Image processing in Perl graphic applications

Dmitry Karasik

Copenhagen University, 2003
The Protein Laboratory

C:

```
void
signal_handler( int signal)
{
   fprintf(“mama\n”);
   signal( signal, SIG_ACK);
}

...

signal( SIGPIPE, signal_handler);
```

Perl:

```
$SIG{PIPE} = sub {
   warn “Who cares?\n”;
};
```

Prima - a Perl GUI toolkit

http://www.prima.eu.org/

Prima is a general purpose extensible graphical user interface toolkit with a rich set of standard widgets and an emphasis on 2D image processing tasks. A Perl program using Prima looks and behaves identically on X11, Win32 and OS/2 PM.

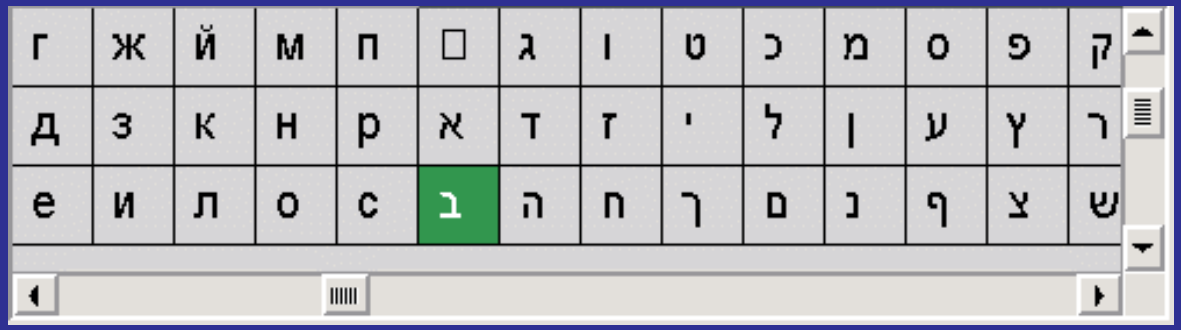
Prima includes:

- **Printing abstraction layer, including generic PostScript**

```
use Prima::PS::Printer;

$lp = Prima::PS::Printer-> create();

$lp-> begin_paint;
$lp-> text_out('Hello, world', 10, 10);
$lp-> end_paint;
```

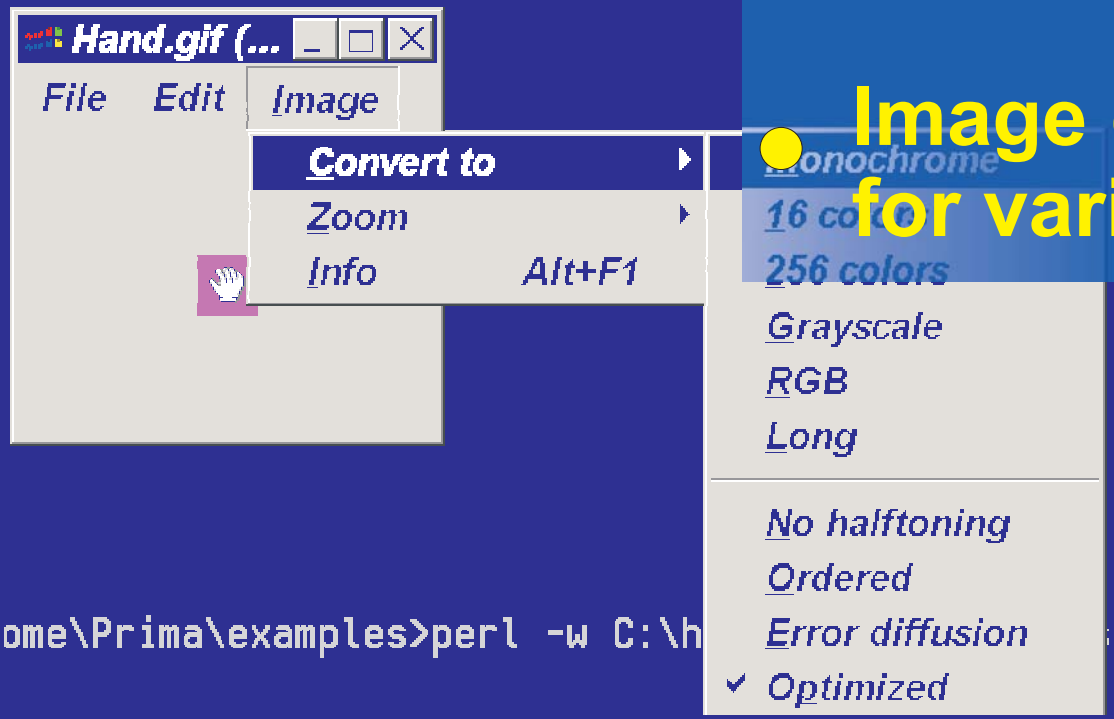
Prima includes:

- Printing abstraction layer, including generic PostScript
- **Unicode support**



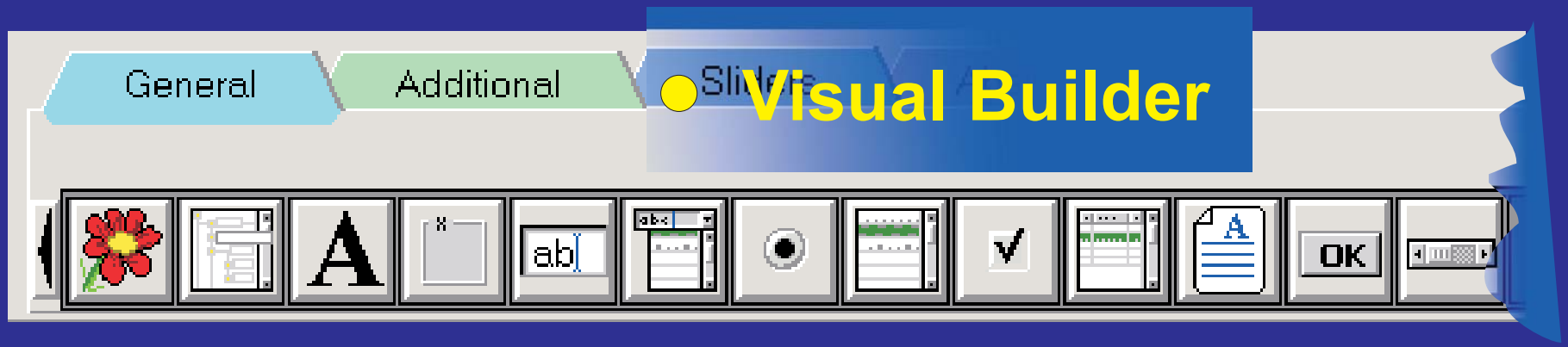
Prima includes:

- Printing abstraction layer, including generic PostScript
- Unicode Support
- **Image conversion routines for various pixel formats**

