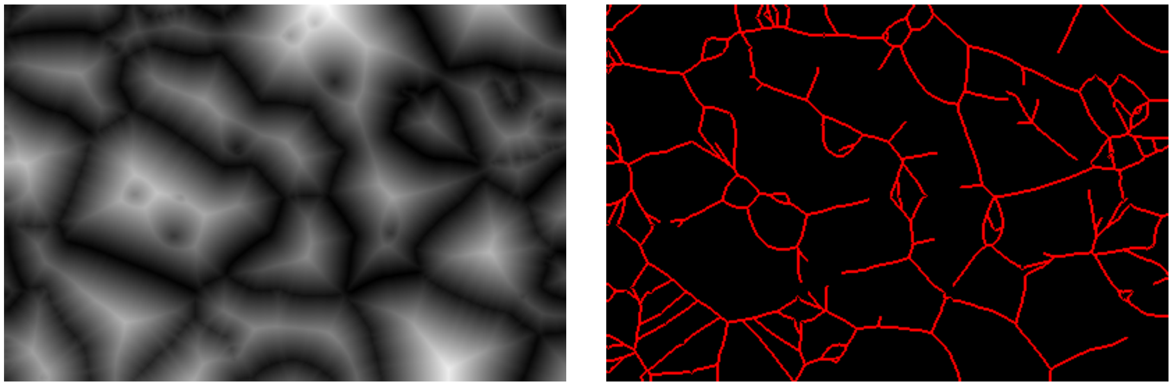
The first examples are on test image alumgrns.png:



Given a grayscale 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
```

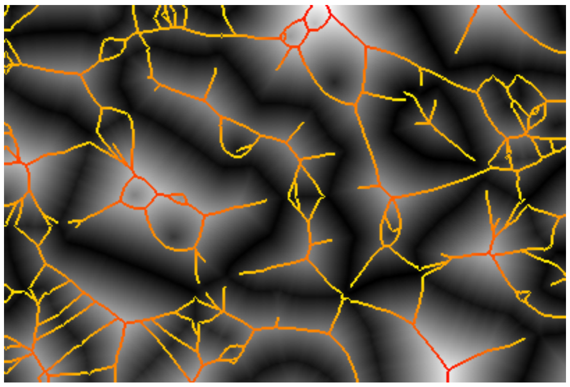
In this case, the two output images are displayed in sequence:

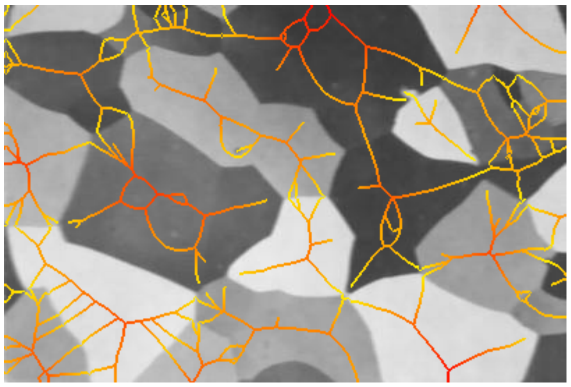


It is more useful to display the medial axis *overlayed* on top of the distance map or the input image, with options -od or -o, respectively:

```
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):



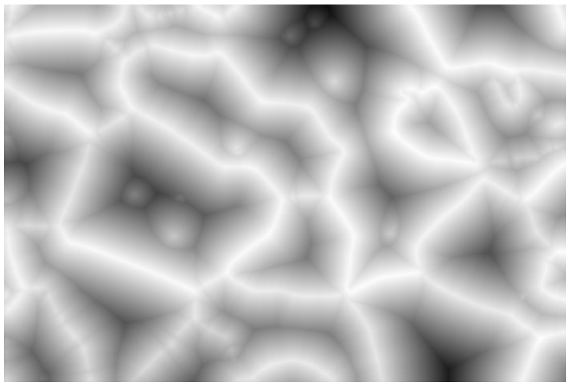


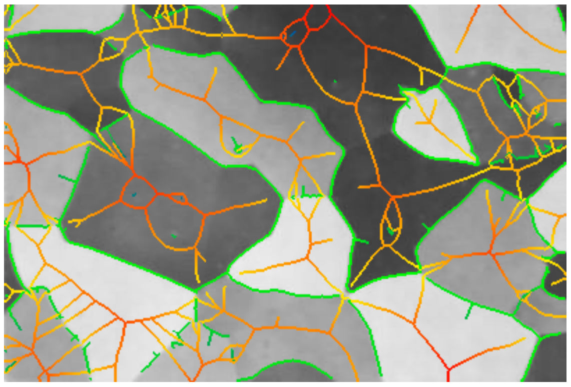
7 Duality and decomposition

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:

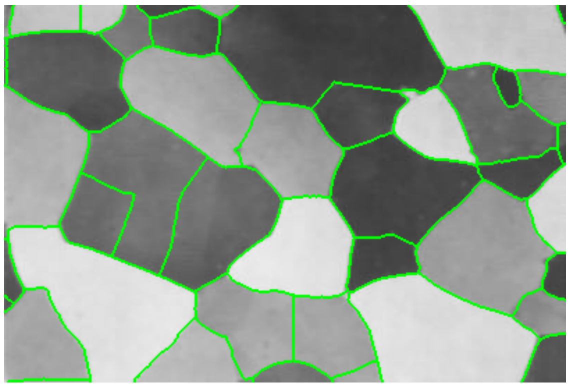
