UTSouthwestern
Medical Center
Lyda Hill Department of Bioinformatics

BioHPC

Image Processing with MATLAB

Training will begin at 10:35AM – please mute your microphones!

19 Oct 2022

Overview

- Image Processing

- MATLAB and its image representation
- Image display

- Whole-image processing (filtering, quantifiers)
- Segmentation
- Using regions of interest (ROIs) for selective analysis and processing.
- Deconvolution and deblurring

- Closing Summary

As always, if you need assistance or information (or a nudge in the right direction!), feel free to contact biohpc-help@utsouthwestern.edu

MathWorks® -provided training and toolboxes

- MathWorks provides regular training webinars on their website – you can filter by topics of interest.
  - https://www.mathworks.com/company/events/search.html?s_tid=gn_ev_sch&q=image&fq[]=event_format:online&page=1

- Very good documentation and tutorials:
  - https://www.mathworks.com/help/overview/image-processing-and-computer-vision.html
  - https://www.mathworks.com/help/images/referencelist.html
  - Updated with every release of MATLAB – new functions and tools all the time.

here

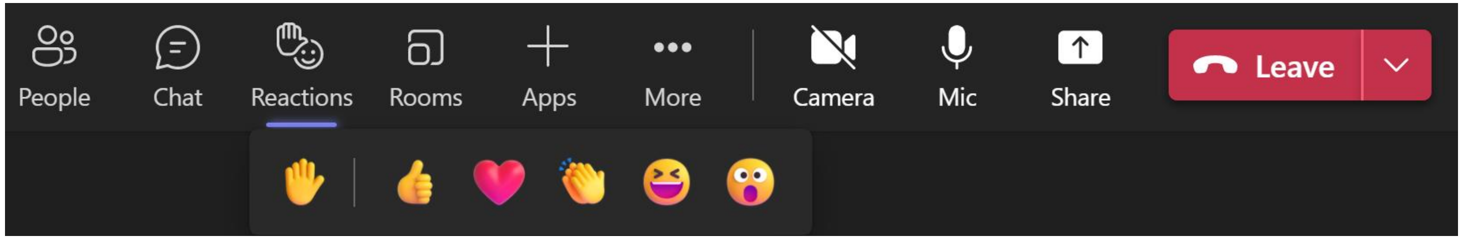
Follow along and supplementary materials for today’s training

- Each major section has an associated MATLAB Live Editor notebook, which will be noted up
  - Notebooks are only tested on 2020a/2021a!

- Notebooks can be found on the Portal Training page for October 19th, 2022.
  https://portal.biohpc.swmed.edu/content/training/training-slides/
  - Slides will be uploaded after the training.

- Some sections have interactive controls you can adjust and see how different parameters effect results.

- Use the ‘raise hand’ reaction in Teams if you want to ask a question, or write your questions in the chat



Basic Image Processing Concepts

- Image processing can mean multiple things:
  - **Image conditioning**
    - Changing the quality of the data at a low level
    - Filtering, interpolation
    - Data -> Better data
  - **Image analysis**
    - Deriving other, abstract features from your data
    - Histograms, mean/variance of a region
    - Data -> Information

- Image processing is **not** image interpretation.
  - Statistics and machine learning do that!

- Garbage in – garbage out is amplified in multi-step processing pipelines
  - Make sure each step is functioning as you expect before adding another step.

MATLAB on BioHPC

MATLAB is a scripting language and computational engine based around matrices as data.

- Effective for rapid prototyping of analyses and workflows
- Good ‘iteration loop’ for image processing and figure generation

It IS licensed and is not free to use. If you want to use MATLAB on BioHPC, you will be charged according to Information Resources’ policy regarding access to the UTSW license ($675).

- https://www.utsouthwestern.edu/research/core-facilities/research-software-core/operations.html#mat
- https://www.utsouthwestern.net/intranet/administration/information-resources/software-apps/matlab-faq.html
  - Add’l toolboxes will cost $200 – $500 per year
  - Strangely, doesn’t have the Medical Imaging Toolbox – contact IR if you think it should be added!

Most of the examples and functions shown here have additional parameters and arguments which can tune their behavior – ***highly*** suggested that you read the documentation for any you’re wanting to use.

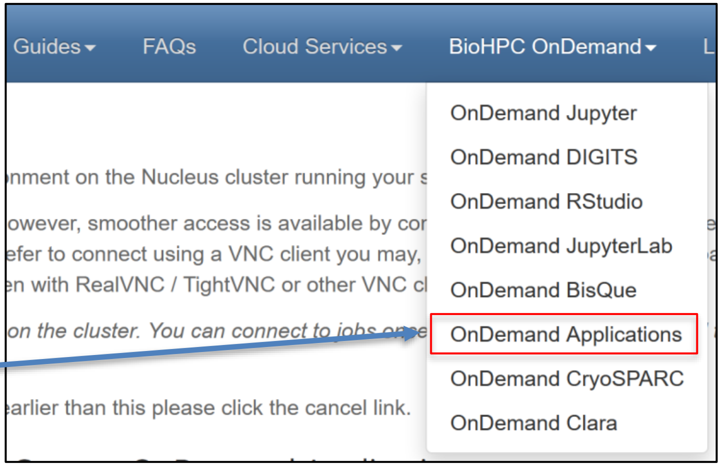
- https://www.mathworks.com/help/images/referencelist.html

Getting a MATLAB session

1. Go to https://portal.biohpc.swmed.edu/terminal/webgui/
2. Get an interactive job (32 GB is adequate for most cases)
3. Connect to your session
4. Right-click on the desktop -> **Open Terminal**
5. Execute **module add matlab/2021a** (or whichever release you like)
   - Some features and functions only exist in the recent releases.
6. Execute **matlab**
7. You're in!

Alternatively:

- https://portal.biohpc.swmed.edu/intranet/terminal/ondemand_apps/
- select 'matlab'
  - Only uses 2020a, currently.



nment on the Nucleus cluster running your s

owever, smoother access is available by cor
efer to connect using a VNC client you may,
en with RealVNC / TightVNC or other VNC cl

on the cluster. You can connect to jobs once

earlier than this please click the cancel link.

Download and open the .mlx notebooks provided on the training page if you want to follow along or experiment later.

Interactive image processing apps

- *imtool*
  - Good tool to quickly test out image analysis.
  - Documentation: https://www.mathworks.com/help/images/ref/imageviewer-app.html
  - Tutorial: https://www.mathworks.com/help/images/interact-with-images-using-image-viewer-app.html
- *volumeViewer*
  - Allows you to easily explore 3D data.
  - Documentation: https://www.mathworks.com/help/images/ref/volumeviewer-app.html
  - Tutorial: https://www.mathworks.com/help/images/explore-3-d-volumetric-data-with-volume-viewer-app.html
- *imageSegmenter*
  - Allows you to segment data using many different methods
  - Documentation: https://www.mathworks.com/help/images/ref/imagesegmenter-app.html
  - Tutorial: https://www.mathworks.com/help/images/image-segmentation-using-the-image-segmenter-app.html
  -
- Others: DICOM browser, Image Region Analyzer, Color Thresholder…
  - https://www.mathworks.com/help/images/referencelist.html?type=app

How MATLAB thinks about images

- MATLAB makes the assumption that images are matrices of some data type held in memory.

- Common types
  - **logical** (binary/Boolean) – [0,1]
  - **uint8** (unsigned 8-bit) – [0,255]
  - **uint16** (unsigned 16-bit) – [0,65535]
  - **double** (double-precision floating point) – Decimal numbers (e.g. 2.2251e-308, 0.4, 0.333333…)

- Important to know your data and its format
  - Typical values
  - Dynamic range

- This is *not* the same as the file format that contains your data (DICOM, TIFF, ~~JPEG~~)

https://www.mathworks.com/help/matlab/creating_plots/image-types.html

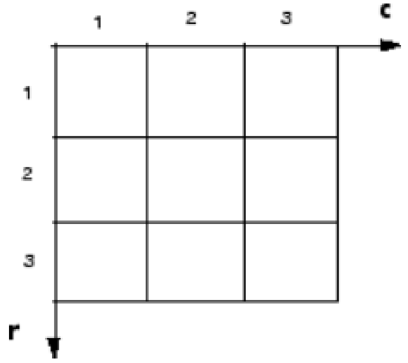
How MATLAB thinks about images – Coordinate Systems

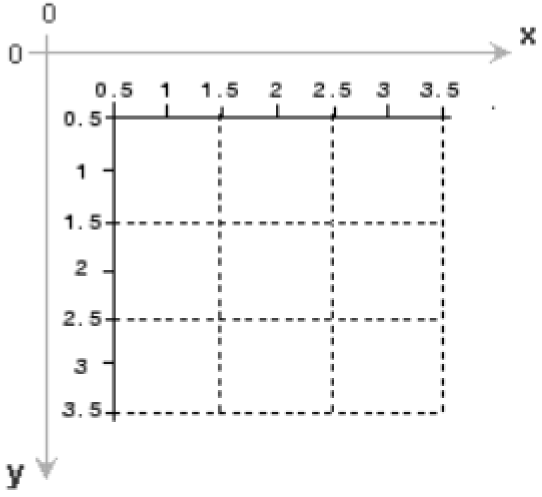
- MATLAB assumes images are matrices, so (1,1) is in the upper left-hand corner of the image.
  - Coordinates are (row,column)

- We often think of the origin as 0,0, going right and up (1st quadrant)
  - Coordinates (X,Y)

- How MATLAB internally holds the data and how it shows it to you are different.
  - **Different functions can assume different coordinates**

- You can add XData and YData to your images -> usually a good idea if you're wanting quantitative size information





https://www.mathworks.com/help/images/image-coordinate-systems.html

MATLAB Graphics/Figures/Axes documentation

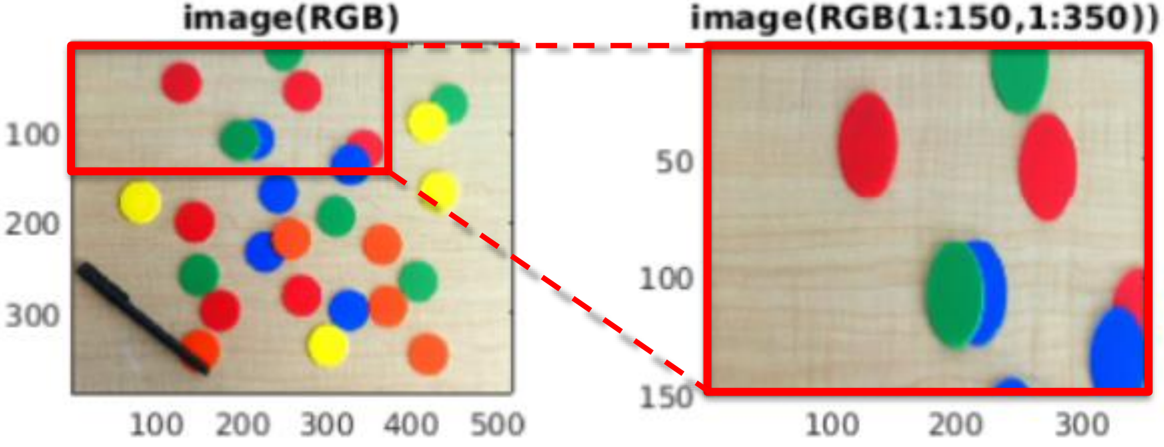
- MATLAB is a strongly object-oriented language - understanding **objects** vs. **handles** (AKA pointers) can be very helpful when trying to display graphics the way you want.
  - **Not necessary to know in-depth – just be aware!**
  - https://www.mathworks.com/help/matlab/graphics-objects.html
  - https://www.mathworks.com/help/matlab/matlab_oop/handle-objects.html