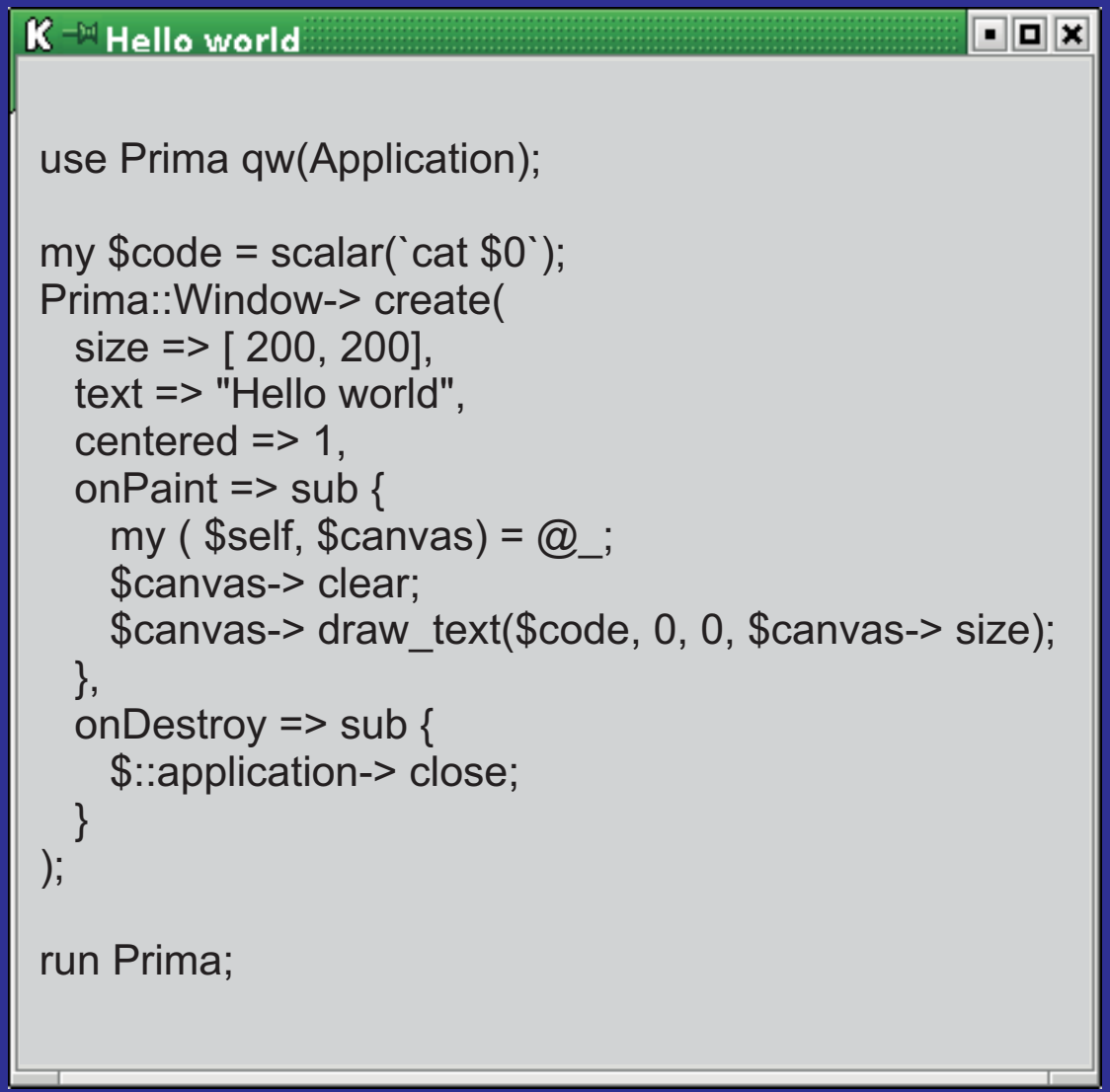


Prima includes:

- Printing abstraction layer, including generic PostScript
- Unicode Support
- Image conversion routines for various pixel formats