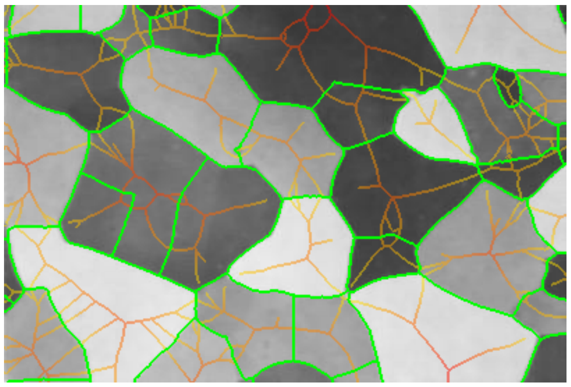




Though the latter may be explored with option -sd, let's proceed to the *medial axis decomposition* and image *partitioning* with command -p:

```
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:

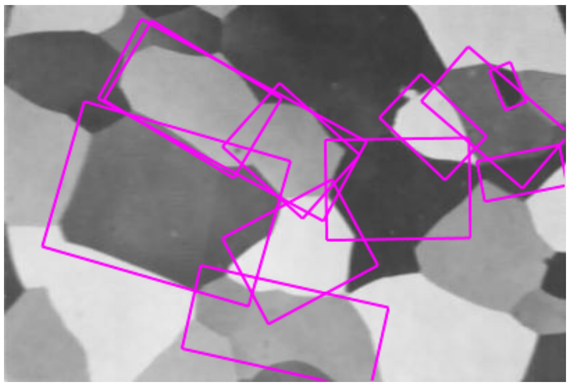
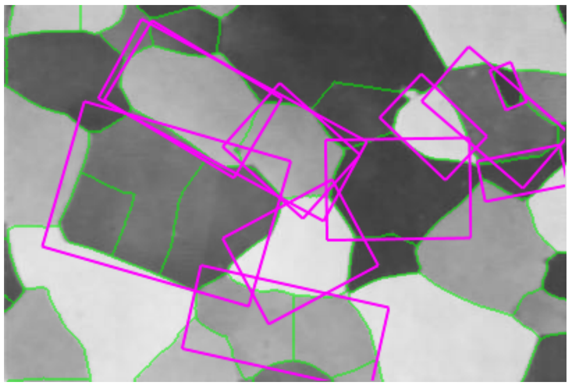



8 Medial features

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

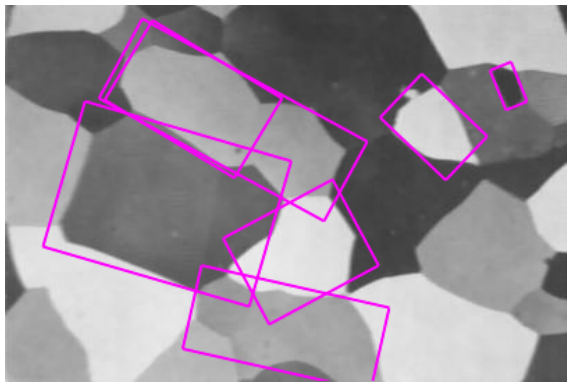
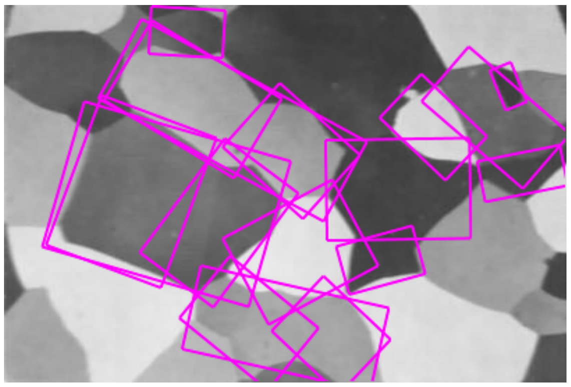
```
    mfd -f img\alumgrns.png -o
    mfd -f img\alumgrns.png -o -fp
```

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

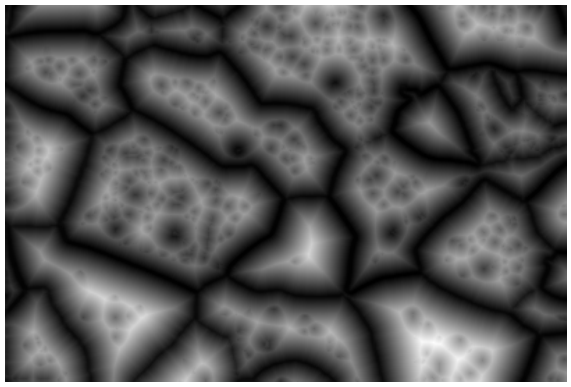
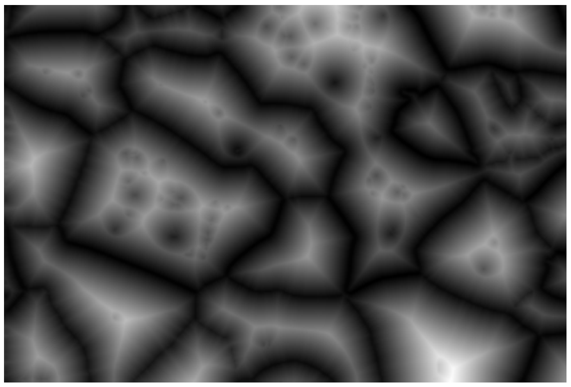