- Graphics in general:
  - https://www.mathworks.com/help/matlab/graphics.html

- Formatting axes (text, scale, colormaps)
  - https://www.mathworks.com/help/matlab/formatting-and-annotation.html

Image Coordinate Systems

image(rgb_chips)

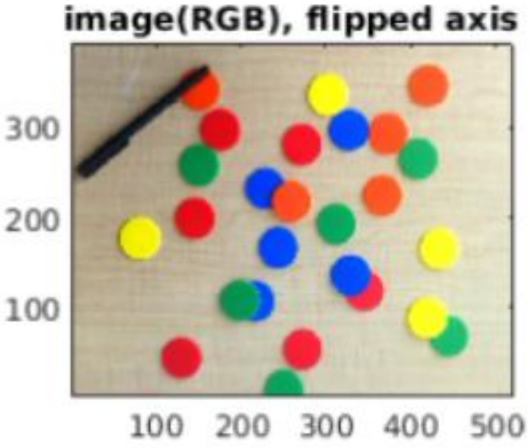


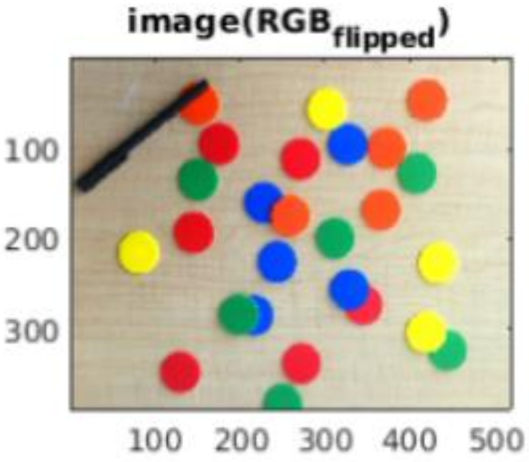
image(rgb_chips(1:150,1:350));

Number of rows: 150
Number of columns: 350

Flipping axis:

image(rgb_chips);
set(gca,’YDir’,’normal’);





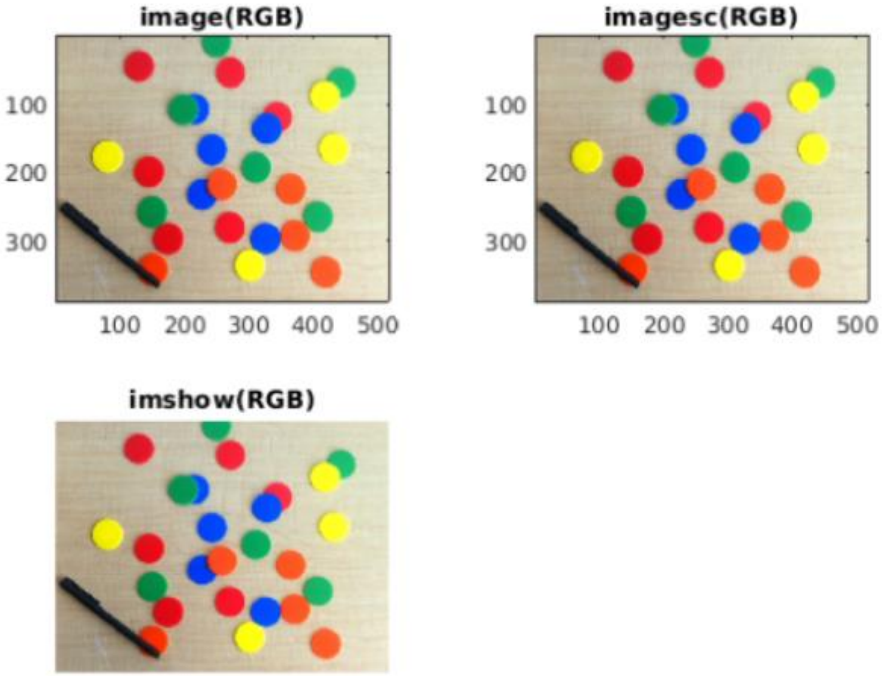
Flipping data:

image(flipud(rgb_chips));

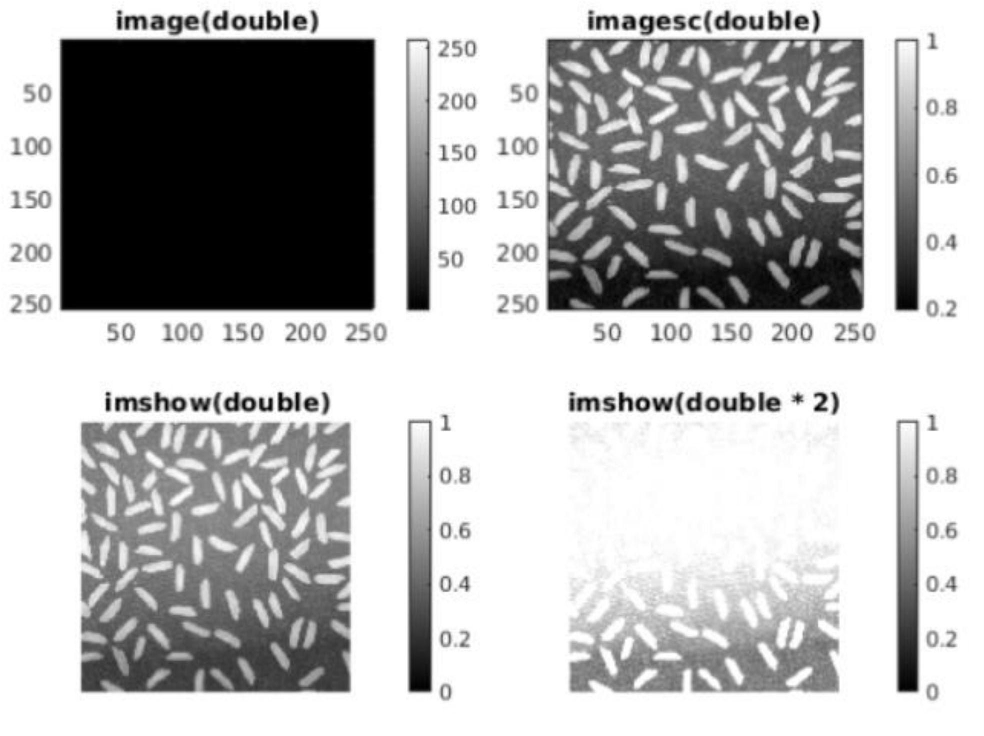
basics.mlx

Image display varies depending on data type

3 channel *uint8*

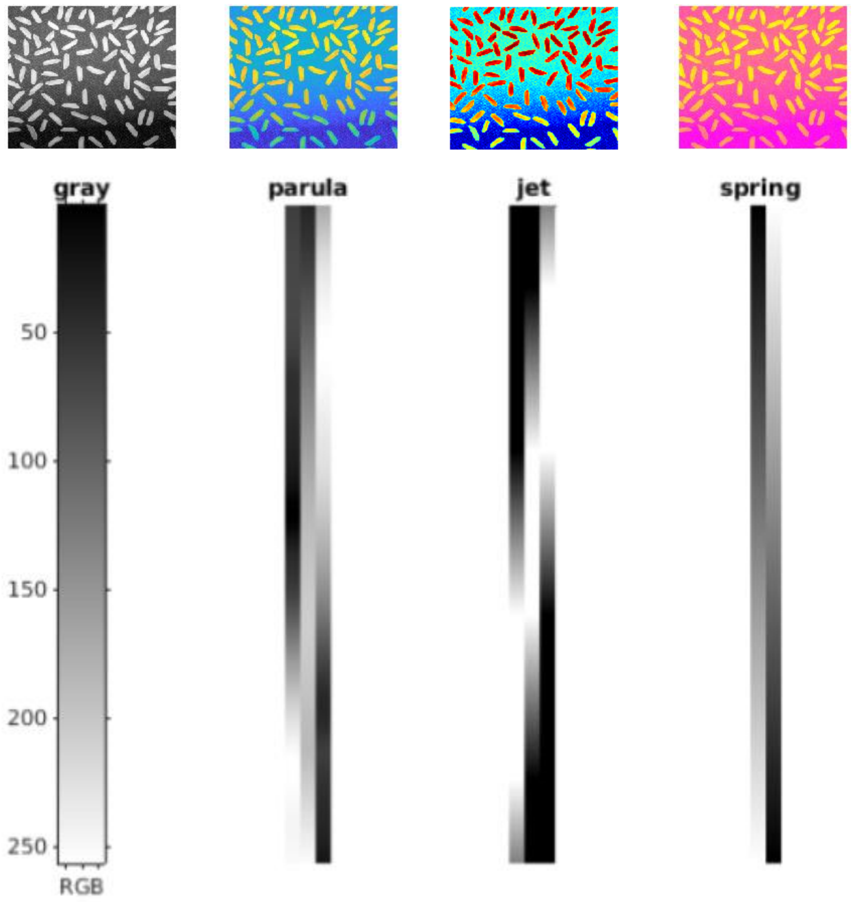


1 channel *double*



https://www.mathworks.com/help/matlab/creating_plots/image-types.html

Colormaps



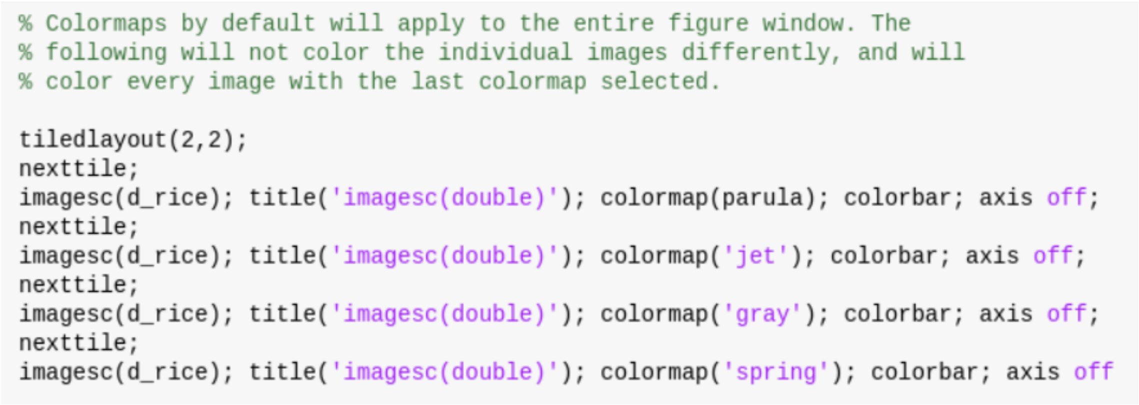
Colormaps are Nx3 matrices that contain RGB triplets which code for a certain grey level or index. Default is 256 levels.