A “Hello, world” example



shades of gray

examples/fill.pl

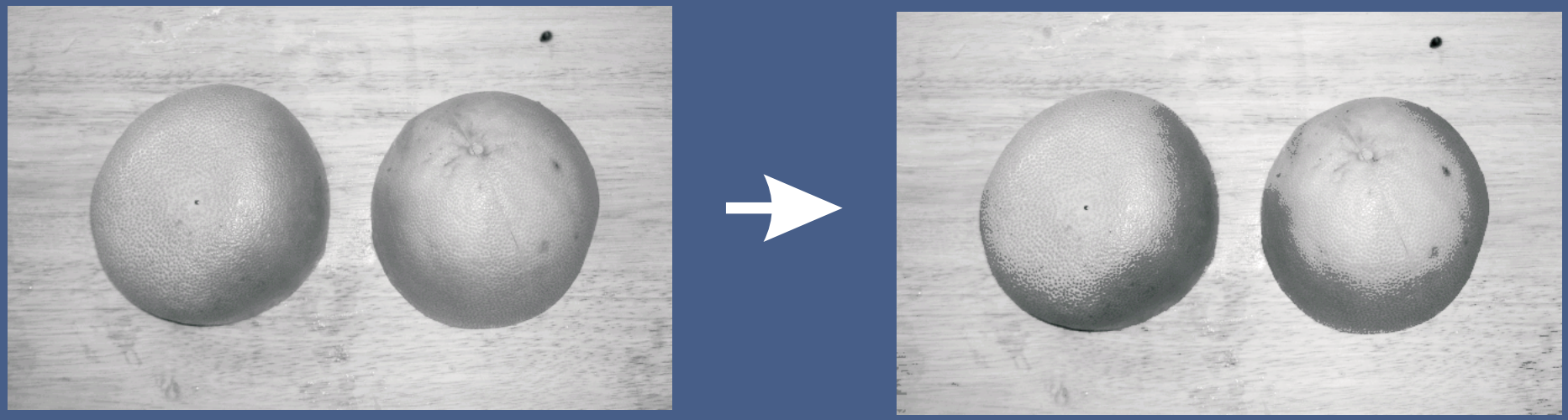
Image downsampling types

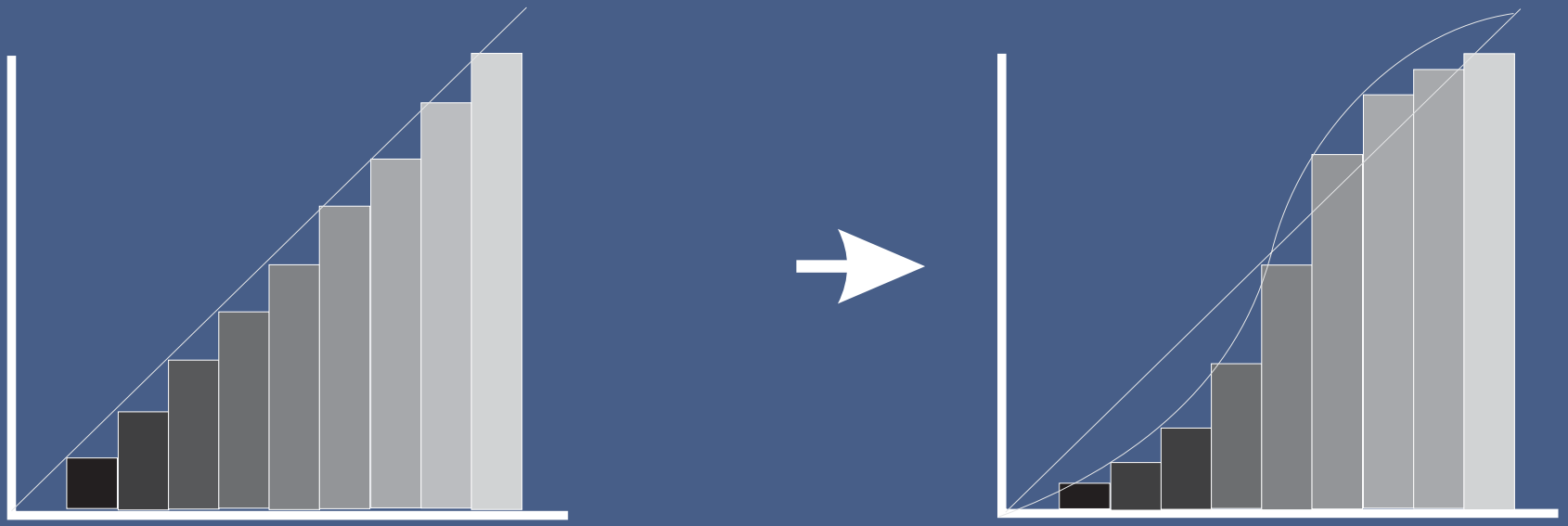
Image Processing Algorithms

http://www.prima.eu.org/IPA

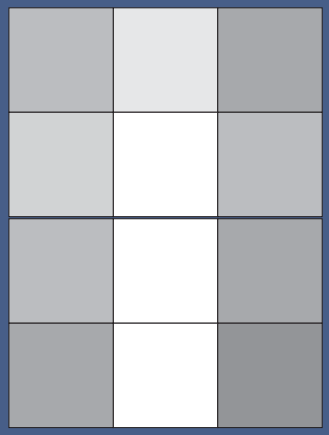
IPA stands for Image Processing Algorithms and represents the library of image processing operators and functions. IPA is based on the Prima toolkit (http://www.prima.eu.org), which in turn is a perl-based graphic library. IPA is designed for solving image analysis and object recognition tasks with perl.