The *shape fragmentation* threshold τ here corresponds to parameter `-ff`. A lower value is more selective, while the default is .6:

```
    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
```

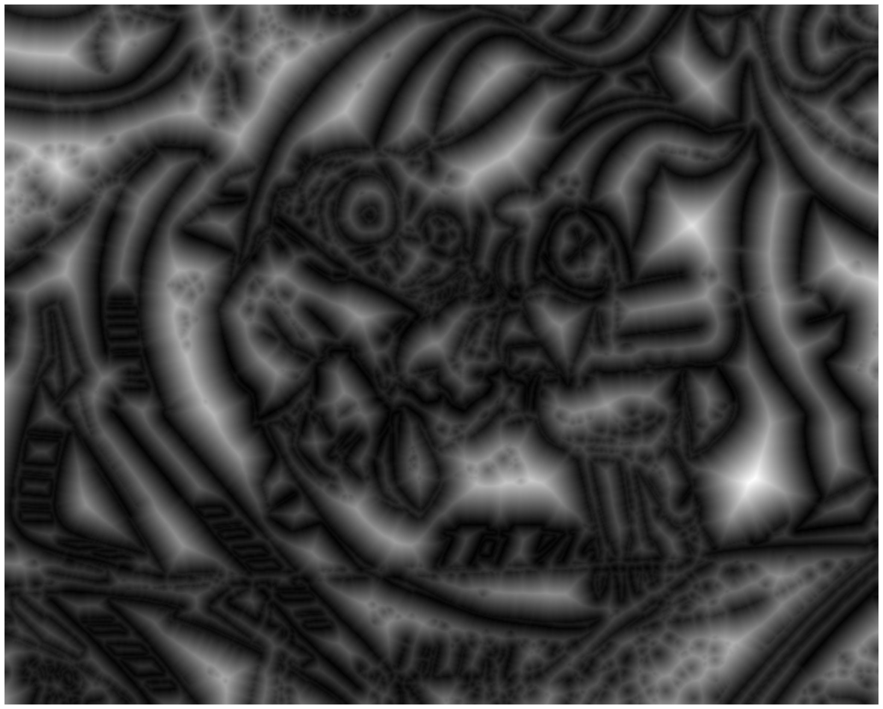



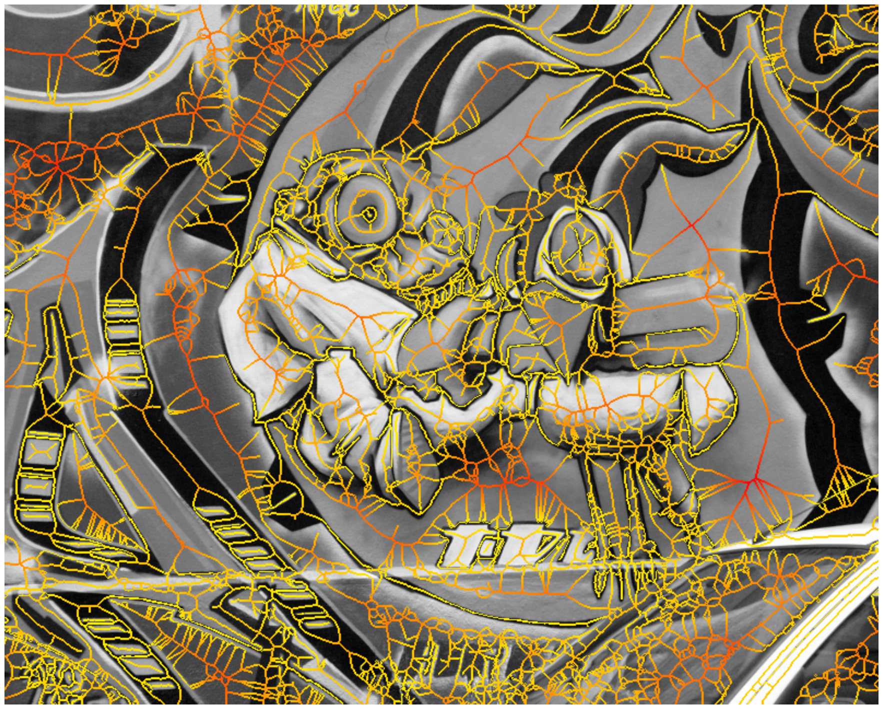
9 More examples

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

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

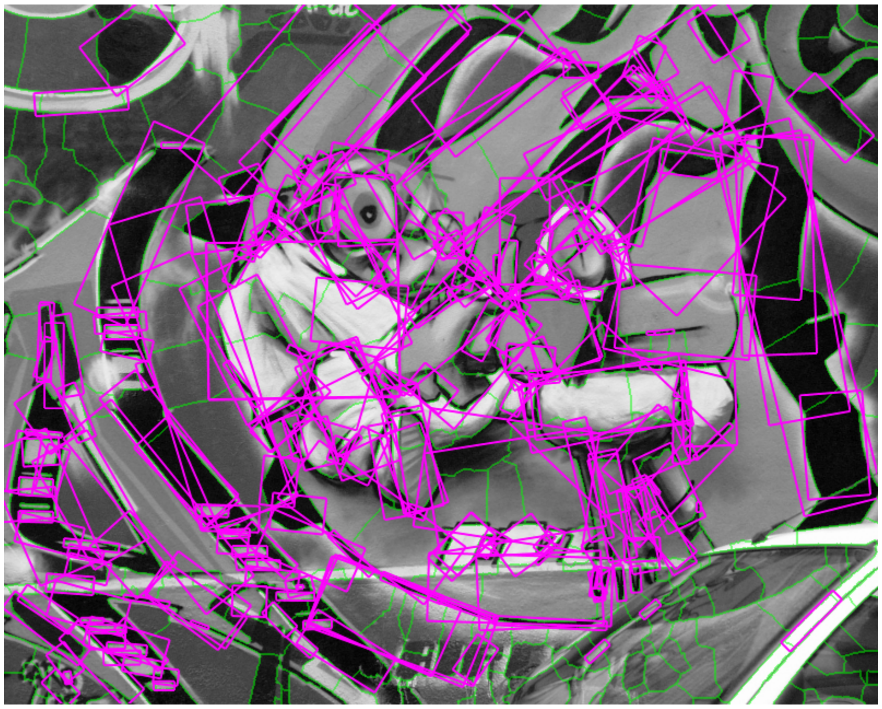
```
mfd -d img\graf1.png
mfd -m img\graf1.png -o -a
```

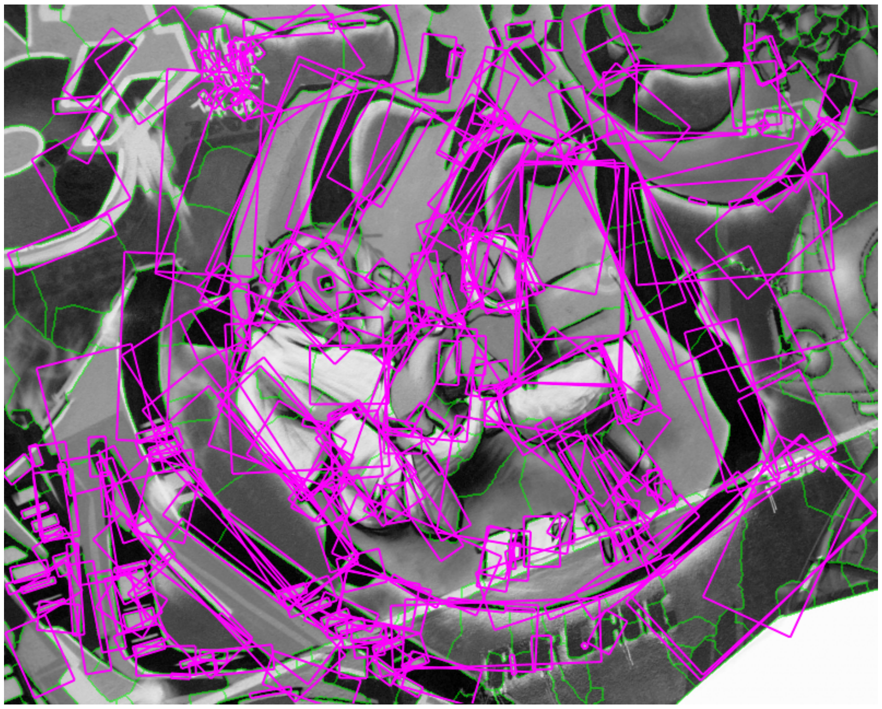




And here are the *medial features*, along with the underlying *partition*, of images 1 and 2:

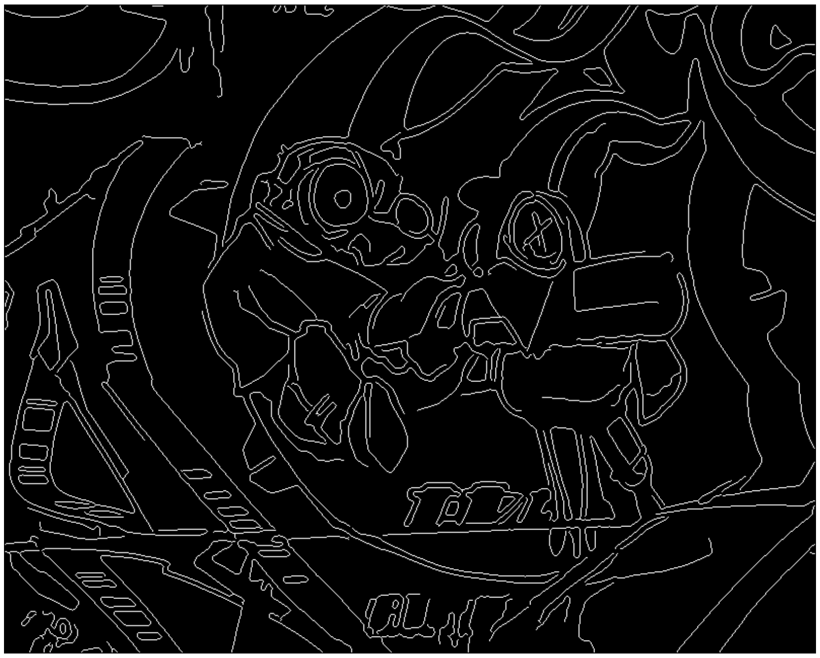
```
mfd -f img\graf1.png -o -fp