- Many options to choose from.
  - https://www.mathworks.com/help/matlab/ref/colormap
- You can create your own quite easily!
- Create/modify interactively using *colormapeditor*

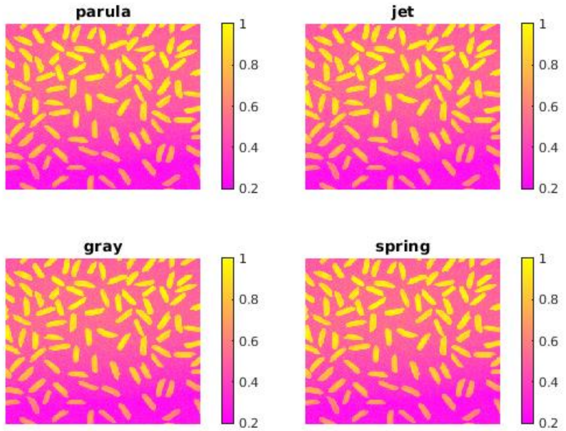
- If you use a colormap, it is good practice to add a *colorbar* to show which color should be interpreted as which value.
- https://www.mathworks.com/help/matlab/creating_plots/change-color-schemes-using-a-colormap.html

Colormaps apply to whole figures by default

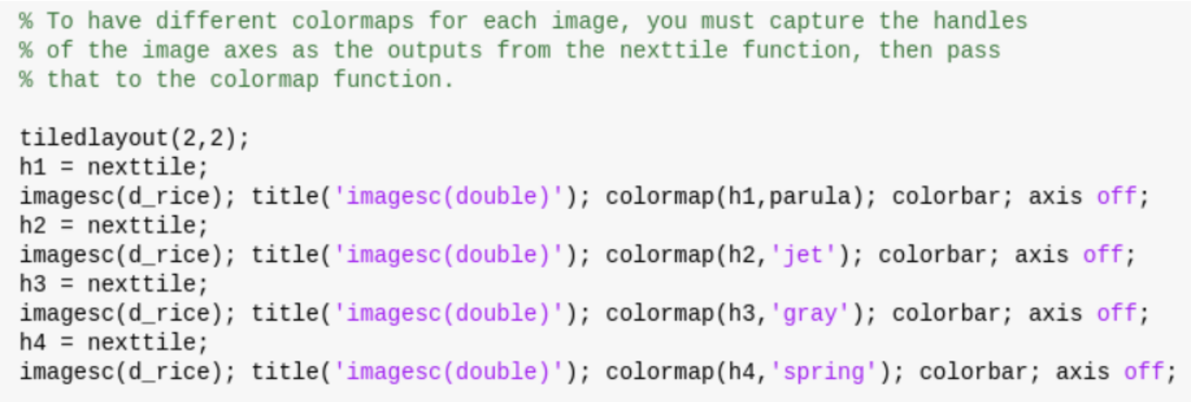
Incorrect



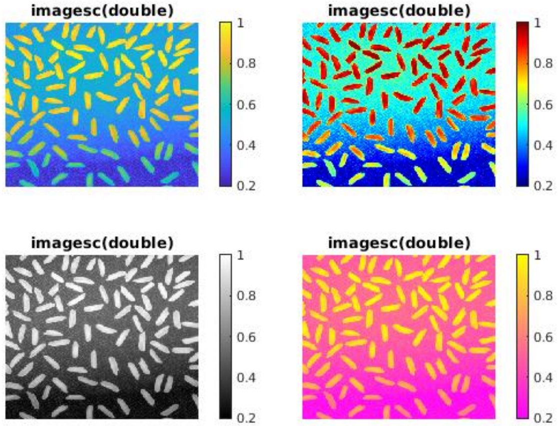
```
% Colormaps by default will apply to the entire figure window. The
% following will not color the individual images differently, and will
% color every image with the last colormap selected.

tiledlayout(2,2);
nexttile;
imagesc(d_rice); title('imagesc(double)'); colormap(parula); colorbar; axis off;
nexttile;
imagesc(d_rice); title('imagesc(double)'); colormap('jet'); colorbar; axis off;
nexttile;
imagesc(d_rice); title('imagesc(double)'); colormap('gray'); colorbar; axis off;
nexttile;
imagesc(d_rice); title('imagesc(double)'); colormap('spring'); colorbar; axis off
```



Correct



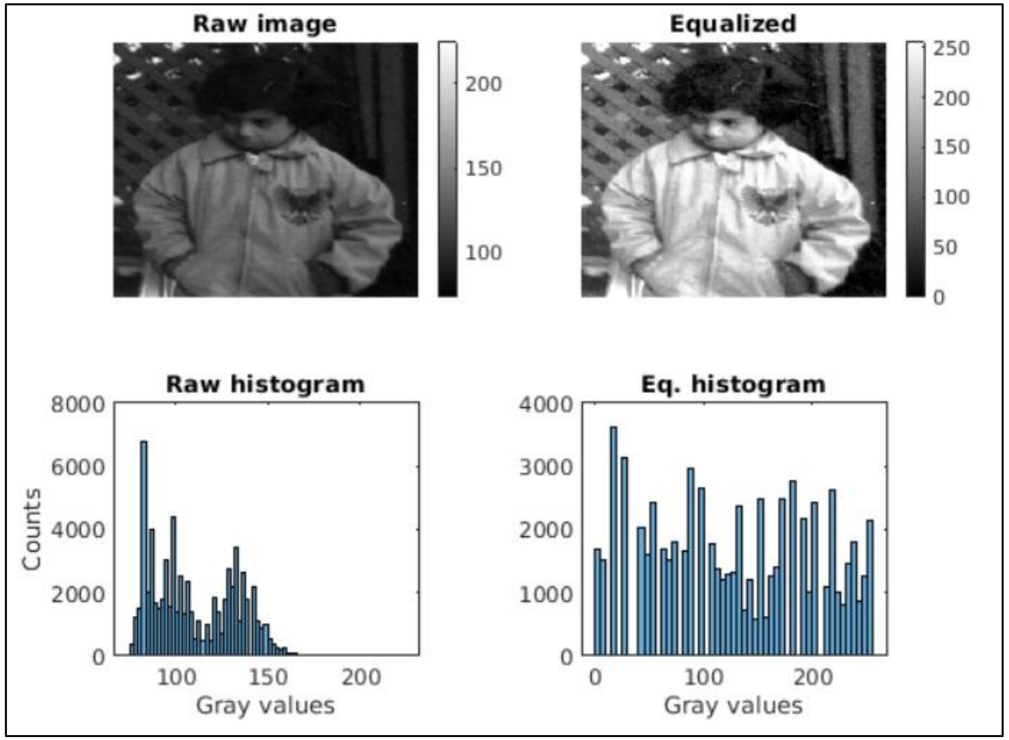
```
% To have different colormaps for each image, you must capture the handles
% of the image axes as the outputs from the nexttile function, then pass
% that to the colormap function.

tiledlayout(2,2);
h1 = nexttile;
imagesc(d_rice); title('imagesc(double)'); colormap(h1,parula); colorbar; axis off;
h2 = nexttile;
imagesc(d_rice); title('imagesc(double)'); colormap(h2,'jet'); colorbar; axis off;
h3 = nexttile;
imagesc(d_rice); title('imagesc(double)'); colormap(h3,'gray'); colorbar; axis off;
h4 = nexttile;
imagesc(d_rice); title('imagesc(double)'); colormap(h4,'spring'); colorbar; axis off;
```

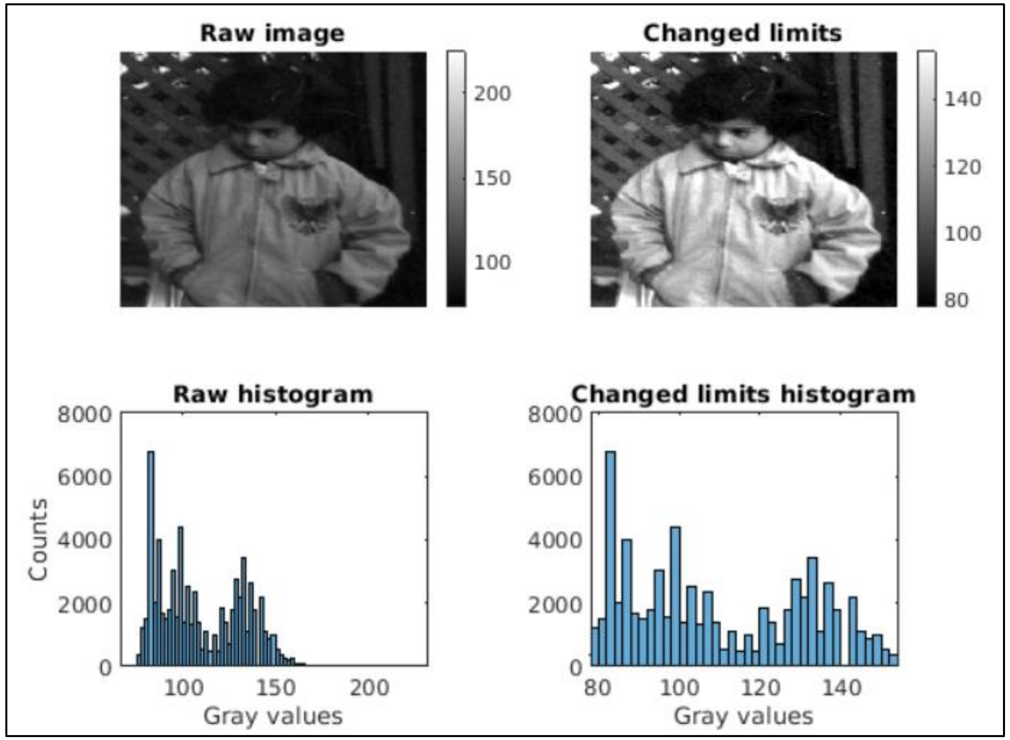


basics.mlx

Image Presentation – Changing the Data vs Changing the Display

You often have the choice to change your data or your display – when you can, change your display so you don’t affect your data.



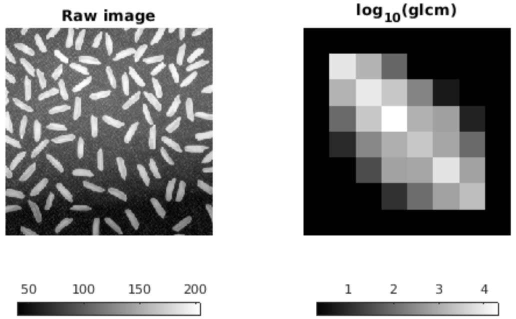


Many contrast adjustment options exist: https://www.mathworks.com/help/images/contrast-adjustment.html

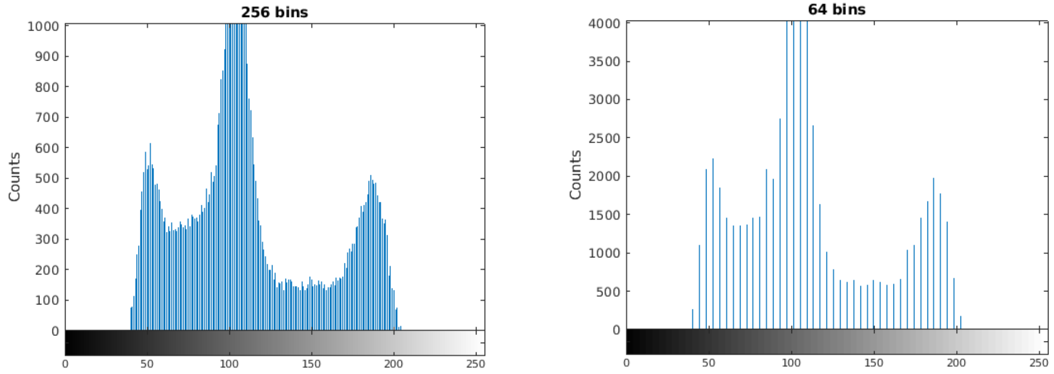
Whole-image quantifiers and processing

Many functions apply to whole images or pairs of images.