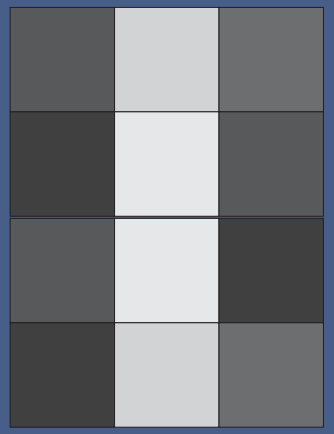
Contrast adjustment as a point operator



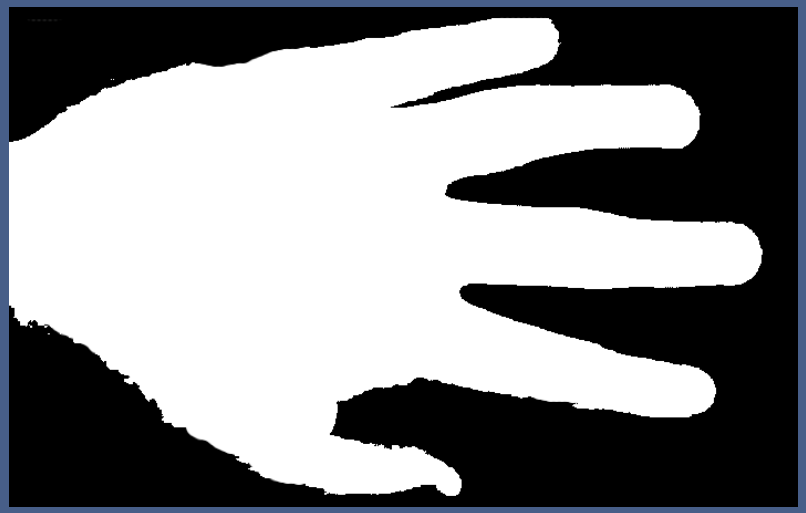


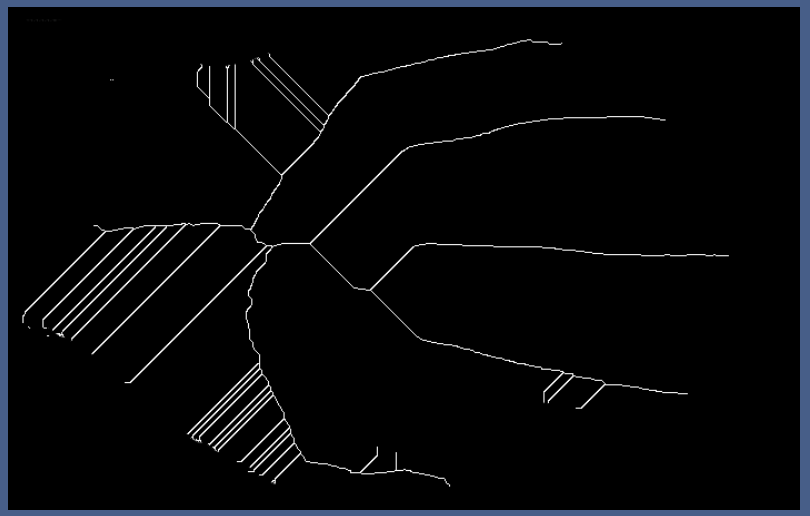
Edge finding as a local operator

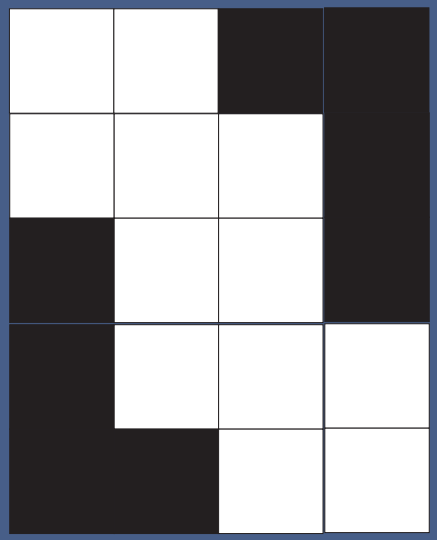




Skeletonizing as a morphology operator







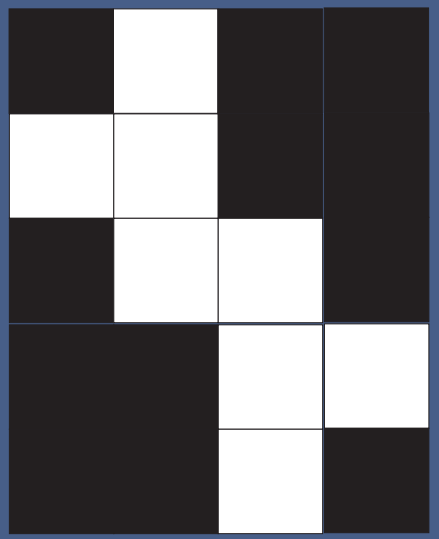
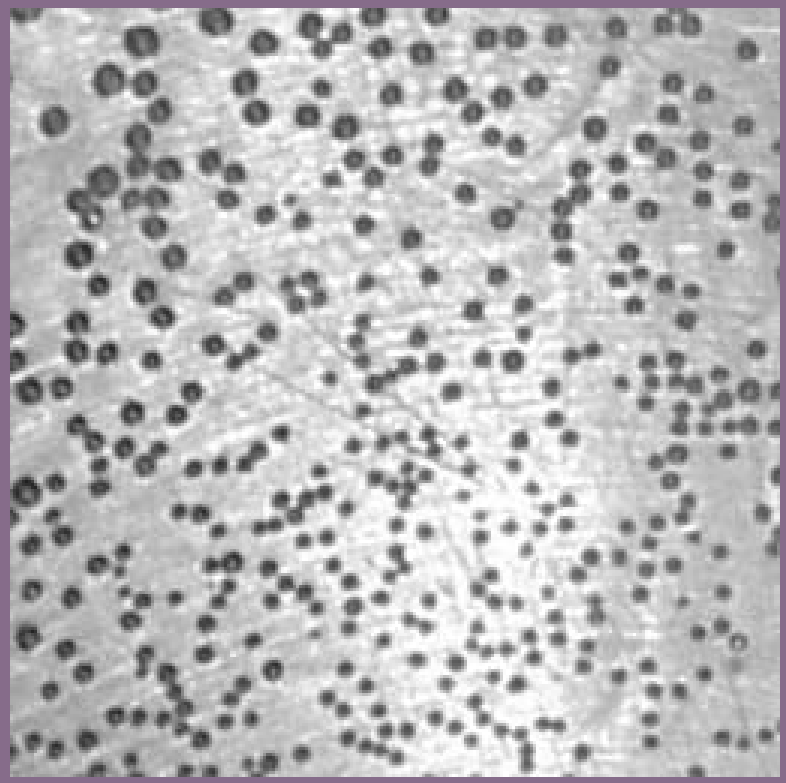


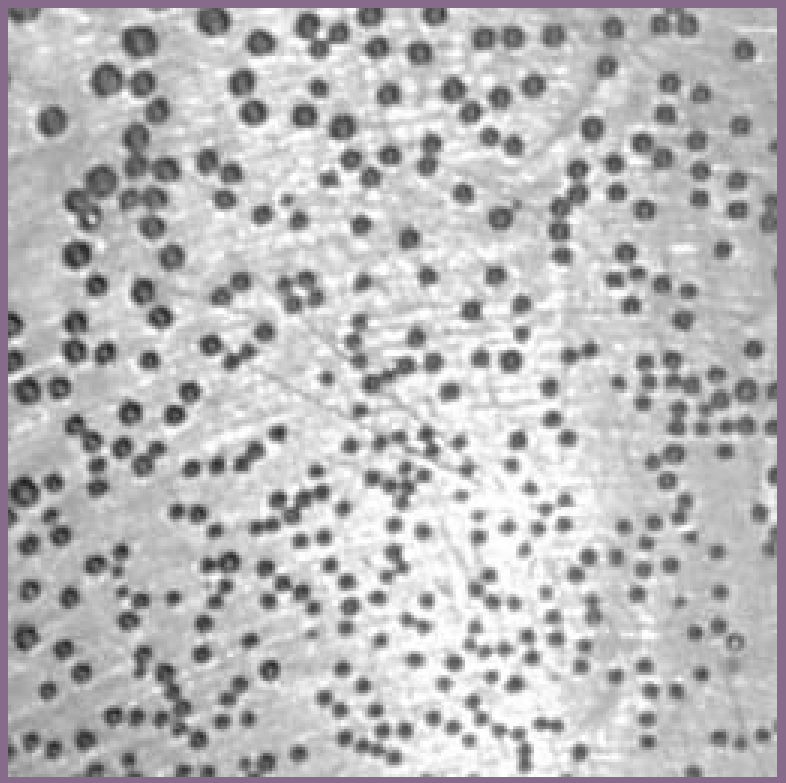
Image processing application example

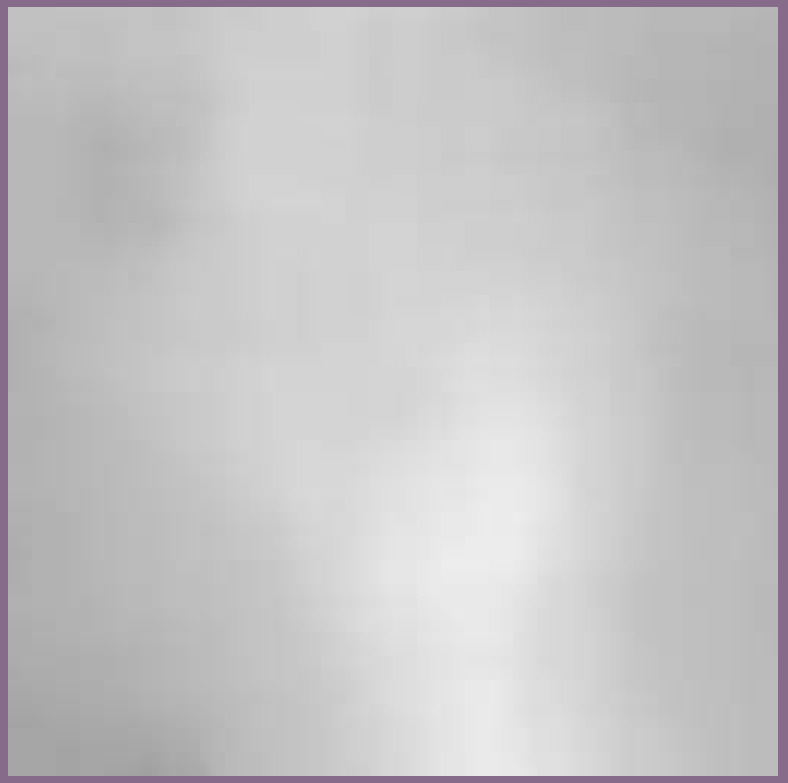
A. Original image



```
use Prima;
use IPA;
use IPA::Local;
use IPA::Point;
use IPA::Global;
use IPA::Morphology;

my $i = Prima::Image-> load('input.gif');
die "Cannot load:$@\n" unless $i;
```

