TSMC Image Processing and Computer Vision Seminar

Jeff Mather
Development Manager, Image Processing Toolbox

Harshita Bhurat
Senior Product Manager, Image Processing & Computer Vision

Seminar Agenda

- What's New in Image Processing and Computer Vision?
- Anomaly and Defect Detection
- Working with Very Large Images
- Image Processing Deployment

IDNEO Develops Embedded Computer Vision and Machine Learning Algorithms for Interpreting Blood Type Results

Challenge

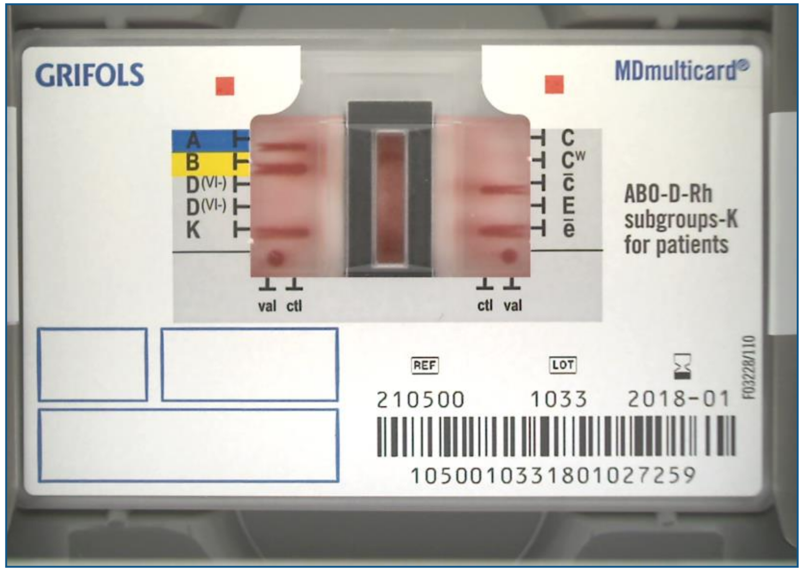
Automate the visual interpretation of cards used by hospital staff to determine patient blood antigenic typing

Solution

Use MATLAB to develop, test, and generate embedded code for image analysis and machine learning algorithms

Results

- Accuracy requirements exceeded
- Project completion time halved
- Code Optimized system delivered



The Grifols MDmulticard.

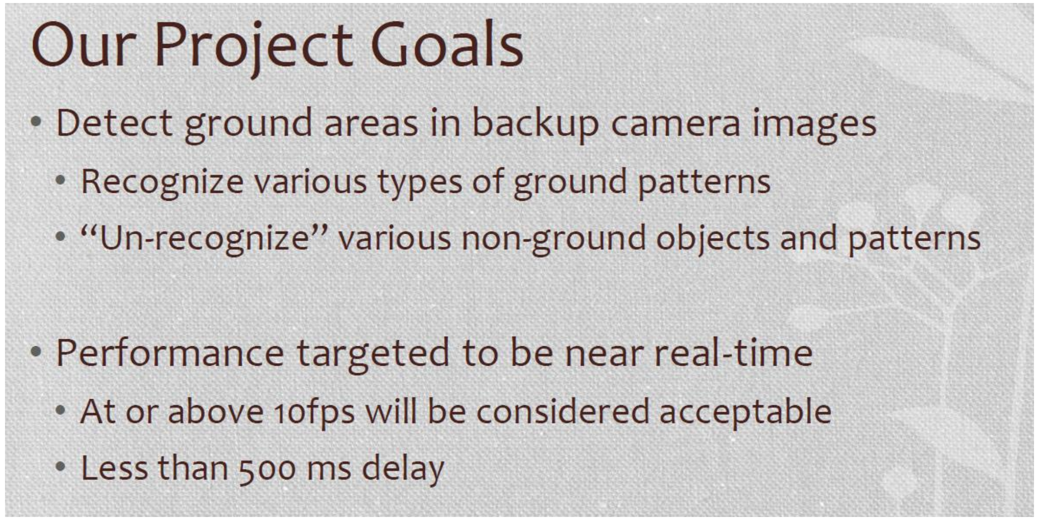
"MATLAB enabled us to rapidly analyze images and improve our algorithms over several iterations. After we had developed the algorithms, MATLAB made it easy to deploy them to an embedded system. It would have been much more difficult to do this in C or another language, particularly on a short timeline."