mfd -f img\graf2.png -o -fp
```





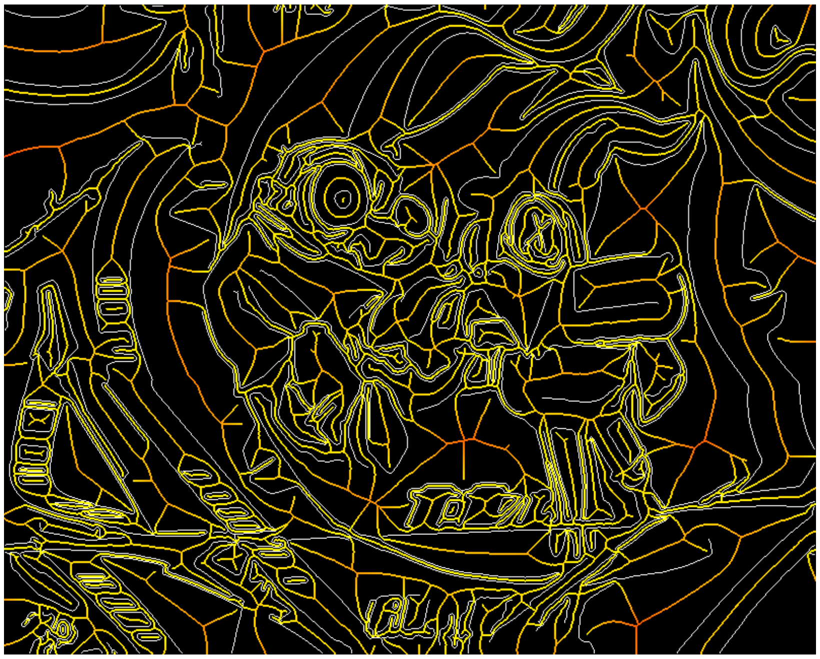
10 Binary input

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



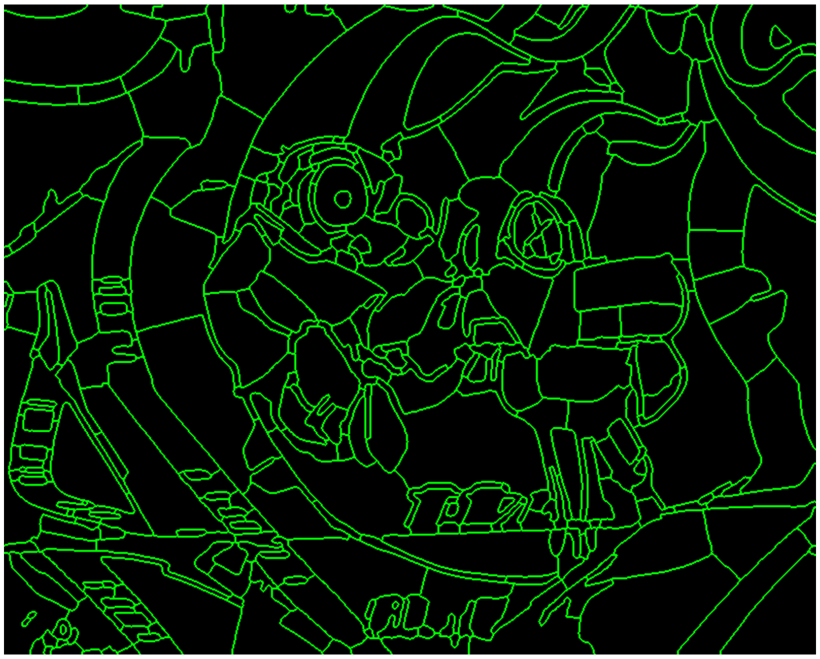
Binary images are processed in MFD with option `-b`. Everything works the same way as for grayscale images, with the main difference being a faster underlying implementation and that no gradient is computed. For instance, the medial axis:

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



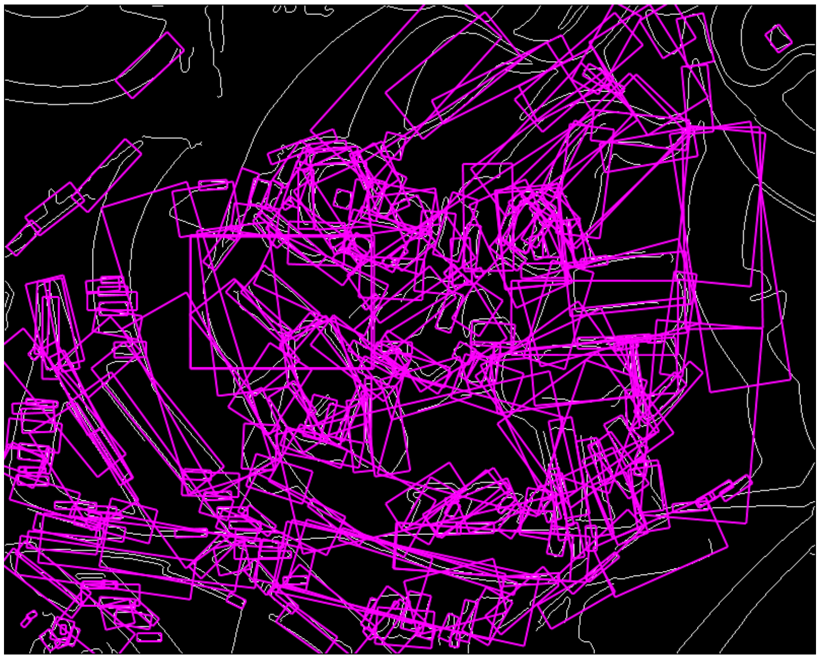
It is interesting to look at the *image partition* obtained from the edge map and similarities with the grayscale case:

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



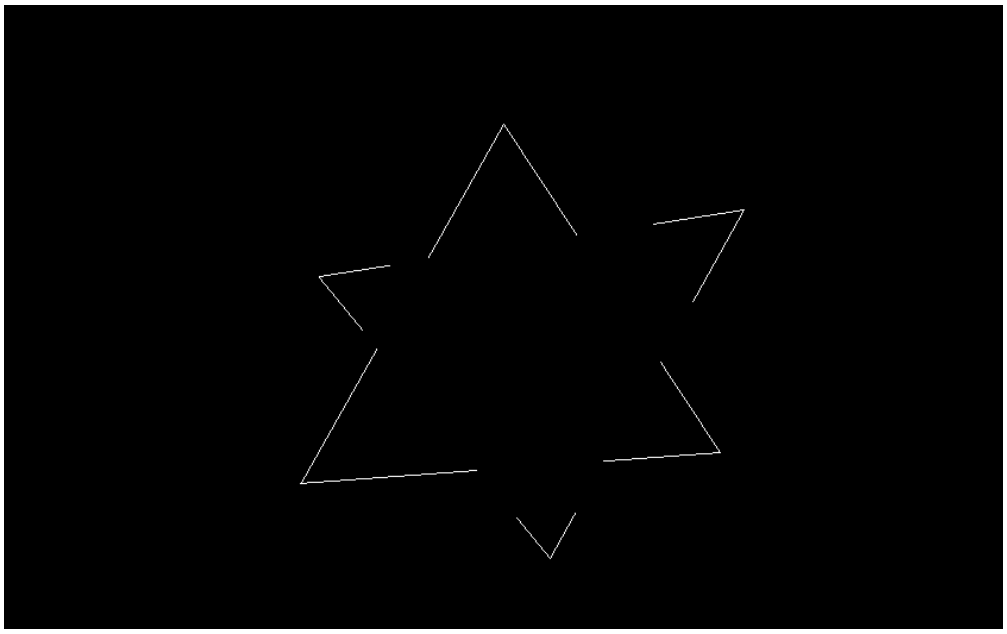