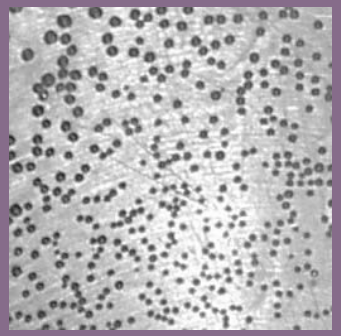
B. Median transform with large window

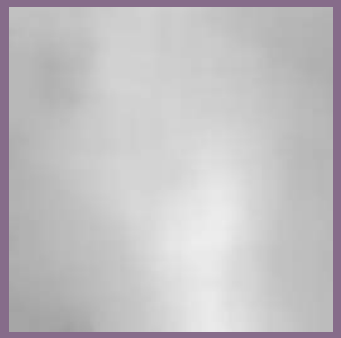


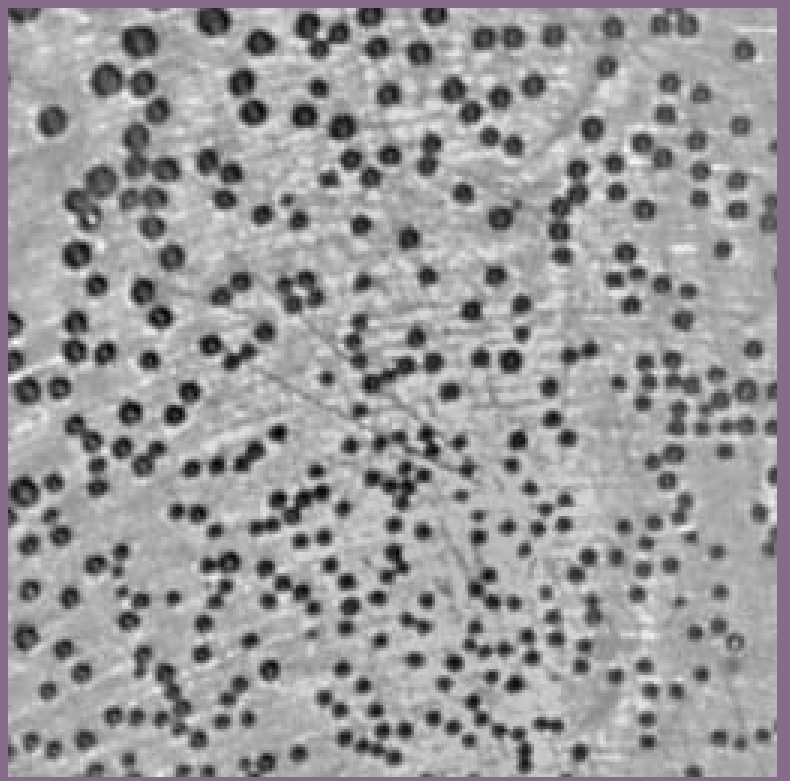


```
my $median = IPA::Local::median( $i, w => 51, h => 51);
```

C. subtract B from A to equalize the background

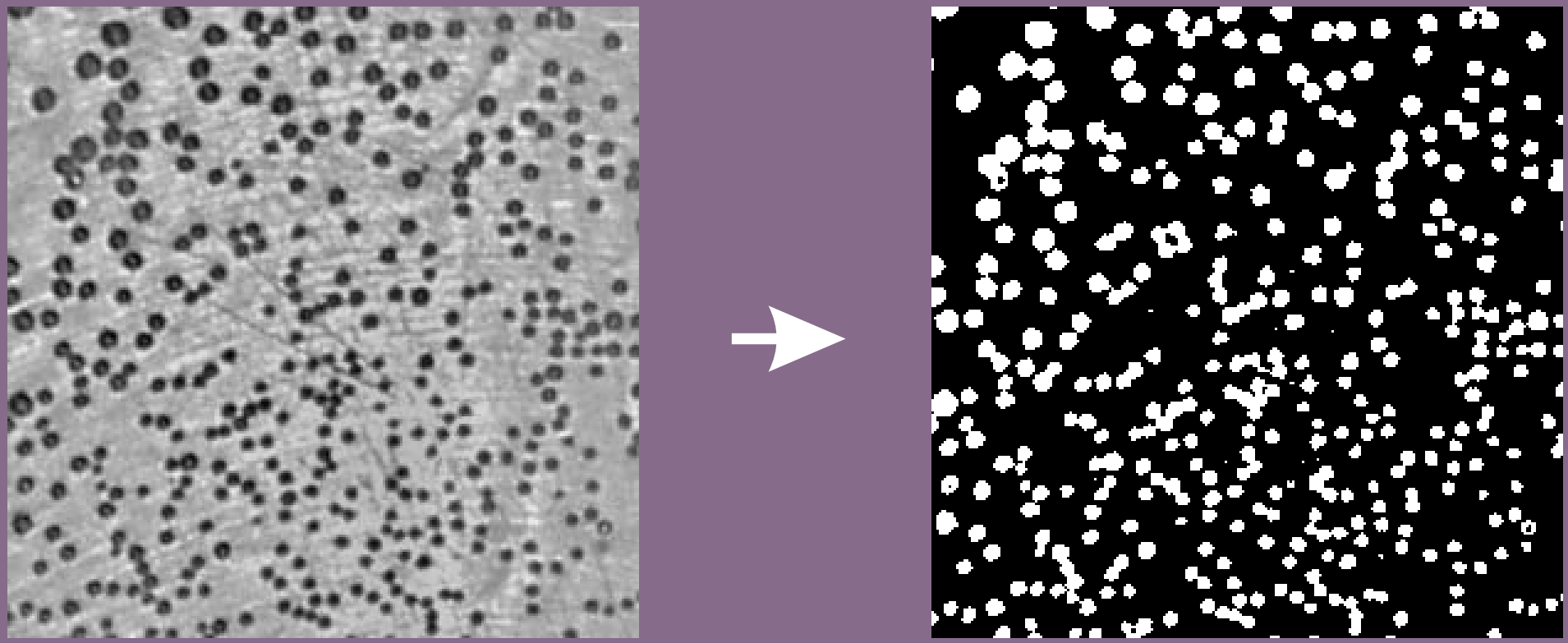






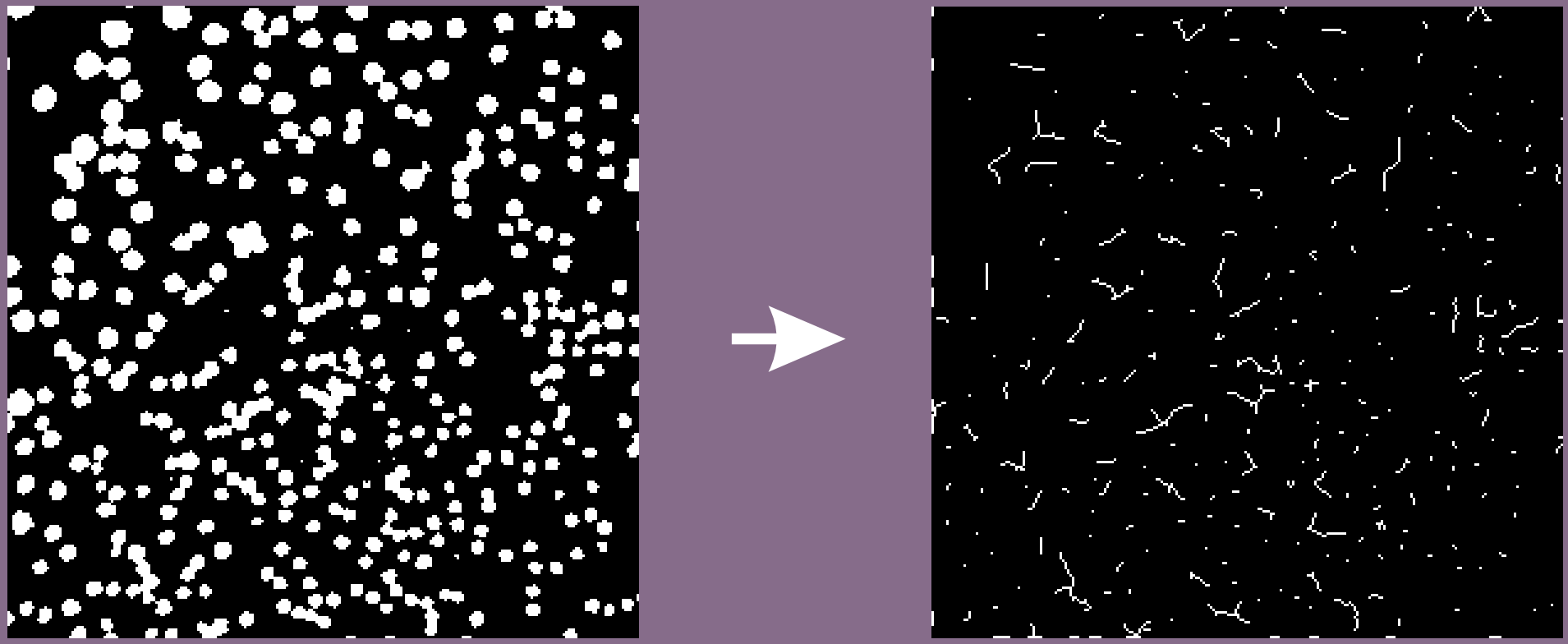
```
$i = IPA::Point::subtract( $i, $median);
```