- *graycomatrix*
  - Generate gray-level co-occurrence matrix (GLCM), a measure of image texture
- *graycoprops*
  - Given a GLCM, derives numerical statistics from it.
- *imhist*
  - Generates a histogram for the image.
  - Can specify number of bins
- *entropy*
  - Calculates the entropy of the image, a measure of disorder/randomness
- etc.



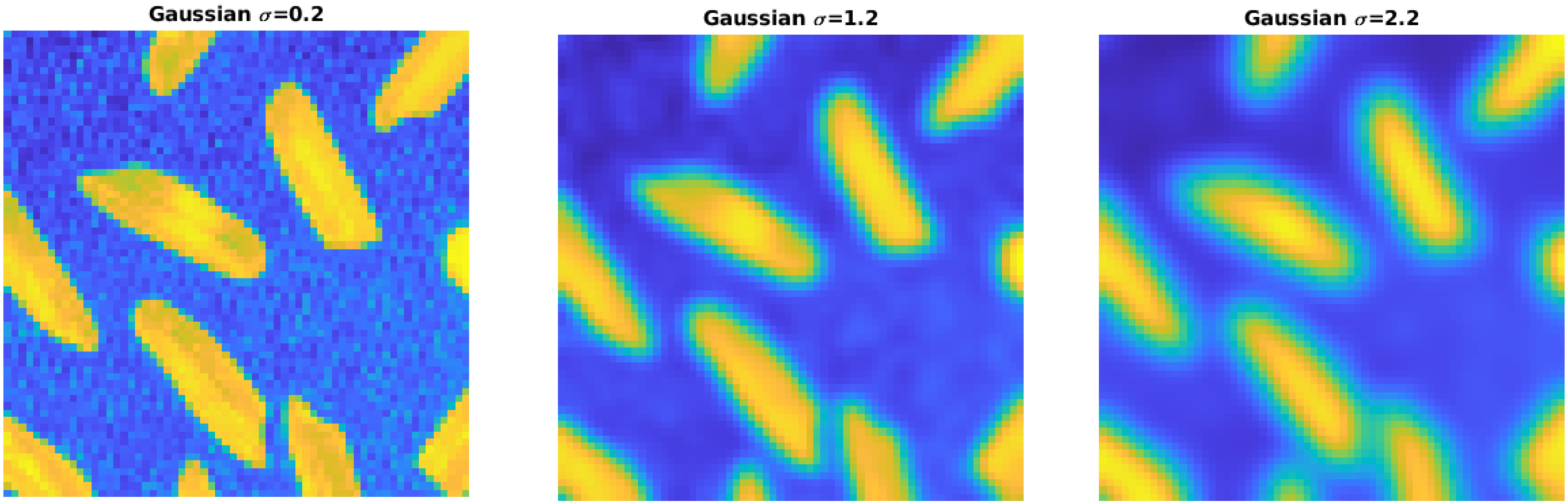
```
   Contrast: 0.3463
Correlation: 0.9053
     Energy: 0.1544
Homogeneity: 0.8757
```



whole_images.mlx

Whole image filtering

Filtering is the process of estimating a data object from another. Typically we think of applying an operator to an image.



Gaussian σ=0.2

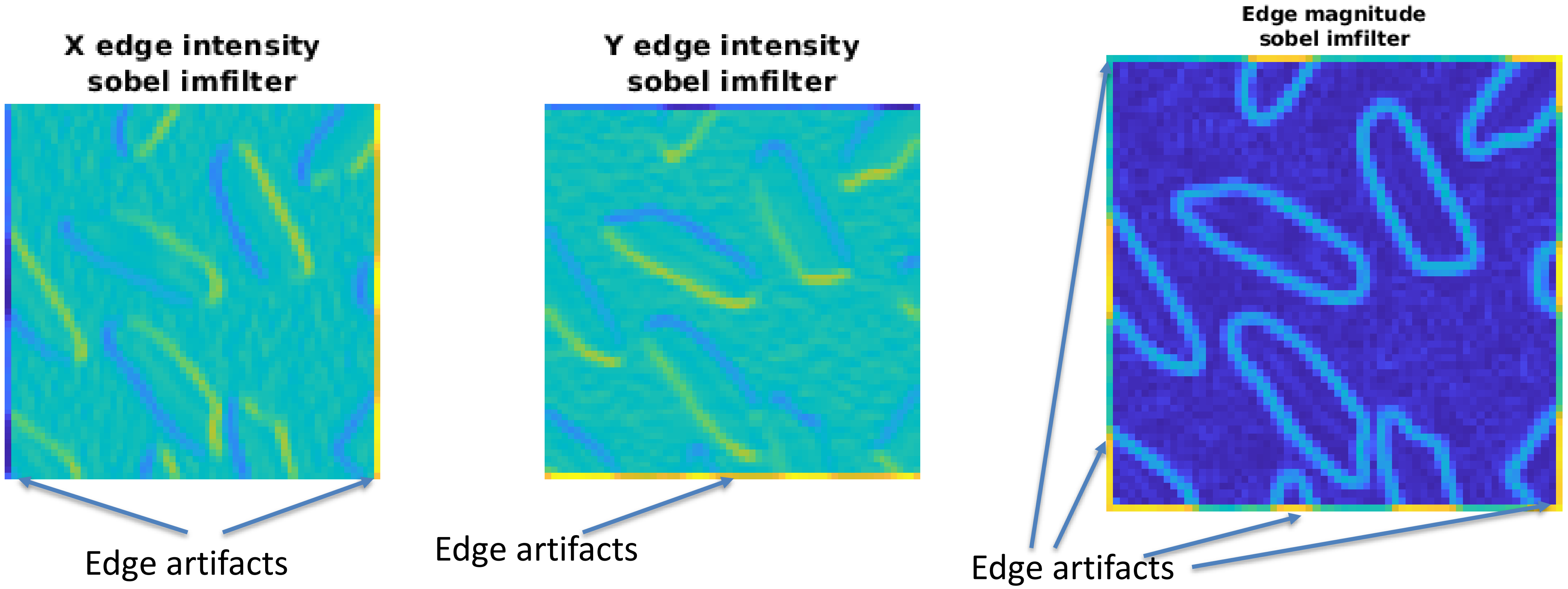
Gaussian σ=1.2

Gaussian σ=2.2

whole_images.mlx

Making your own image processing steps – edge filtering

- You can usually make your own processing steps, but this is sometimes prone to edge-cases and errors.



Edge artifacts

Edge artifacts

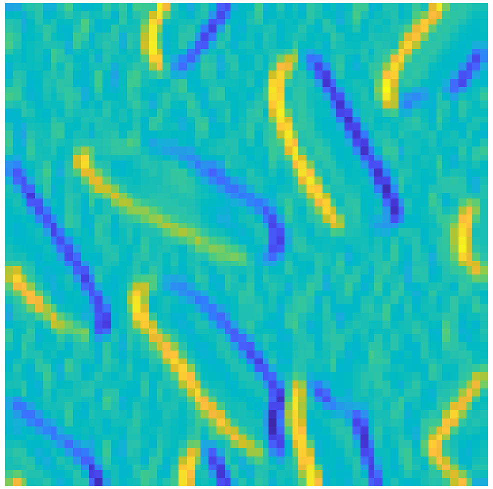
Edge artifacts

whole_images.mlx

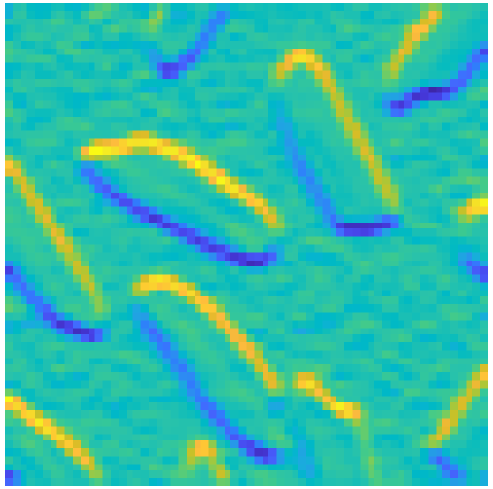
Using built-in gradient functions

- Check the documentation, see what’s available, lean on the tools that have already been developed.

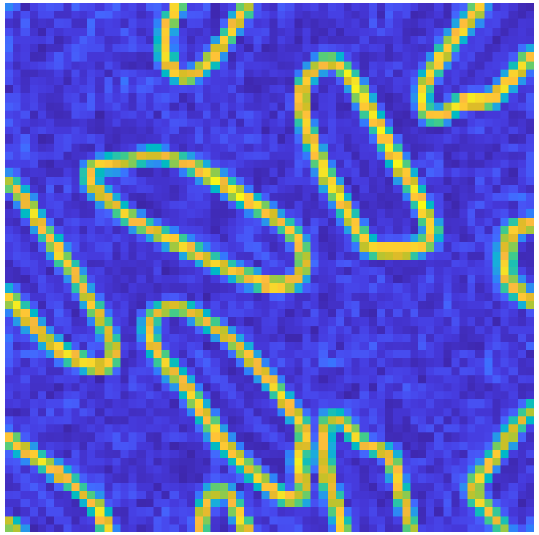
X edge intensity imgradientxy



Y edge intensity imgradientxy



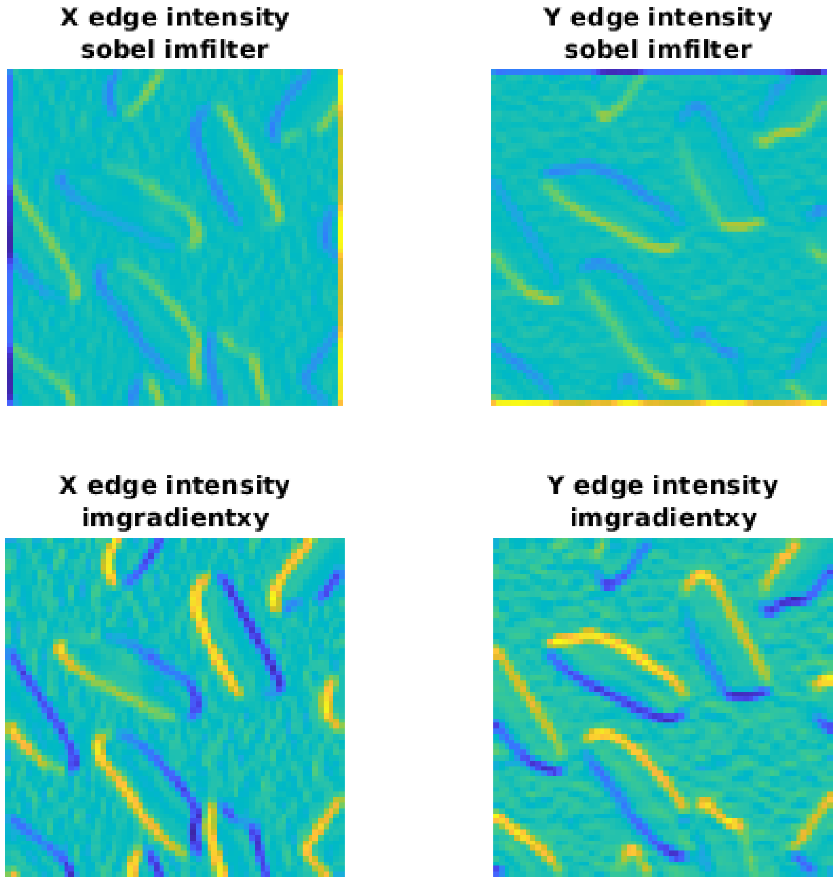
Edge magnitude imgradient



No edge artifacts!

whole_images.mlx

Make sure things do what you expect!