Similarly for the *medial features*:

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



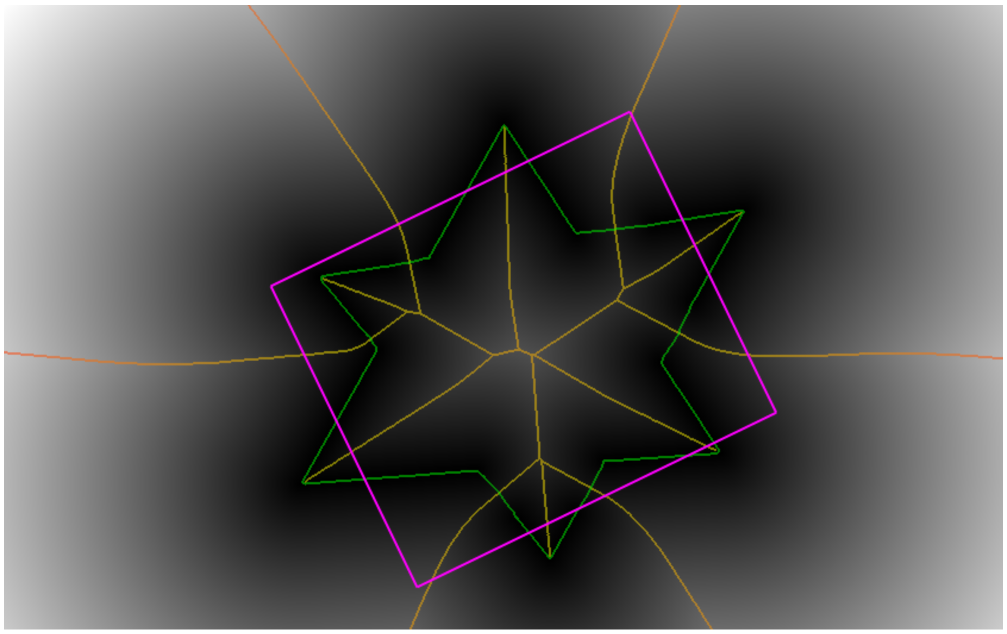
11 Fragmented shapes

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 Feature output format

Detected features are displayed as an image in view mode (`-v`), and saved in a text file otherwise (file mode). If you want to save this image under file mode, you may use option `-fi`. The text file contains one line per detected feature. Each line contains 5 numbers, separated by spaces. The first two numbers are always the x and y position of the feature centroid. The remaining 3 numbers depend on the format chosen:

`-fmp`	width w, height h, orientation θ (default)