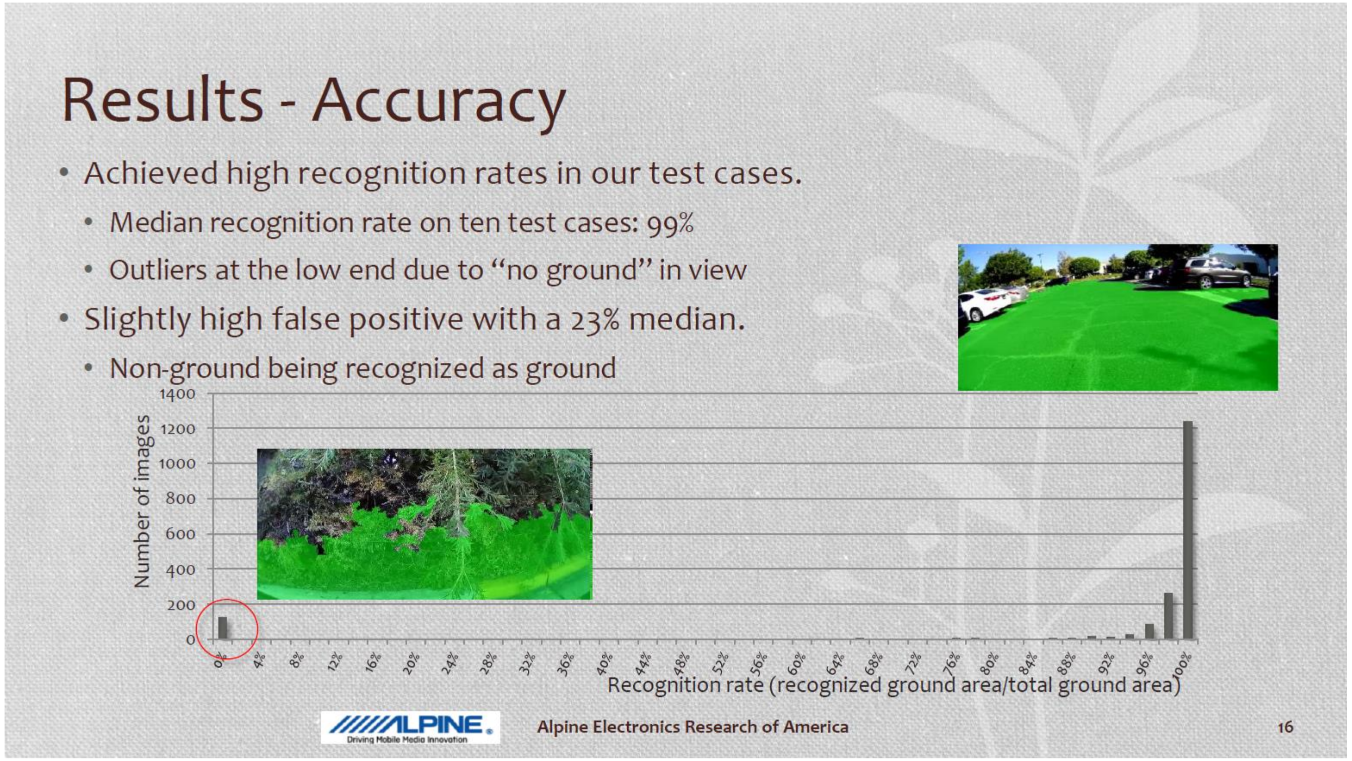
- Marc Blanch, IDNEO

Link to user story

Alpine

Computer vision concept development and code generation



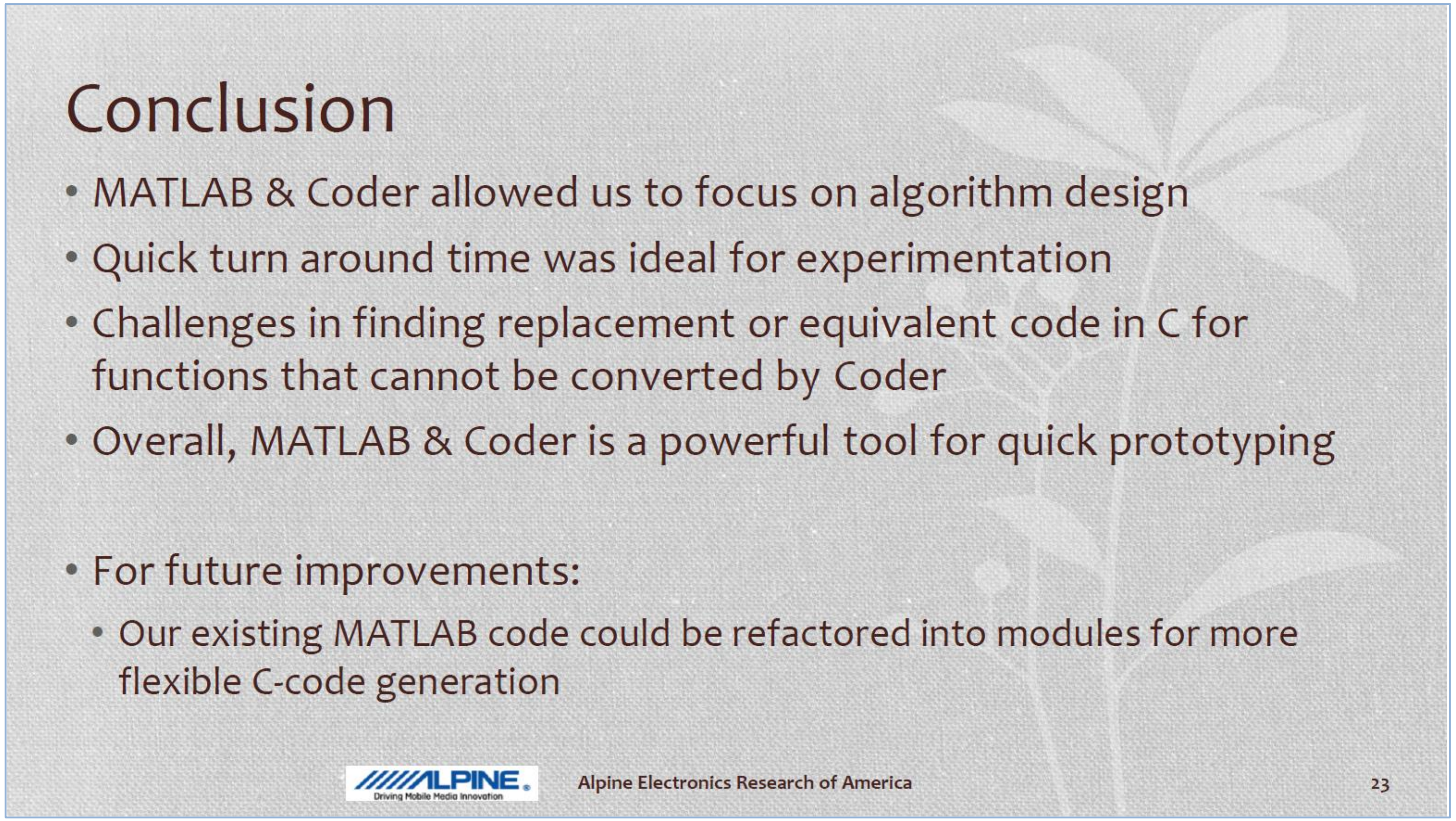


Source: Application of MATLAB and MATLAB Coder for an Automotive Vision Proof of Concept, Maung Han, Alpine Electronics Research