D. Binary thresholding



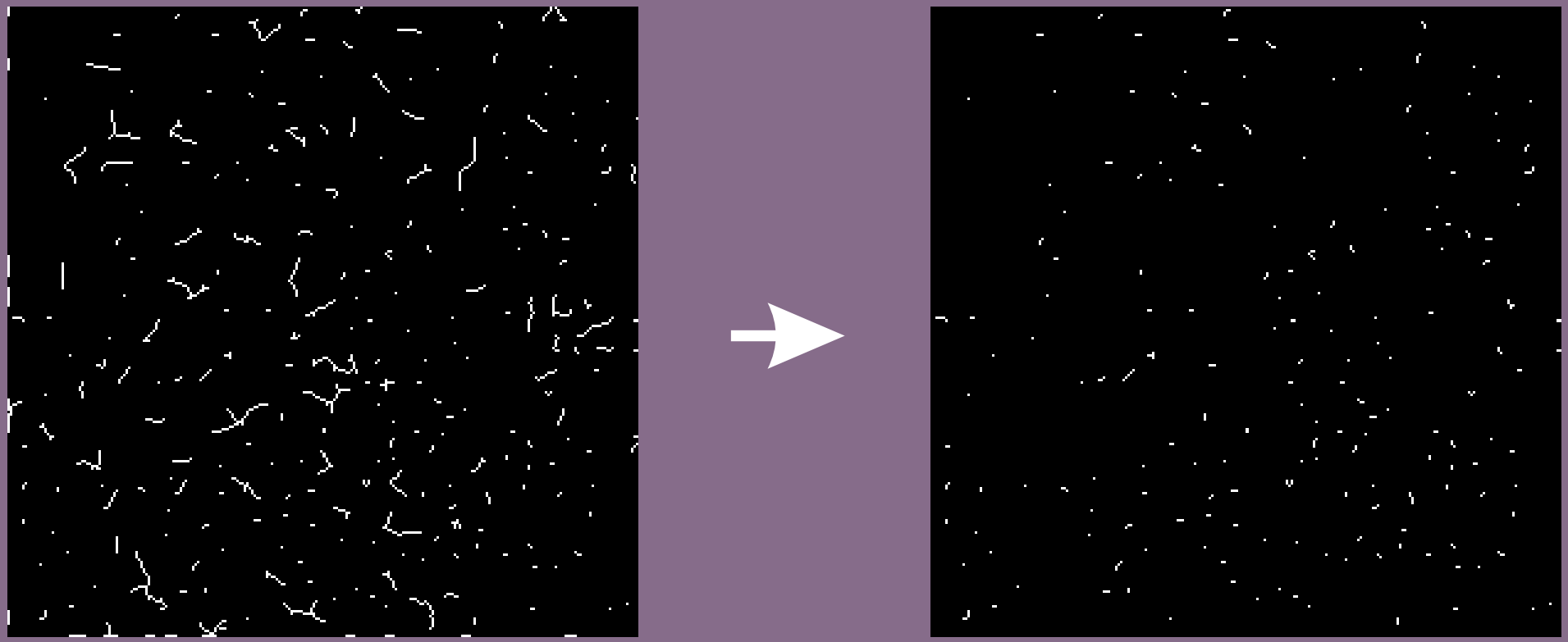
```
$i = IPA::Point::threshold( $i, minvalue => 0, maxvalue => 128);
$i = IPA::Global::fill_holes( $i);
```

E. Extract skeletons



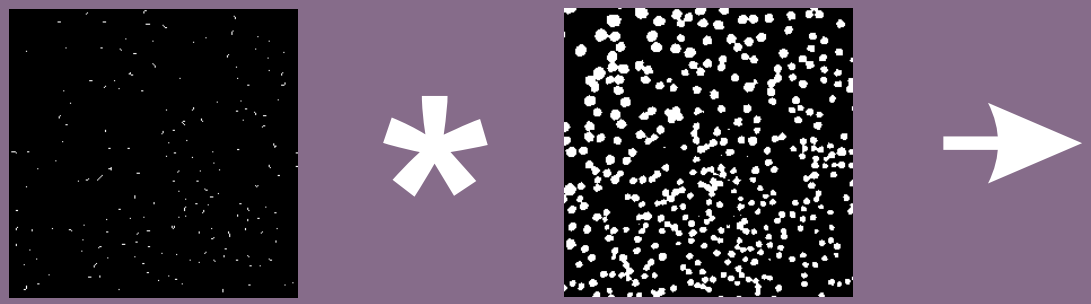
```
my $skeletons = IPA::Morphology::thinning( $i);
```

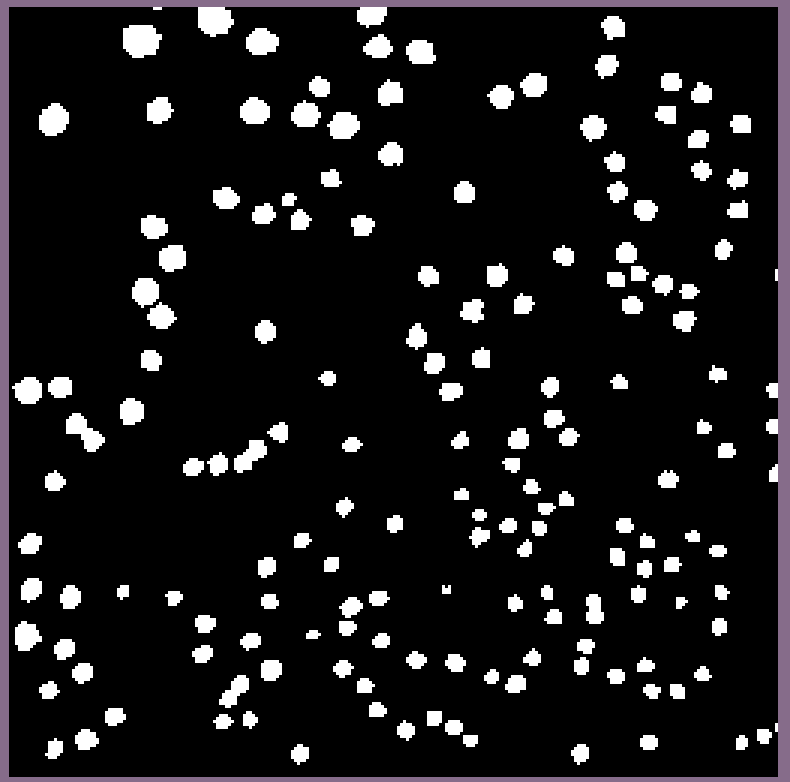
F. Filter large skeletons



```
$skeletons = IPA::Global::area_filter( $skeletons, maxArea => 5);
```