`-fma`	elements a, b, c of covariance matrix C
`-fmm`	corresponding elements of inverse covariance matrix C^{-1}

Format `-fma` specifies the 2×2 covariance matrix

$$C = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

containing the second order central moments $a = \mu_{2,0}$, $b = \mu_{1,1}$, $c = \mu_{0,2}$ of all grid positions within the region covered by the feature. Format `-fmp` can be used to draw a rectangle around each feature. This rectangle is centered at the specified centroid, with width w and height h, and rotated counterclockwise around its center by θ radians. This is how the feature image is drawn. These quantities are obtained by eigendecomposition of C:

$$\begin{aligned} w^2 &= \frac{1}{2}(a + c + d) \\ h^2 &= \frac{1}{2}(a + c - d) \\ \theta &= \texttt{atan2}\left(\frac{1}{2}(a - c - d), b\right) \end{aligned}$$

where $d^2 = (a - c)^2 + 4b^2$. Finally, the inverse covariance matrix C^{-1} specified by format `-fmm` is the one used in K. Mikolajczyk's project *Affine Covariant Features*:

`http://www.robots.ox.ac.uk/~vgg/research/affine/`

In this case the text file begins by first specifying two additional numbers on separate lines. The first is always 0, meaning that no descriptors are given, and the second is the number of features detected. Then, Matlab script

`http://www.robots.ox.ac.uk/~vgg/research/affine/det_eval_files/display_features.m`

can be used to draw an ellipse around each feature.