Alpine presentation slides from MathWorks Automotive Conference 2016, Michigan
Alpine presentation recording from MathWorks Automotive Conference 2016, Michigan

Alpine

Computer vision concept development and code generation

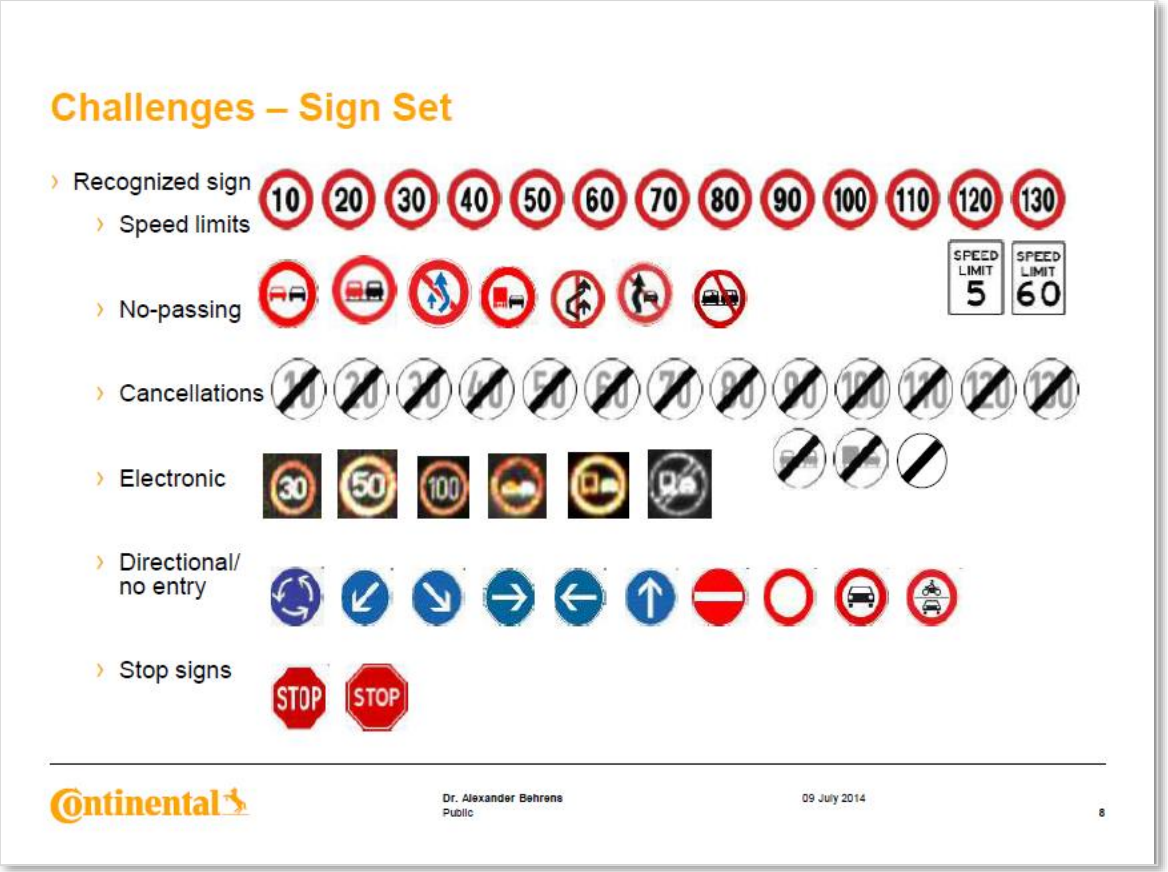


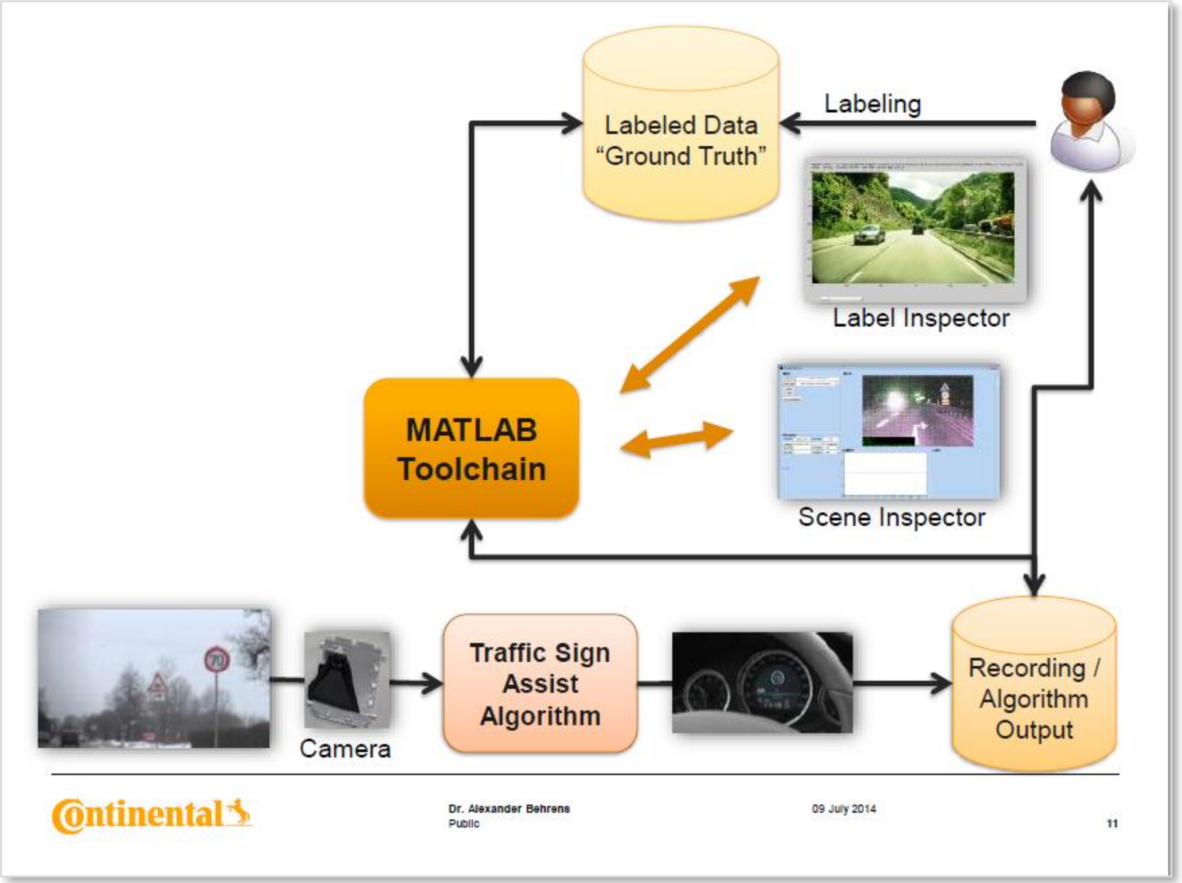
Source: Application of MATLAB and MATLAB Coder for an Automotive Vision Proof of Concept, Maung Han, Alpine Electronics Research

Alpine presentation slides from MathWorks Automotive Conference 2016, Michigan
Alpine presentation recording from MathWorks Automotive Conference 2016, Michigan

Continental

Ground truth labeling and algorithm training for traffic sign recognition





[Continental presentation slides]() from MathWorks Automotive Conference 2015, Stuttgart, Germany
[Continental presentation recording]() from MathWorks Automotive Conference 2015, Stuttgart, Germany

Continental