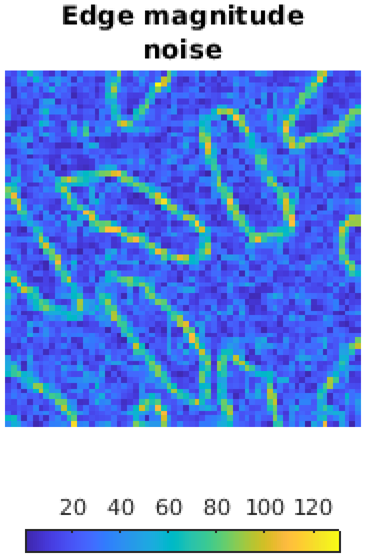


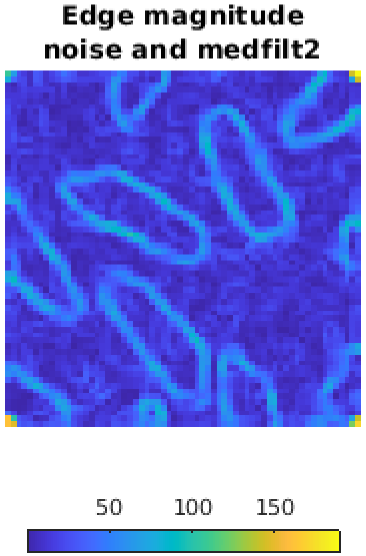
- The orientation of the gradient is reversed between the ‘manual’ and built-in methods.

- Every step of image processing is important. Output of one step is the input for the next.
    - Errors can compound and make downstream analysis challenging or impossible.

whole_images.mlx

Noise


Edge magnitude
noise


Edge magnitude
noise and medfilt2

- Noise can introduce artifacts during processing.
- Particularly problematic whenever gradients/derivatives are used.

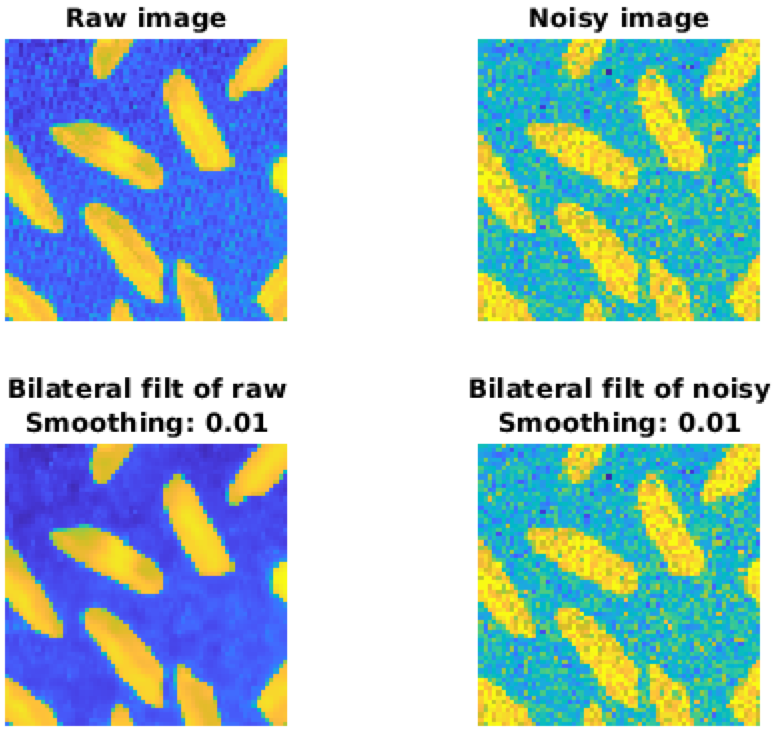
- The choice of filter can affect image and analysis quality.

- Here the noise is reduced in the gradient image, but corner artifacts are introduced by the filtering method.

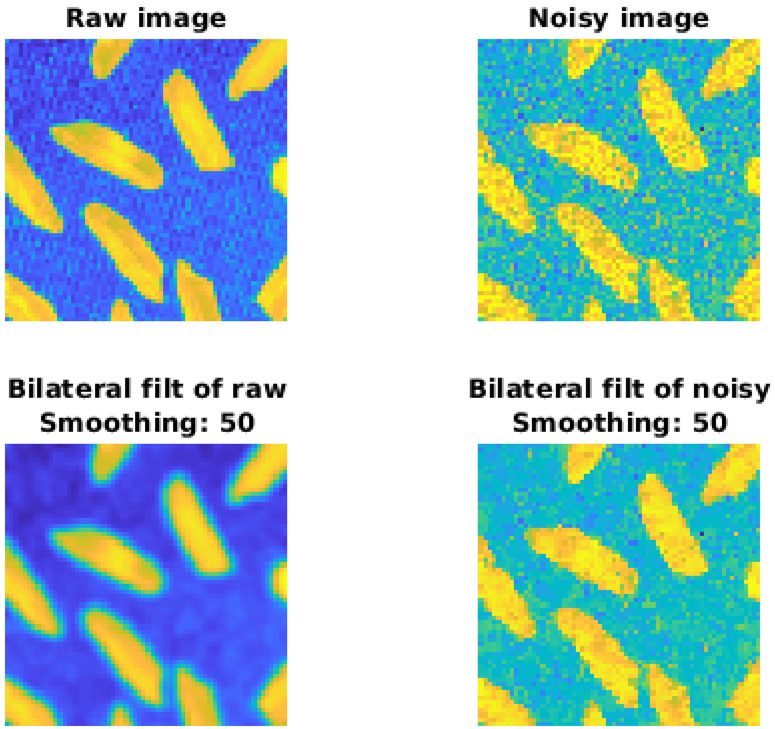
whole_images.mlx

Edge- and noise-aware smoothing

- MATLAB has some more advanced methods which can filter images in ways that suppress noise without degrading edge clarity.
  - Performance will vary based on the data and parameters applied.

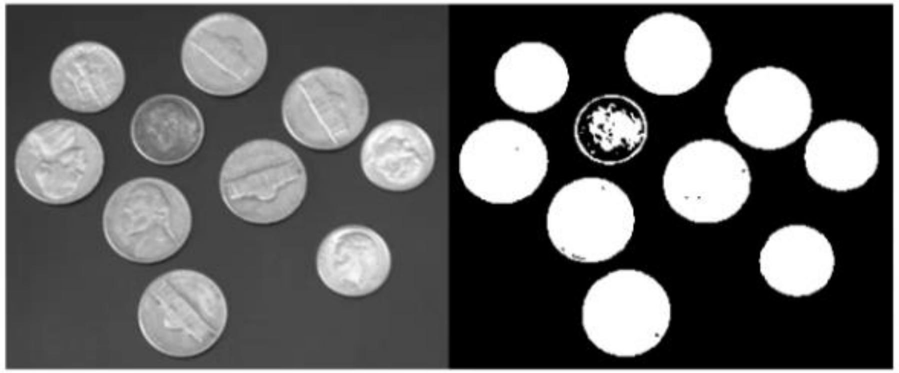


imbilatfilt



Segmentation – Separating an image into parts

- Most basic: Foreground/background
  - Bright or dark background with a dark or bright foreground, respectively.
  - Choose a cutoff value, threshold.
  - Global thresholds can work, but can miss important elements



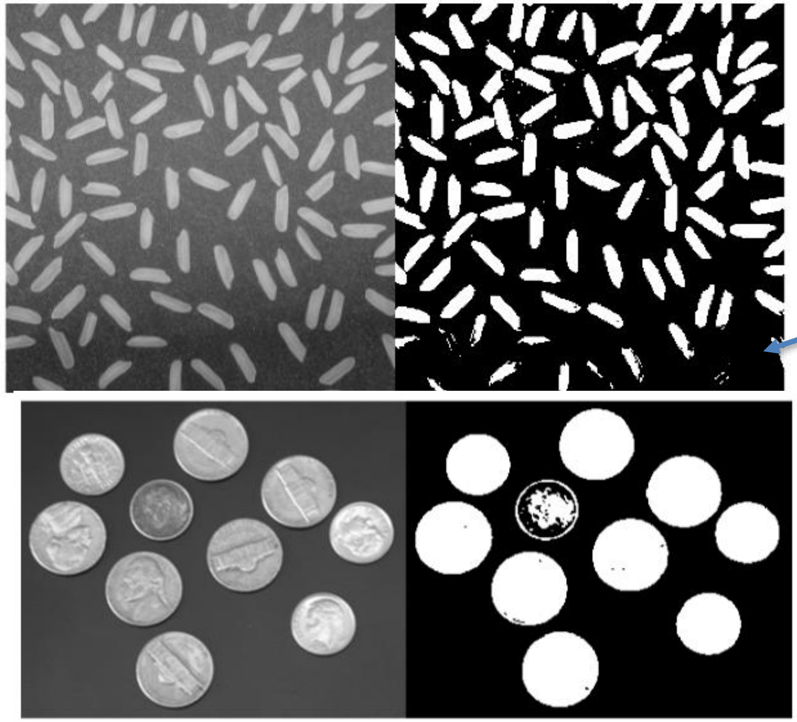
- More complex:
  - Adaptive thresholds
  - Texture clustering
  - Machine learning methods

Many excellent tutorials at: https://www.mathworks.com/help/images/image-analysis.html

segmentation.mlx

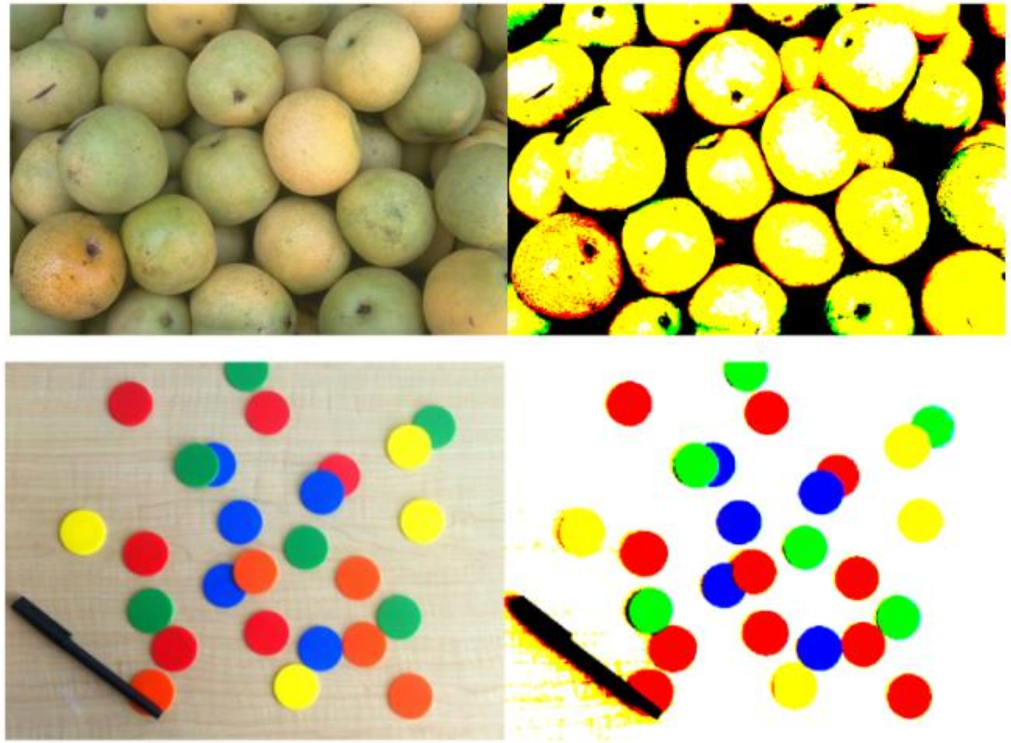
Binary thresholding - Otsu

Otsu thresholding statistically chooses the “best” global separation threshold for foreground/background