13 Descriptors

When feature format `-fmm` is used, the detector and descriptor binary file available for both Linux and Windows at

> http://www.robots.ox.ac.uk/~vgg/research/affine/det_eval_files/extract_features2.tar.gz

can be used to compute descriptors for the detected features. In fact, if you copy the corresponding binary file to the same folder as MFD and rename it to `mik` (Linux) or `mik.exe` (Windows), then you may use MFD's command `-de` to call it internally and compute SIFT descriptors.
Please note that in this case the number of features increases, as each feature may give more than one descriptors corresponding to different dominant orientations. Also note that command `-de` forces feature format `-fmm` and scales each feature by a factor of 3 so that a larger region around the feature is used to compute the descriptor. This region can be further scaled by the "zoom" factor specified by option `-z`.

14 Output file names

Depending on the command(s) used, the corresponding output file(s) are named after the input file by appending an underscore and a keyword before the extension, as follows:

`dist`	(`-d`)	distance transform
`med`	(`-m`)	medial axis
`udist`	(`-ud`)	dual distance transform
`umed`	(`-um`)	dual medial axis
`uboth`	(`-u`)	dual distance transform and medial axis
`part`	(`-p`)	partition
`feat`	(`-f`)	features
`desc`	(`-de`)	descriptors

For instance, our first example

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

without `-v` generates `img\alumgrns_dist.png` and `img\alumgrns_med.png`.
Most output files are images, following the same image format and extension as the input. The exceptions are `feat` and `desc`, which are text files (`.txt`). As mentioned above, you may also save features as an image using option `-fi`. An input file that follows the above naming convention is normally recognized as previous output and ignored. This behavior is particularly useful in batch mode (see below), but you may override it with option `-rp`.

15 Batch mode

MFD can operate in batch mode and process multiple input files specified by a pattern as supported by the operating system, containing wildcards like `?` or `*`. It may optionally recurse into subfolders with option `-r` as well. Under Linux, you should enclose the filename pattern in single quotes for this to work properly. For instance,

```
mfd <command(s)> 'dir/image*.png' [options]
```

under Linux and

```
mfd <command(s)> dir\image*.png [options]
```

under Windows.

16 Memory allocation

To optimize for speed, all required memory allocation is performed once. Estimated memory sizes are pessimistic but there is no guarantee against overflow and no re-allocation mechanism. This means MFD may indeed crash on specific input, although this has not occurred on thousands of tested images. Should this happen, you may increase the allocated memory by a factor between 1 and 10, controlled by option `-ma`.

17 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
`-s`	medial axis scale; larger scale prunes more
`-mg`	sub-pixel accurate medial axis; only in binary mode
`-pd`	partition up to specified distance only
`-pi`	display partition by randomly colored interiors
`-fc`	show underlying feature cover

Some options cause interactive display, possibly useful for debugging. For instance:

`-df`	simulate "grassfire" in distance propagation, *e.g.* `-df 5`
`-vi`	display merging steps during partition or feature detection

There are dozens more options, displayed in categories by simply running `mfd`. The only available documentation is the 1-2 line text next to each option.

18 Exercise

Download image

`http://image.ntua.gr/iva/tools/mfd/fig/biarcs.png`

from the MFD home page, and reproduce the following results:

`http://image.ntua.gr/iva/tools/mfd/fig/biarcs_med_res.png`
`http://image.ntua.gr/iva/tools/mfd/fig/biarcs_part_res.png`

Contact

Yannis Avrithis
E-mail: `iavr@image.ntua.gr`
Project home: `http://image.ntua.gr/iva/tools/mfd/`

References

[1] Y. Avrithis and K. Rapantzikos. The Medial Feature Detector: Stable Regions from Image Boundaries. In *Proceedings of International Conference on Computer Vision*, Barcelona, Spain, November 2011.