Ground truth labeling and algorithm training for traffic sign recognition

Summary

- MATLAB is used in daily work for development and evaluation of driver assistance functions
- Prototypes are designed with MATLAB for predevelopment and proof of concept
- Data management, evaluation, and interactive analysis are supported by MATLAB tools and GUIs
- Traffic Sign Recognition and other functions make high use of MATLAB tools
- MATLAB and its established features
  - reduces our tool development efforts,
  - accelerates our simulation cost,
  - and allows reliable, repeatable and accurate parameter optimizations

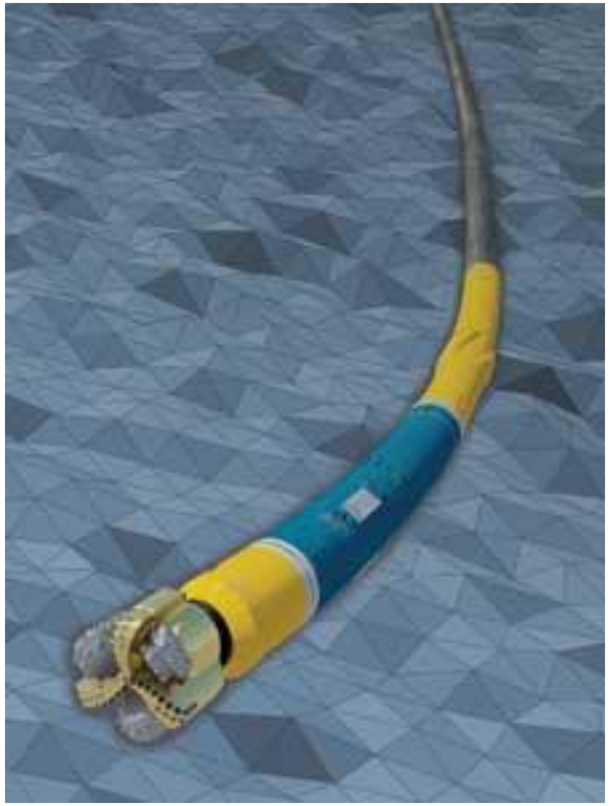
Dr. Alexander Behrens
Public

24 September 2015

18

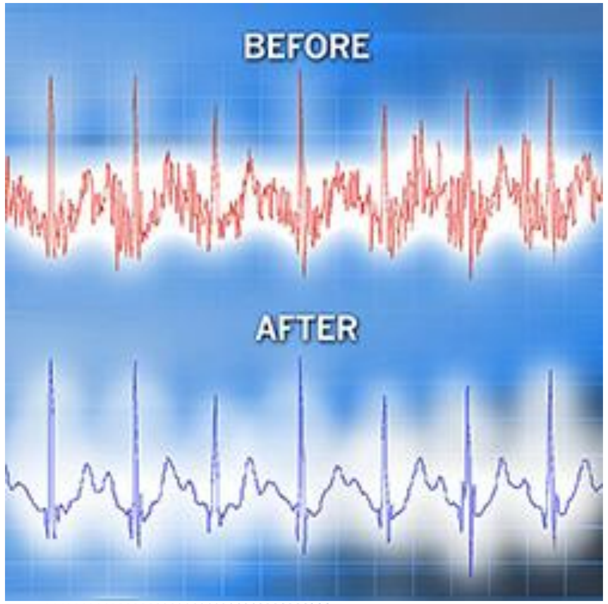
Continental presentation slides from MathWorks Automotive Conference 2015, Stuttgart, Germany
Continental presentation recording from MathWorks Automotive Conference 2015, Stuttgart, Germany