Note missing grains

Applied to multi-channel images, each channel is thresholded separately

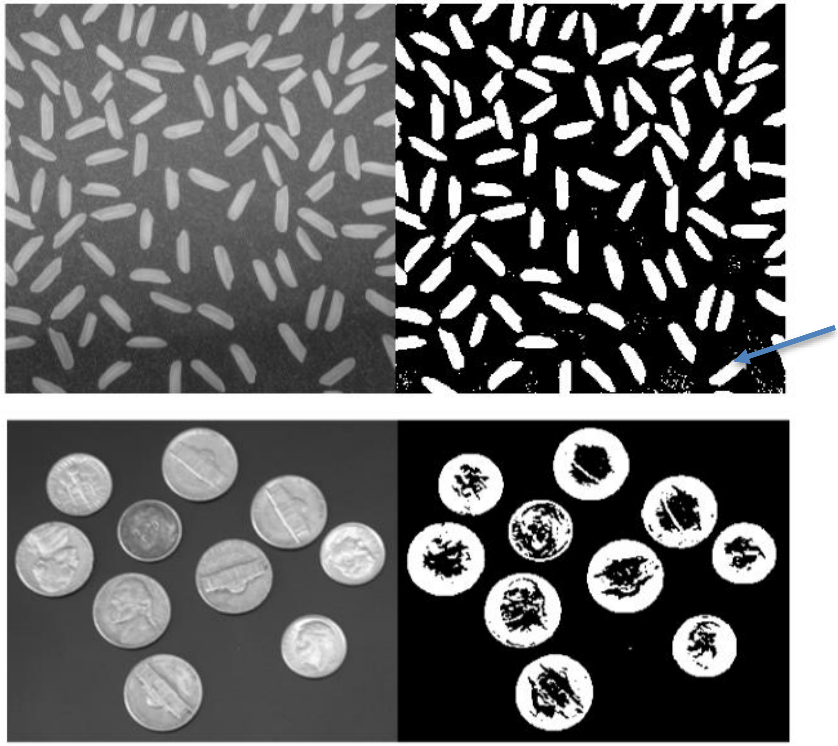


https://www.mathworks.com/help/images/ref/imbinarize.html

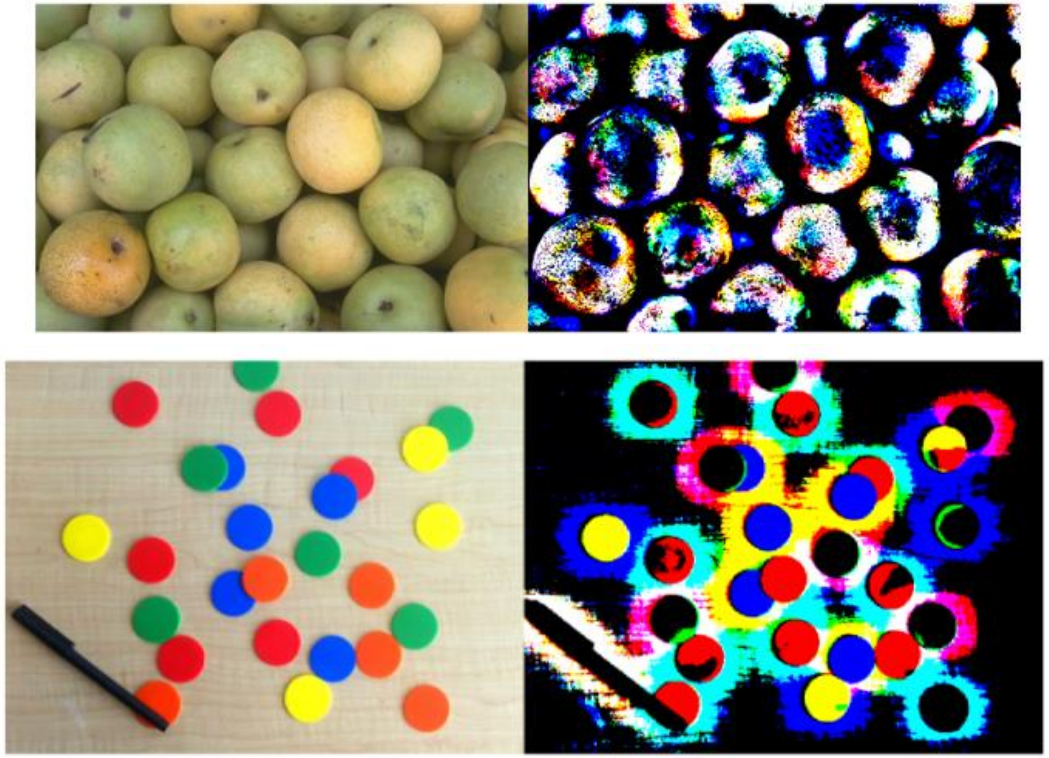
segmentation.mlx

Binary thresholding - Adaptive

Adaptive thresholds are more powerful, but behave in more complex ways and can lead to unpredictable results.



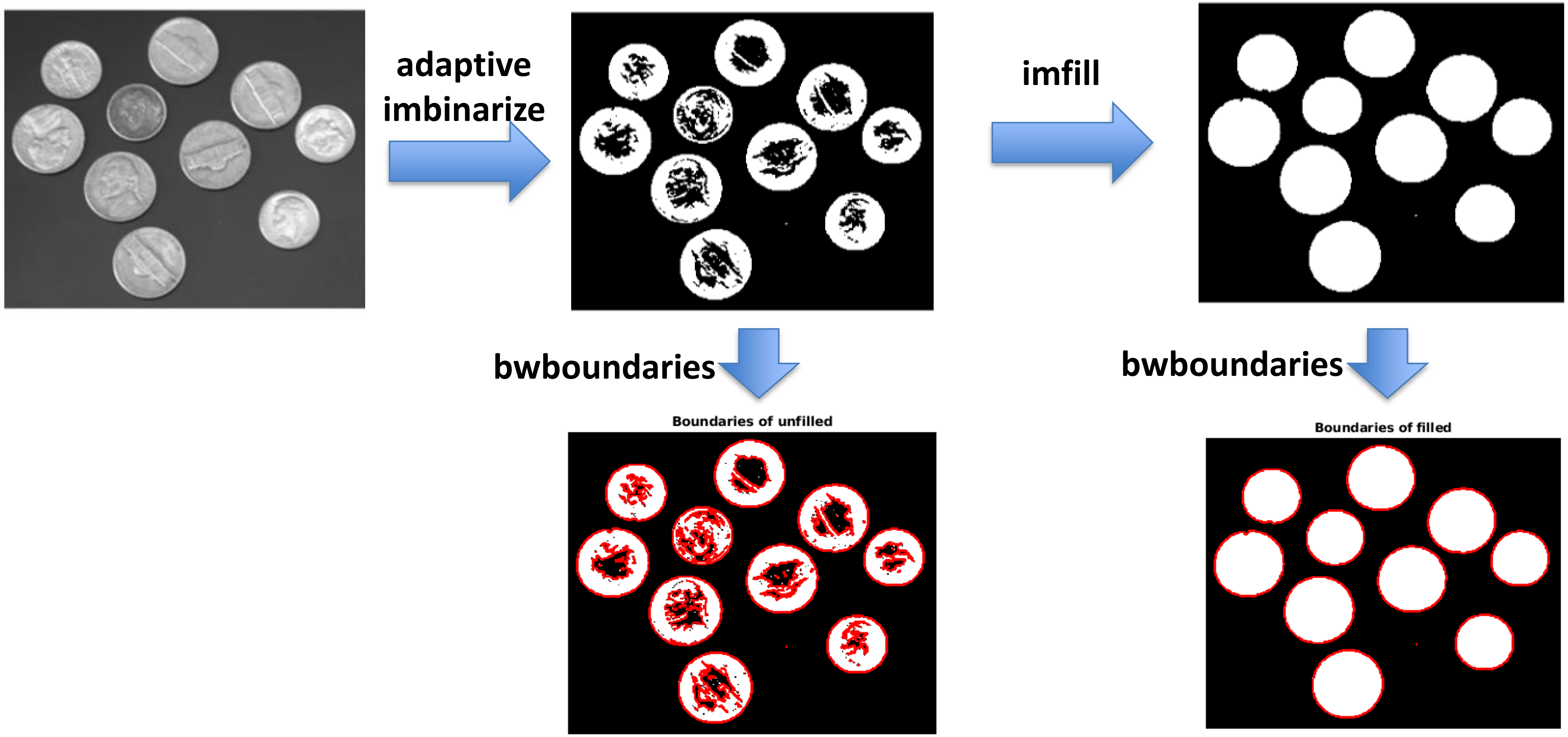
No missing grains



imbinarize(I,’adaptive’)

segmentation.mlx

Filling in regions and finding boundaries



adaptive imbinarize

imfill

bwboundaries

bwboundaries

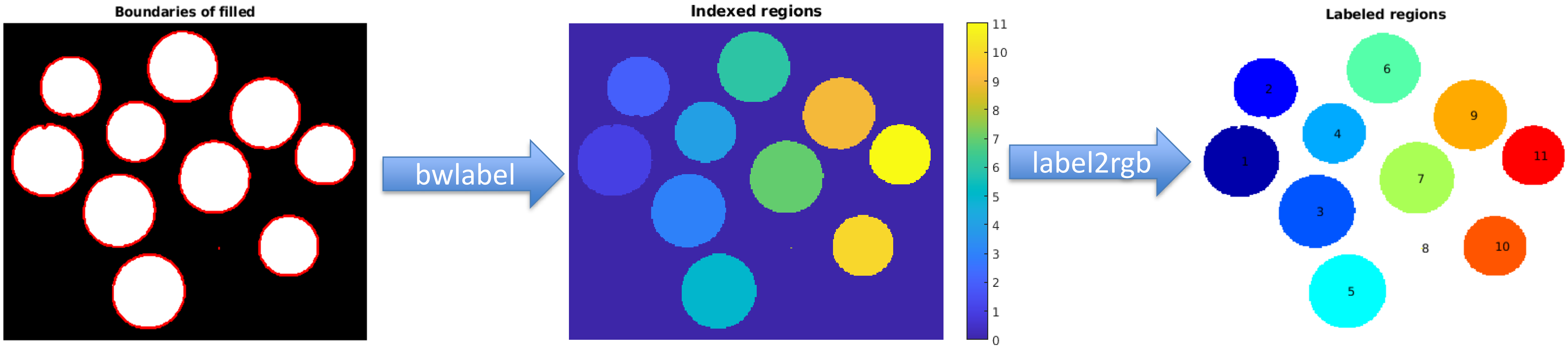
Boundaries of unfilled

Boundaries of filled

segmentation.mlx

Getting labeled regions based on filled-in regions

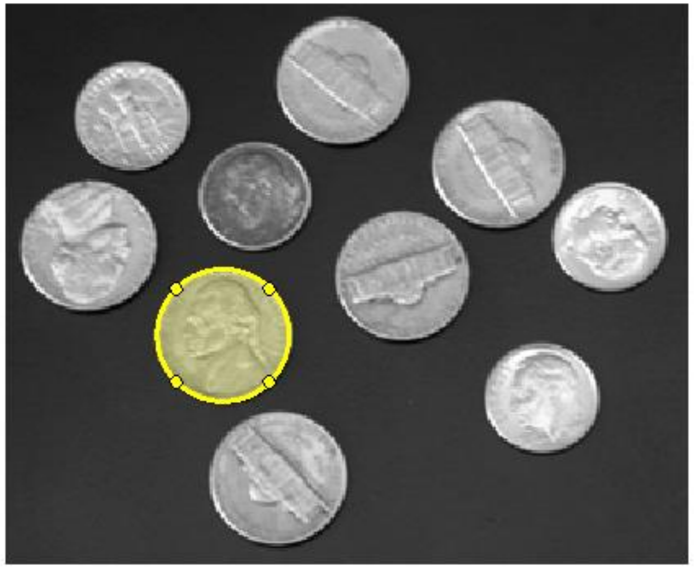
Once you have a visually segmented image, you can get indices and labels for each disconnected region.