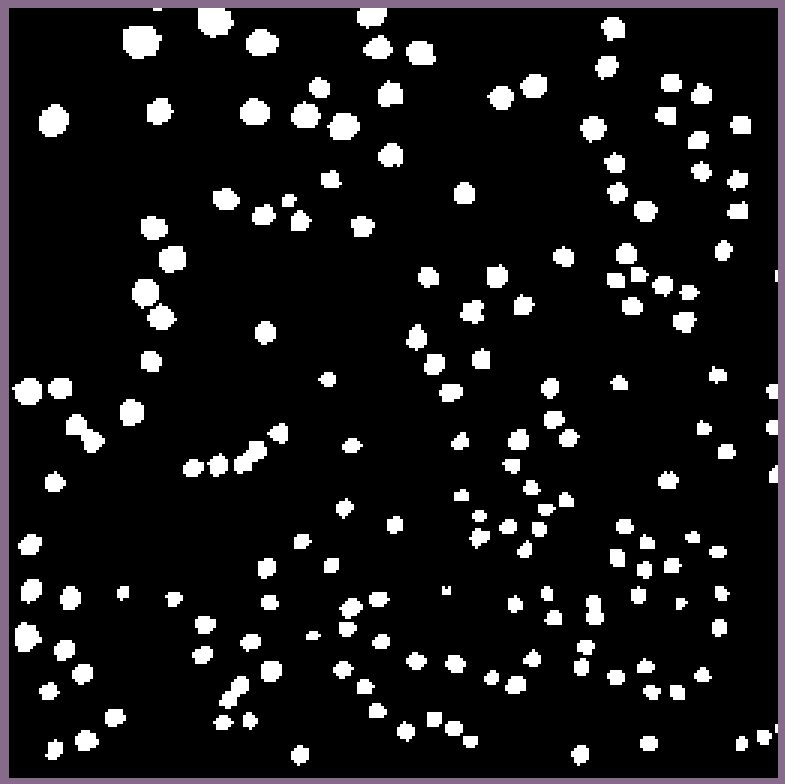
G. Reconstruct - filter out objects that have no corresponding skeletons





```
$i = IPA::Morphology::reconstruct( $i, $skeletons);
```

Calculate average bubble area



```
my $n = scalar @{IPA::Global::identify_contours( $i)};
my $area = $i-> sum / 255;
printf "Average area: %g pixels\n", $area / $n;
```

PDL - Perl Data Language

http://pdl.perl.org/

The perlDL project aims to turn perl into an efficient numerical language for scientific computing. The PDL module gives standard perl the ability to compactly store and speedily manipulate the large N-dimensional data sets which are the bread and butter of scientific computing. e.g. $a=$b+$c can add two 2048x2048 images in only a fraction of a second.

PDL image processing modules

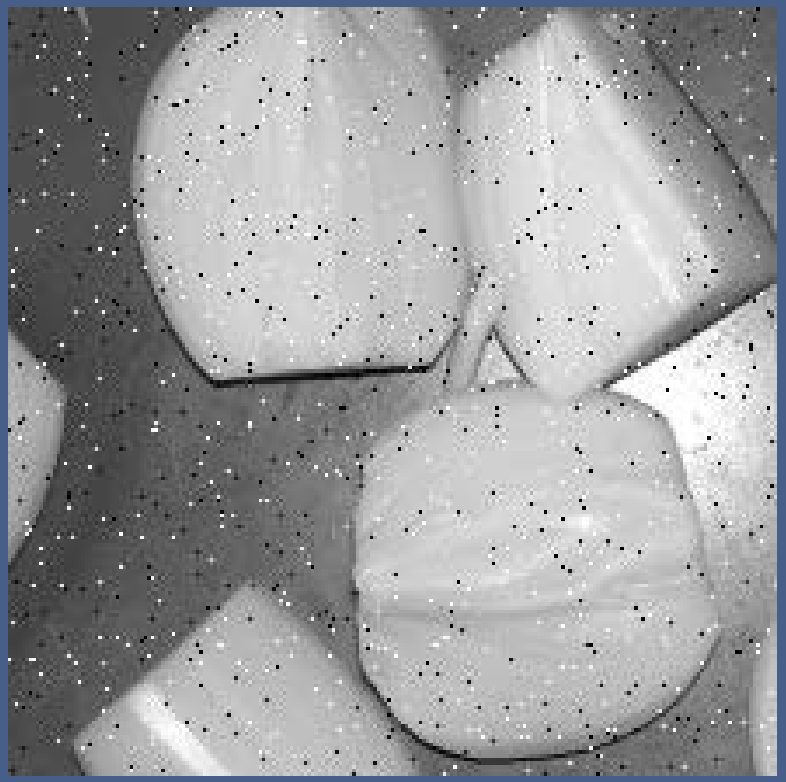
PDL::Graphics::LUT	look-up tables
PDL::Image2D	set of 2-D image processing functions
PDL::ImageND	set of N-D image processing functions
PDL::ImageRGB	utility functions for RGB images