Examples of MATLAB Coder Usage



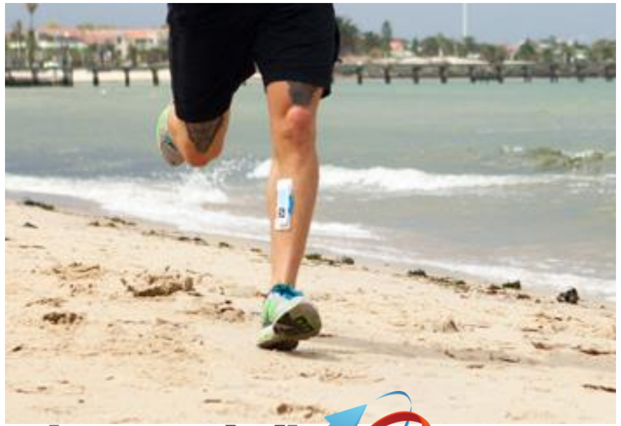
BAKER HUGHES

DELPHI



VivaQuant™
Quantifying vital signs. Accurately.

RESPIRI

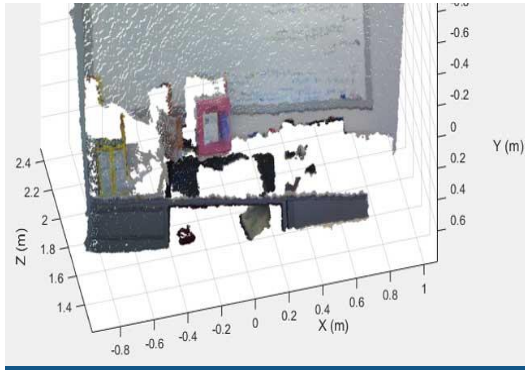


dorsaVi
Inspiring the world to move

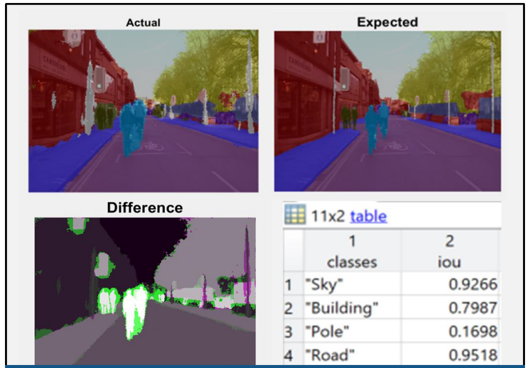
What’s New in Image Processing and Computer Vision?

What’s New in Image Processing and Computer Vision?

Apps



Algorithms

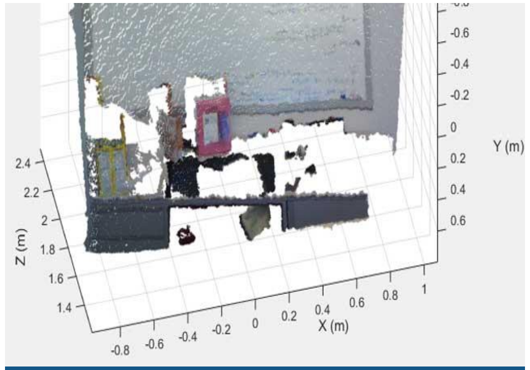


Deep Learning

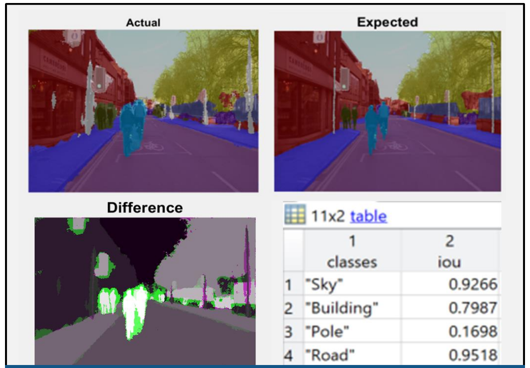
Access

What’s New in Image Processing and Computer Vision?