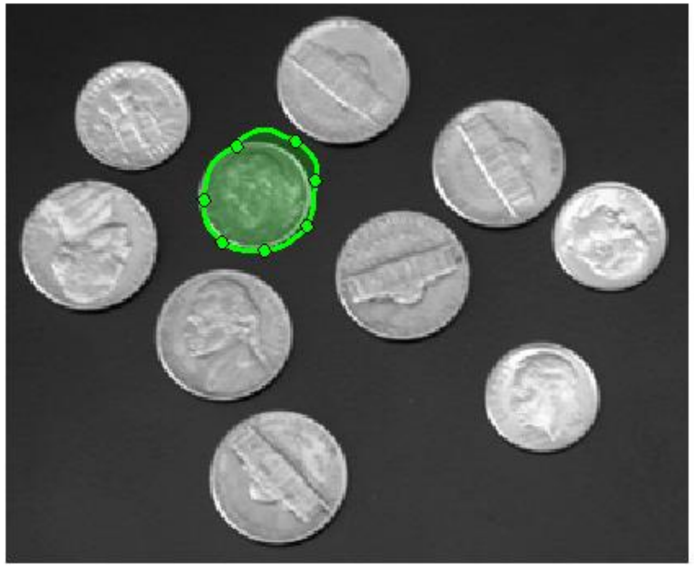
regions_of_interest.mlx

Regions of Interest – Drawing programmatically, or by hand, or with help

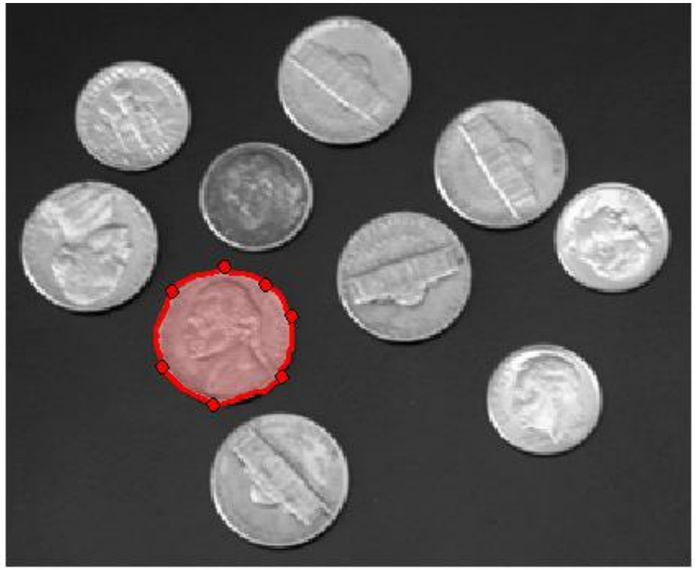
drawcircle()



drawfreehand ()



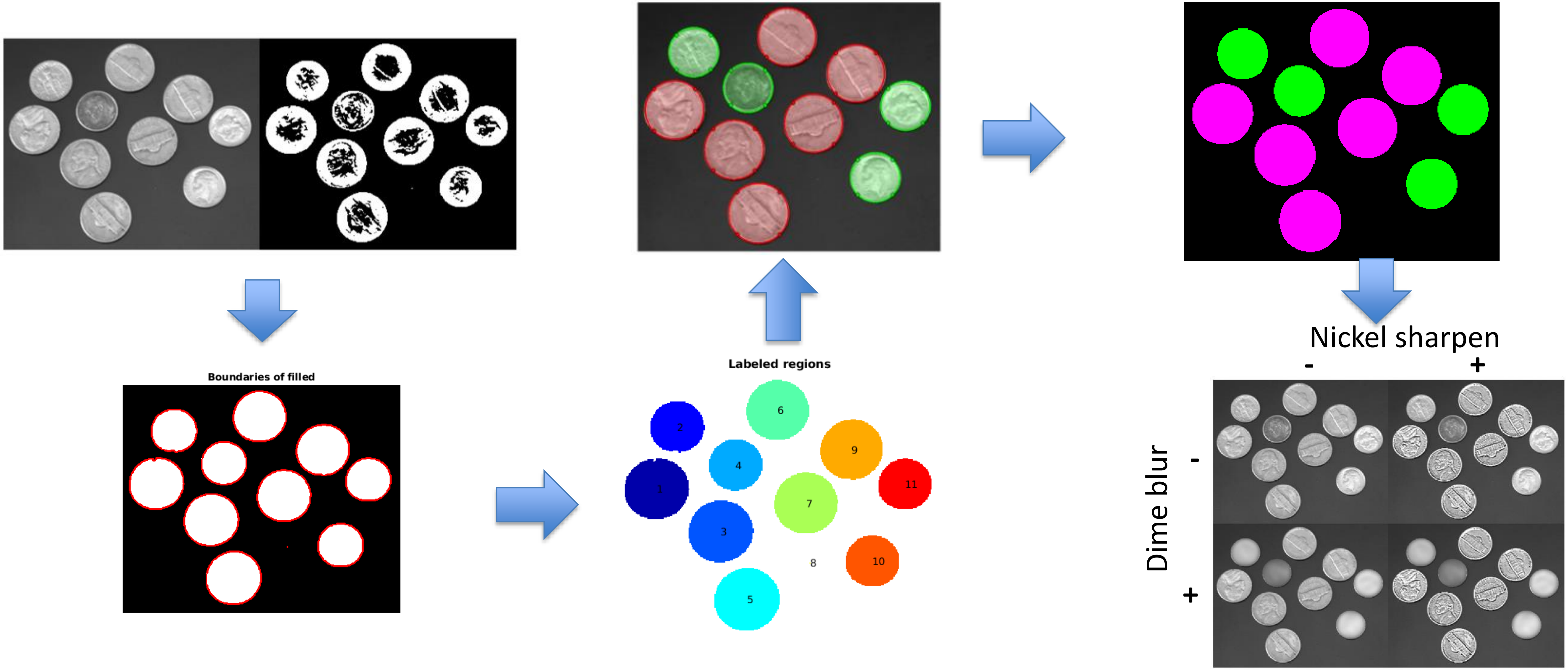
drawassisted ()



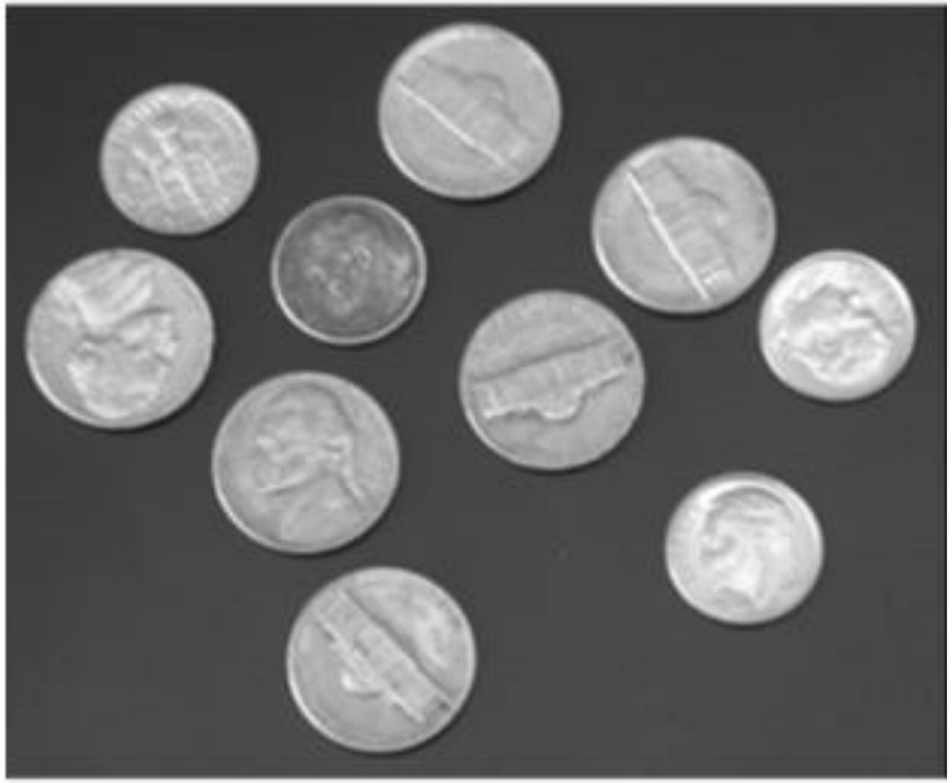
https://www.mathworks.com/help/images/roi-based-processing.html