PDL-PrimaImage

http://www.prima.eu.org/PDL-PrimaImage

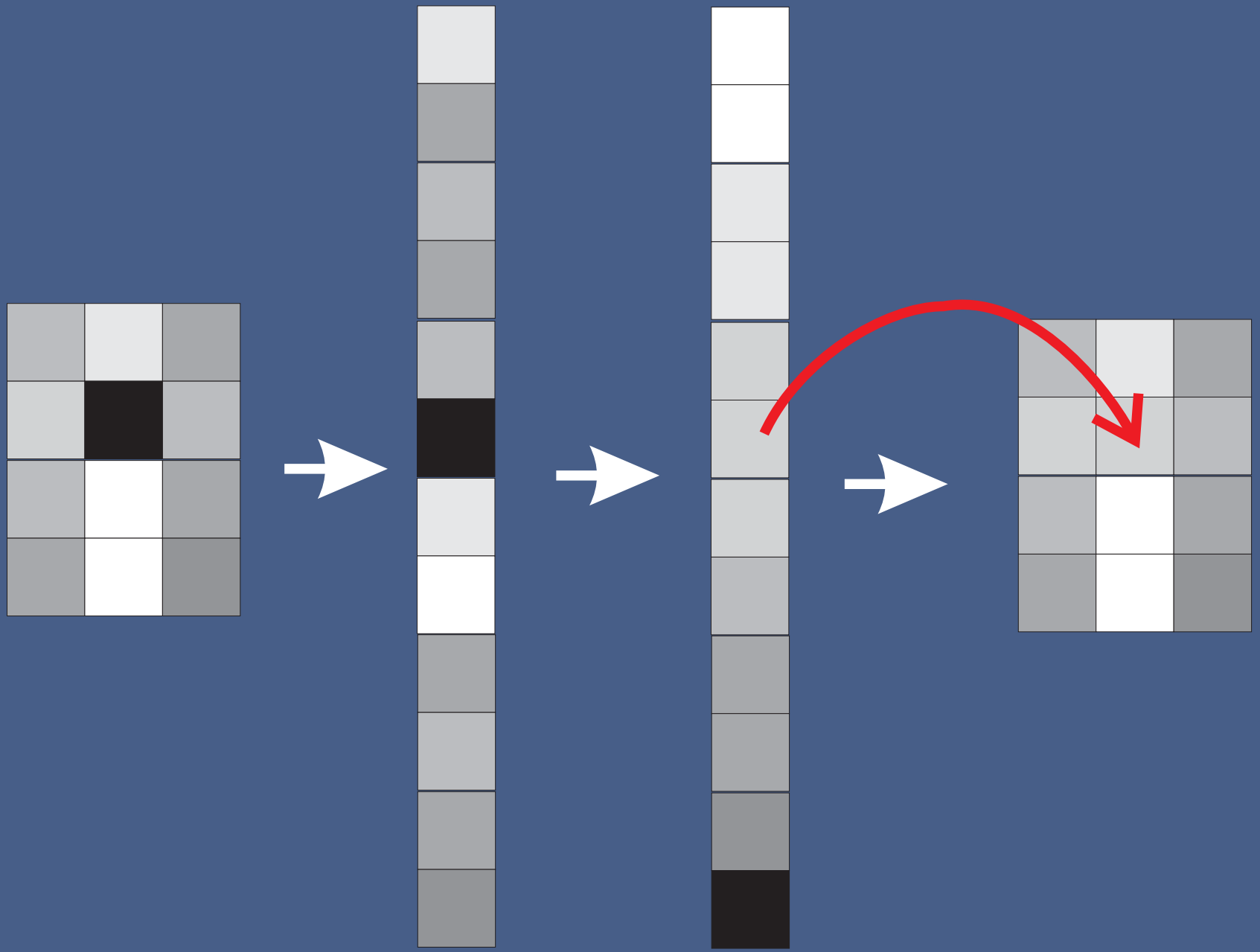
interface between PDL scalars and Prima images

Image processing application example: PDL and Prima



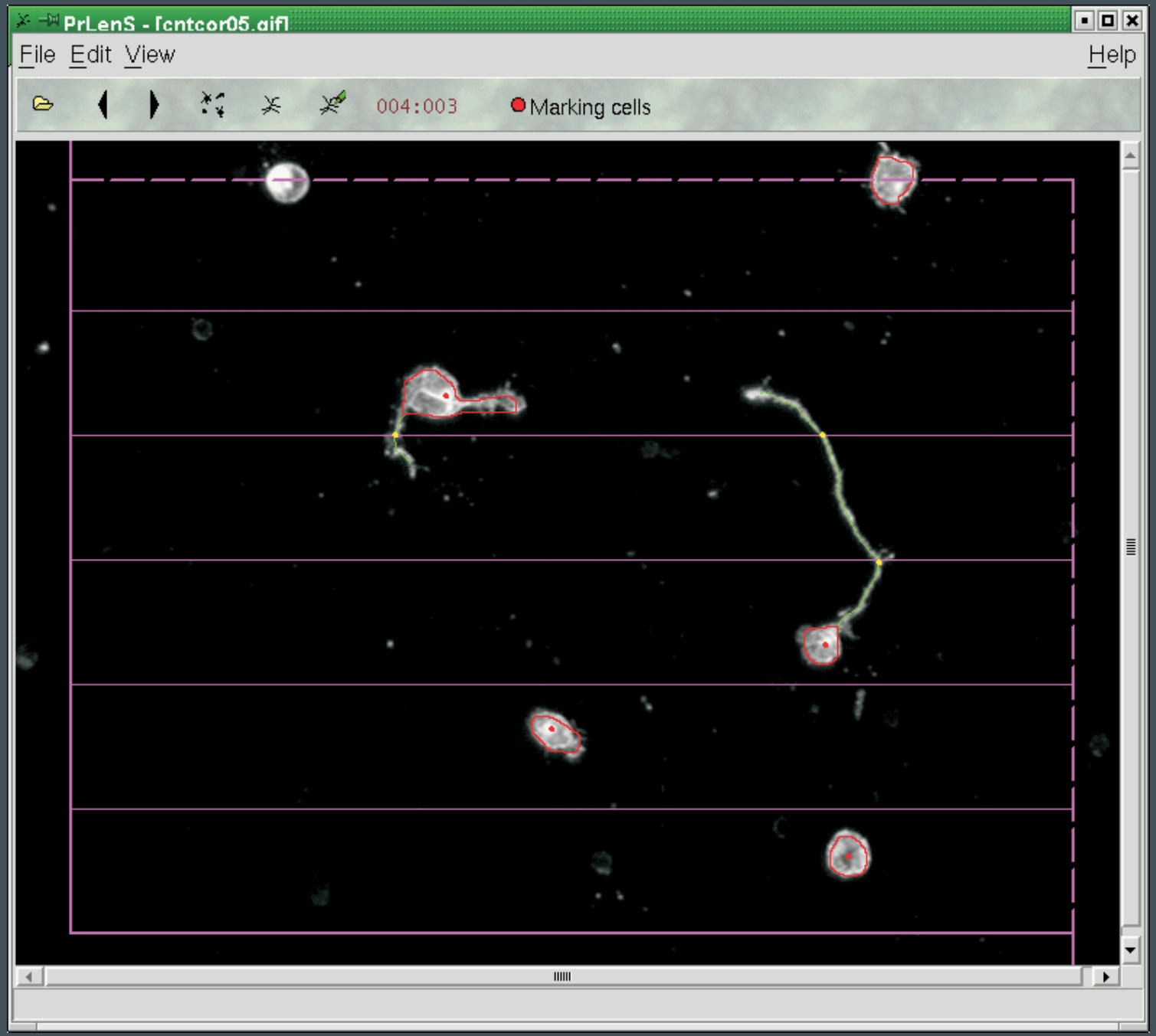
```
use PDL;
use PDL::Image2D;
use PDL::IO::Pic;
use Prima;
use IPA;
use IPA::Point;
use PDL::PrimaImage;

# read image as piddle scalar
my $img  = PDL-> rpic('onions.pgm');
# convert into Prima/IPA image
my $mask = PDL::PrimaImage::image( $img);
# map out all black and white pixels
$mask = IPA::Point::threshold( $mask, minvalue => 1, maxvalue => 254);
```

Median filter as a local operator



Median filter removes “salt-and-pepper” noise

```
# convert back to piddle scalar
$mask = PDL::PrimaImage::piddle( $mask);
# normalize values to 0 and 1
$mask /= -255;
$mask++;
# average by PDL::Image2D::patch2d
my $result = patch2d $img, $mask;
```


PrLenS - [cntcor05.gif]
File Edit View
Help
004:003
Marking cells

Links

Prima	http://www.prima.eu.org/
IPA	http://www.prima.eu.org/IPA
PDL	http://pdl.perl.org/

Authors

Dmitry Karasik - Ph.D. student at the Copenhagen University
Anton Berezin - Cybercity A/S, Copenhagen, Denmark
Vadim Belman - various unconfirmed locations