Apps



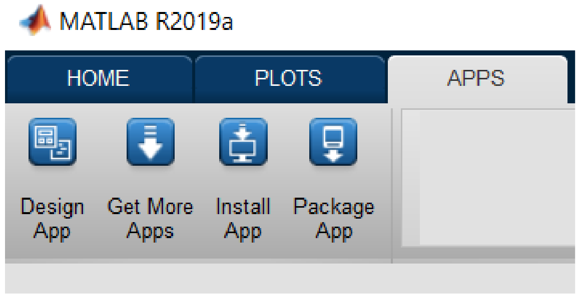
Algorithms

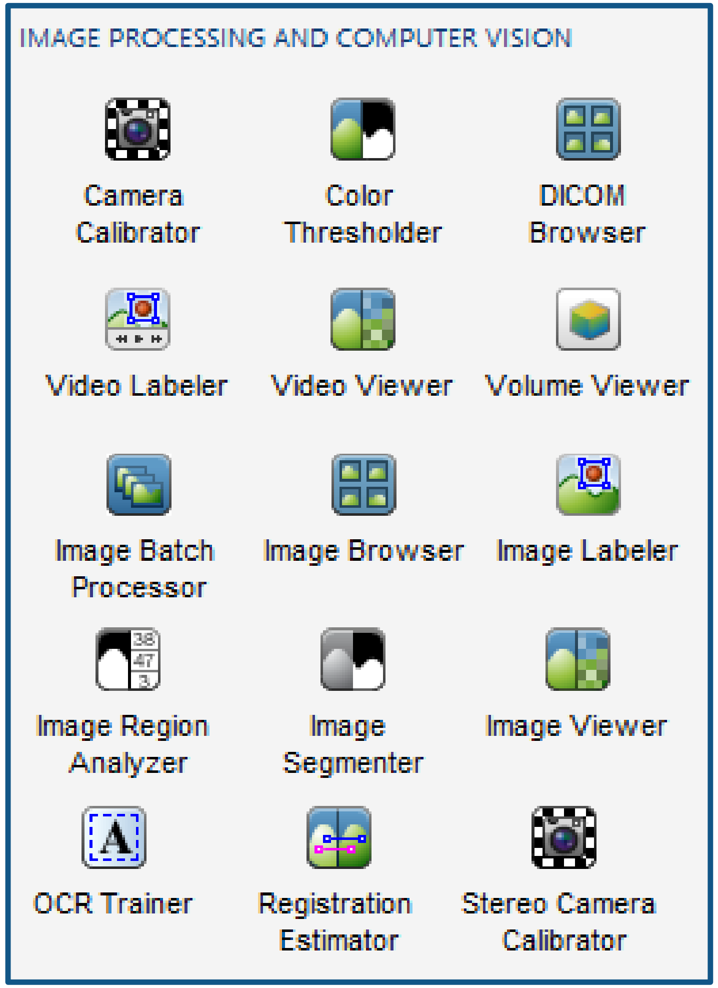


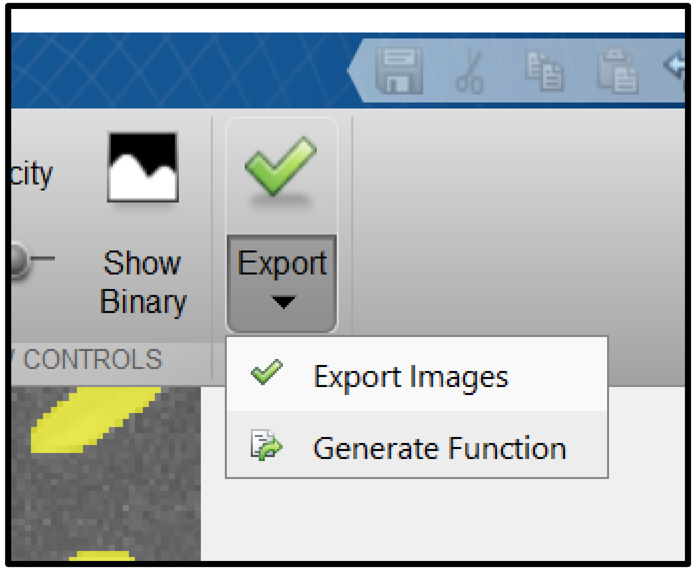
Deep Learning

Access

Apps Accelerate Workflow

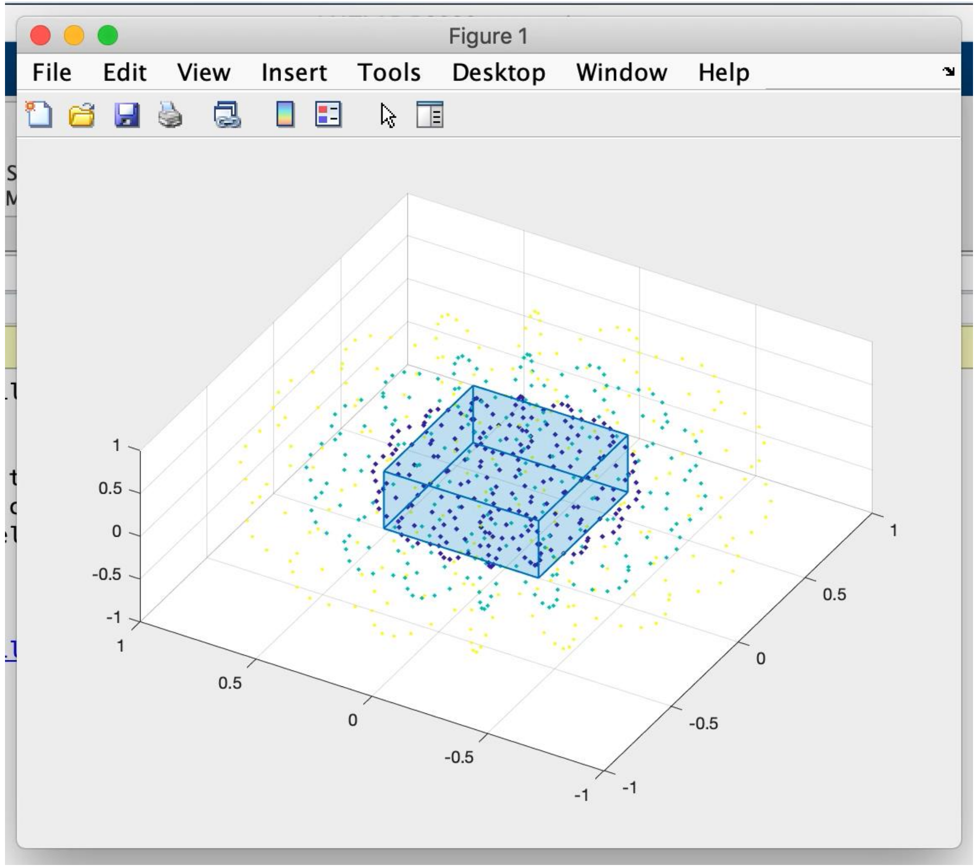




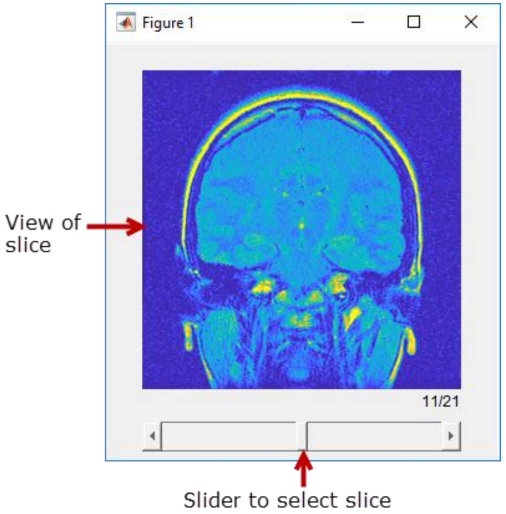


Apps – Just Released

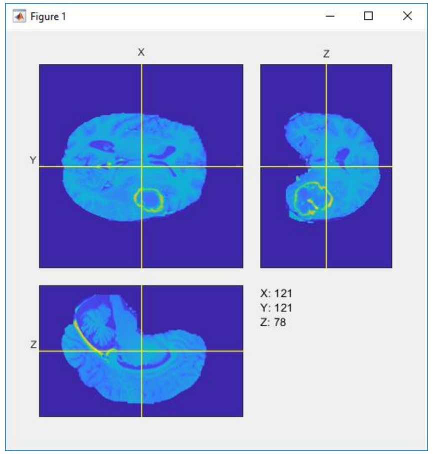