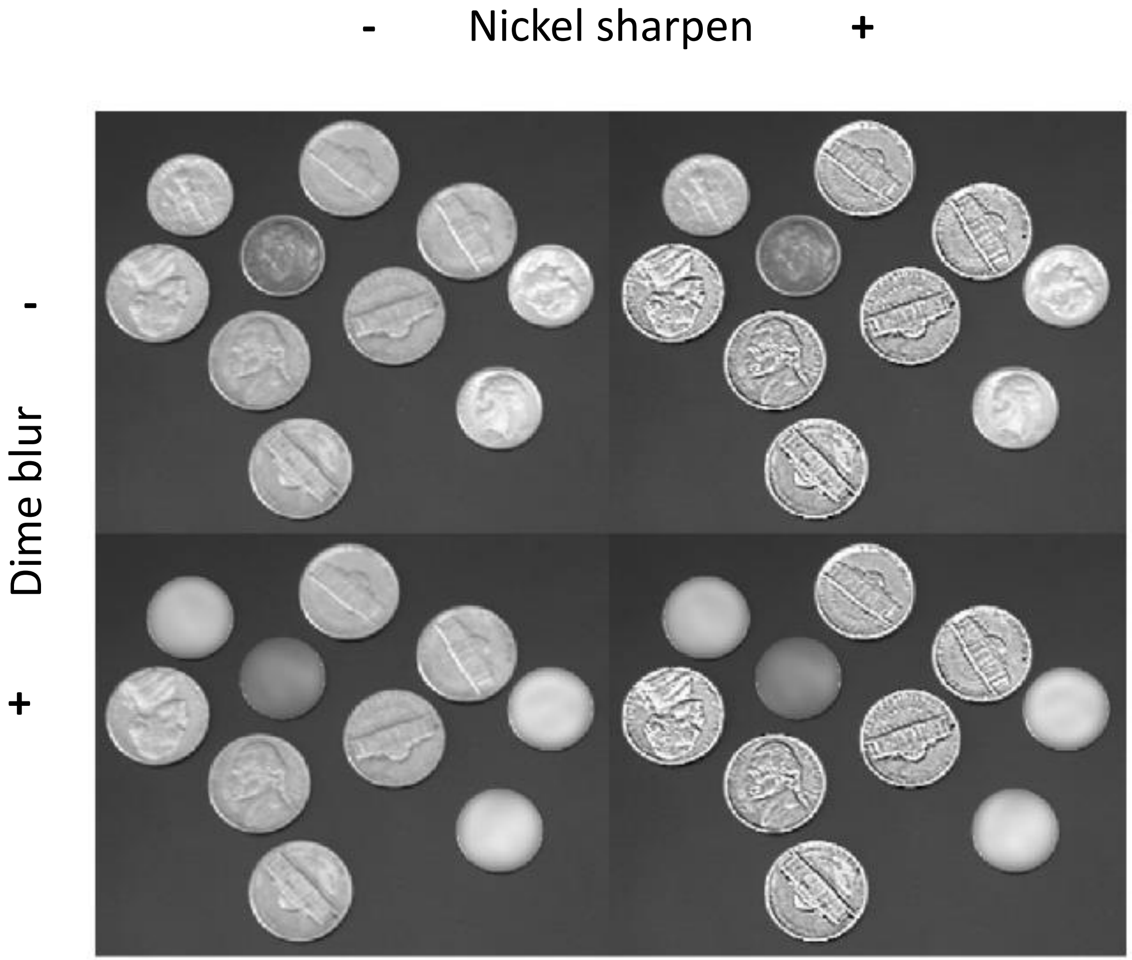
regions_of_interest.mlx

Case Study – Selectively filtering ROIs



Before and After





Deconvolution - Image deblurring

- If blurring/filtering is the forward operation, then deblurring/deconvolution is the inverse.
  - Corrupted by a *blur kernel* or *point spread function* (PSF) plus noise.

- Ill-posed inverse problem
  - Sensitive to noise
  - Sensitive to assumptions

- The more information you have about the image and how it was acquired, the better.
  - Knowing approximate point spread function
  - Knowing noise variance/power
  - Knowing noise statistics

Deconvolution - MATLAB

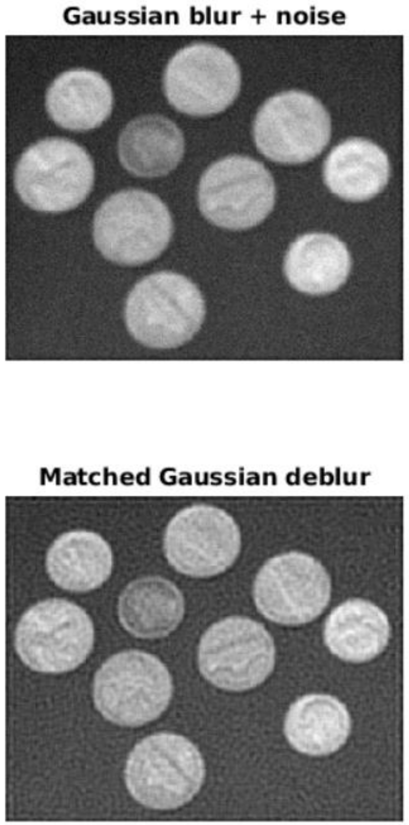
- MATLAB has a built-in **deconv** function. This is NOT for image deconvolution.
  - Properly used for division of polynomials.
  - https://www.mathworks.com/help/matlab/ref/deconv.html

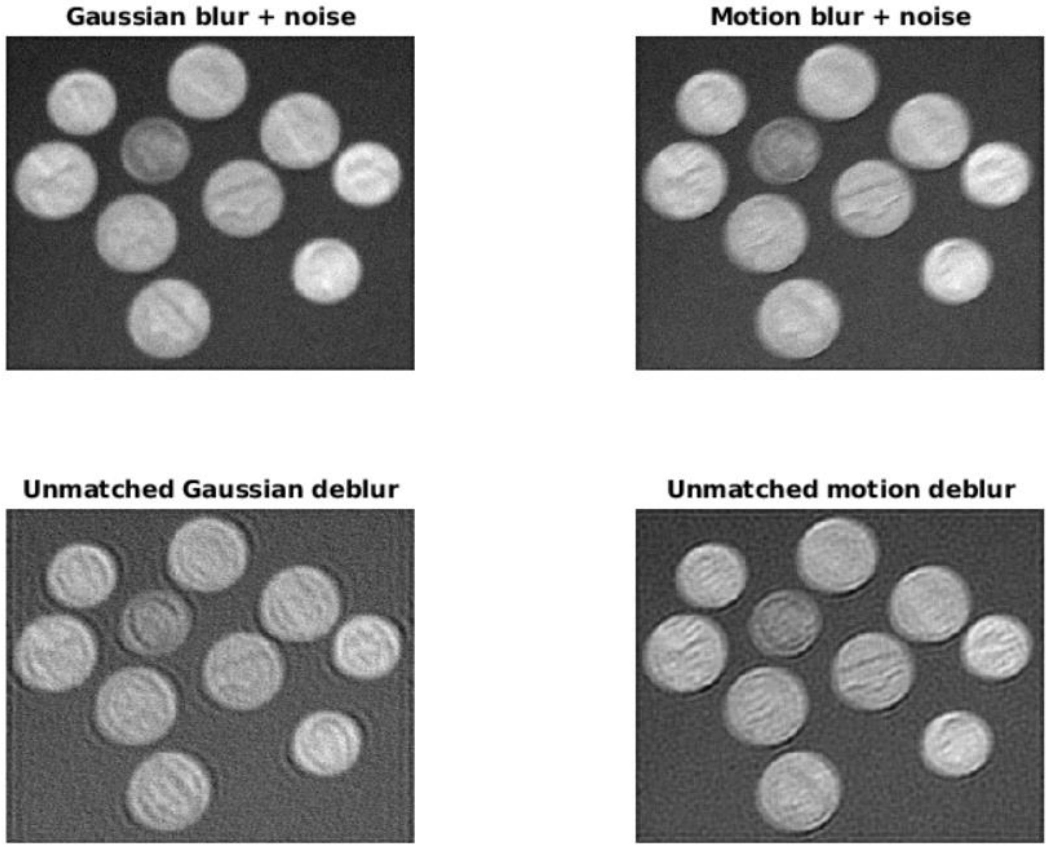
- MATLAB calls image-processing deconvolution ‘deblurring’.
  - https://www.mathworks.com/help/images/image-restoration-deblurring.html

- Good tutorial using histology images:
  - https://www.mathworks.com/help/images/deblurring-images-using-a-regularized-filter.html

deconvolution.mlx

Deconvolution - deconvwnr

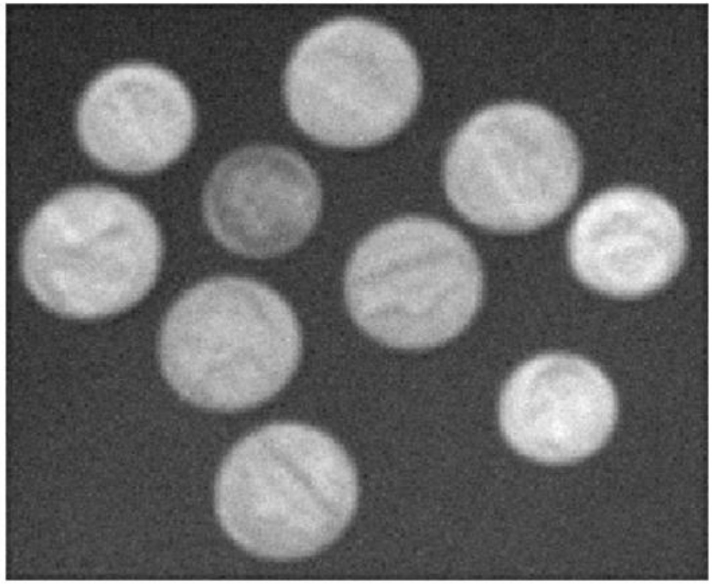




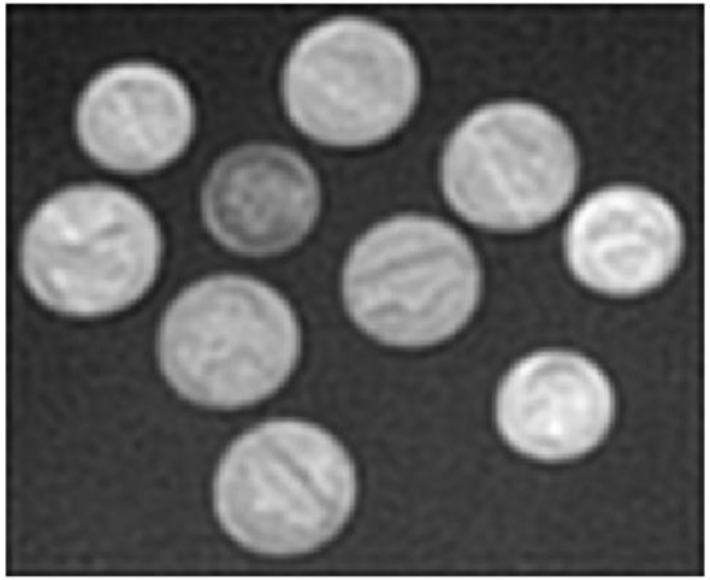
deconvreg – regularized deconvolution

It is always best to know appropriate values of filter parameters if at all possible.

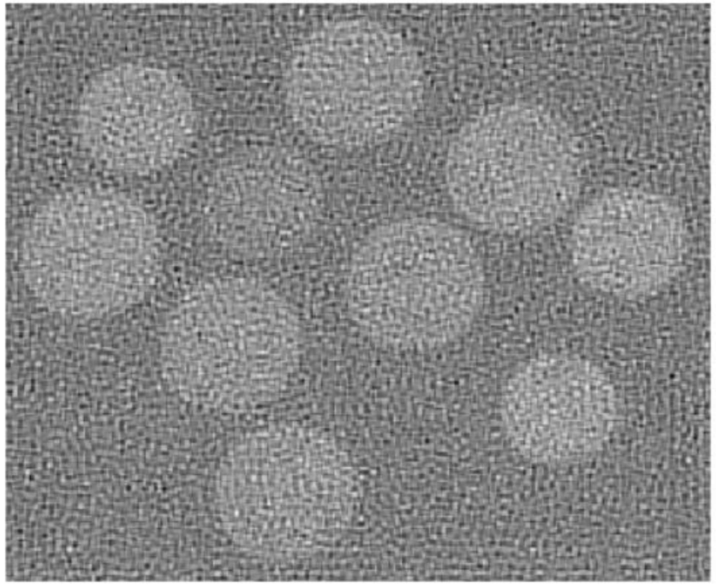
Gaussian + noise



deconvreg w/ est



deconvreg no est



Summary

- Complex processes can be broken down into individual steps
  - Start with manual experimentation / trying things out
  - Automate once you’ve determined a good workflow

- There are many different ways to approach the same image processing problem
  - If a tool already does what you want, read the documentation and use it!
  - Use the simplest tool you can that accomplishes your goal task.

- Know your data – the same method might work differently on different data.
  - Data format, values
  - Noise statistics, point spread function

- Separate data processing and data presentation
  - Presentation should **always** be faithful to the underlying data.

If you have any questions or need any guidance/assistance, contact BioHPC @ biohpc-help@utsouthwestern.edu