R2019b



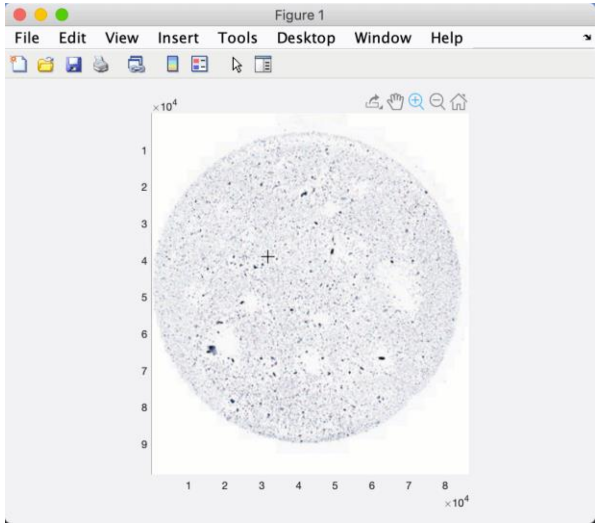
2D & 3D ROIs



SliceViewer



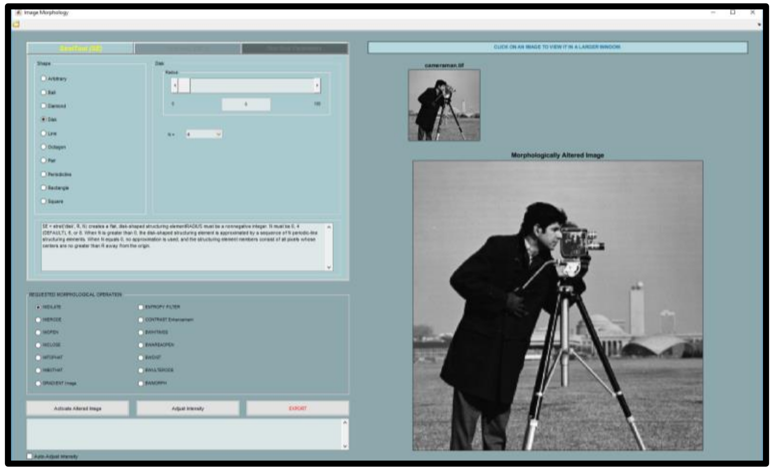
OrthosliceViewer

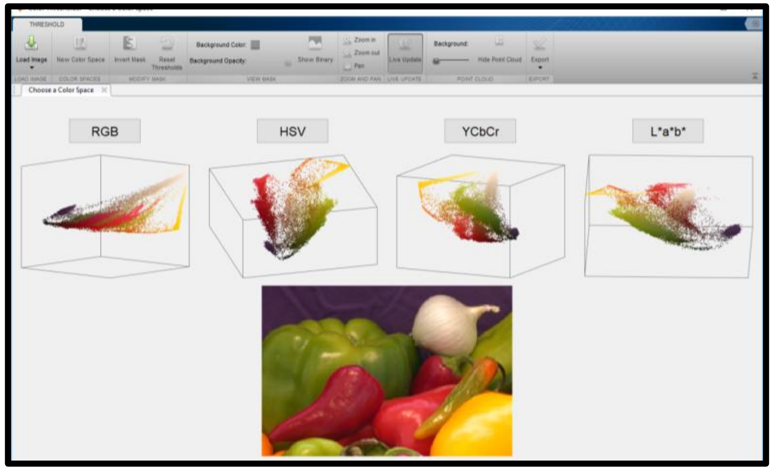


bigimageshow

Apps for Morphology and Segmentation

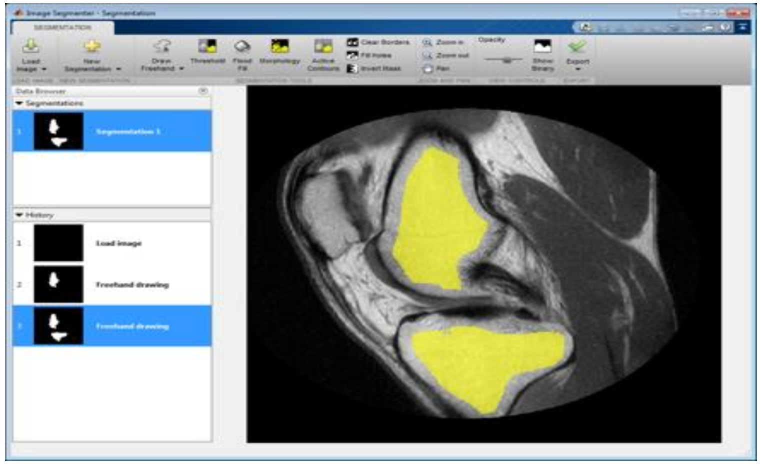
Image Morphology
(file exchange)





Color Thresholder
R2014a

Image Segmenter
R2014b



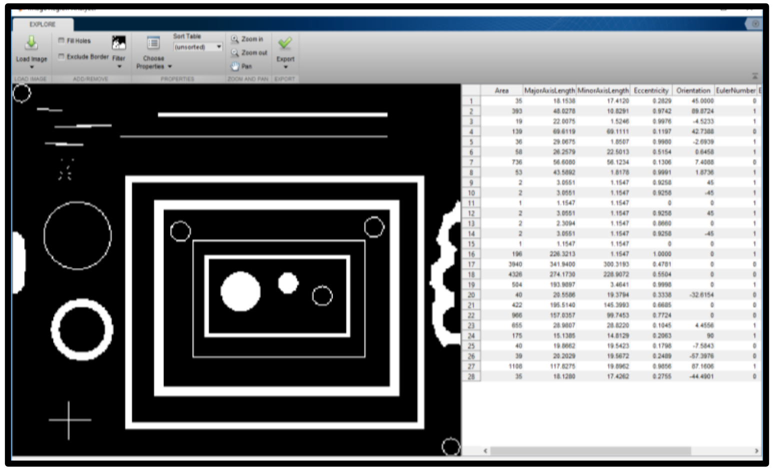


Image Region Analyzer
R2014b

[Example: Segment lungs form 3-D Chest Scan]()

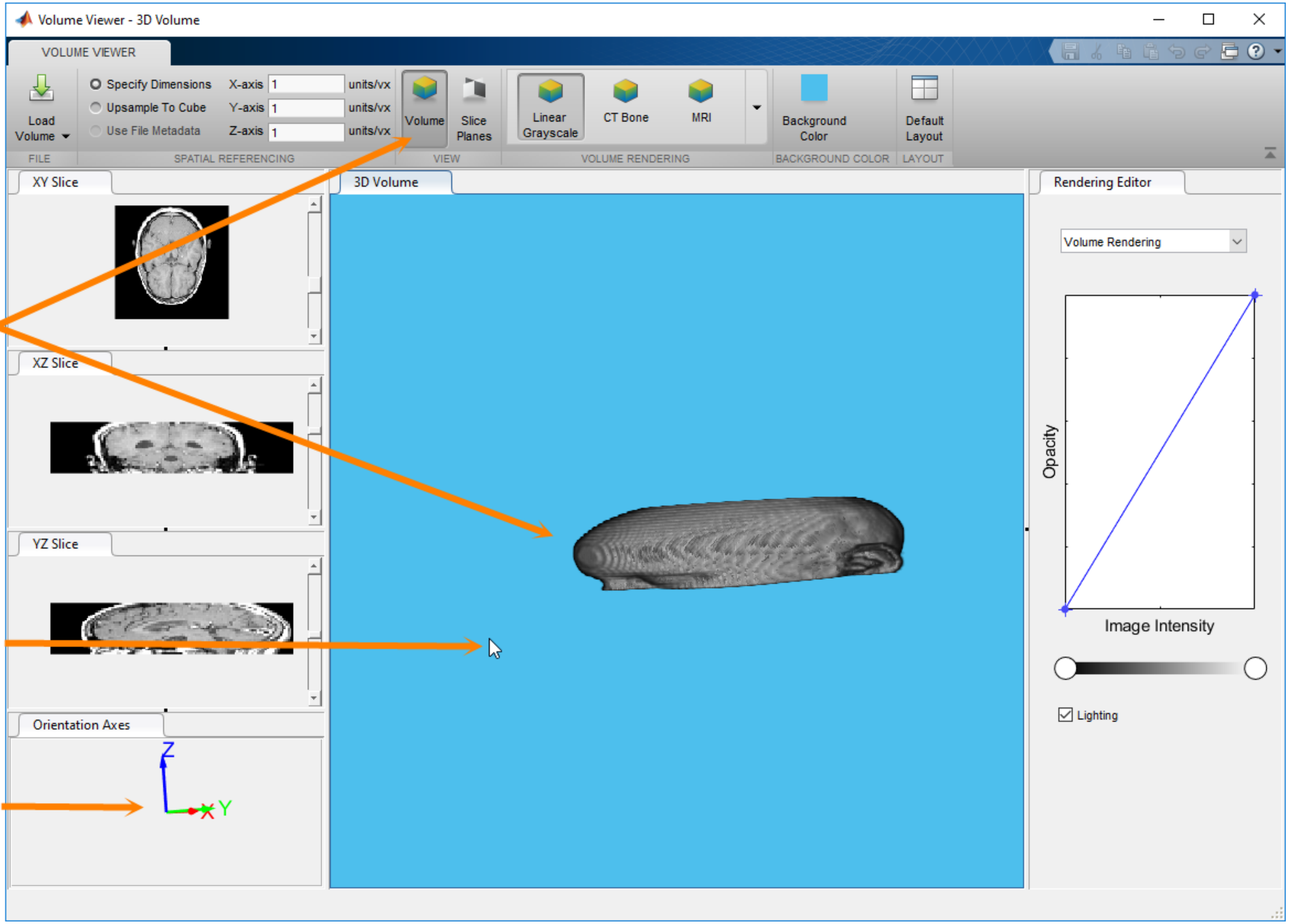
Volume Viewer

R2017a

Volume Viewer displays data as a volume, by default.

Move the cursor in the window to rotate the volume.

Axes move to reflect orientation of volume.

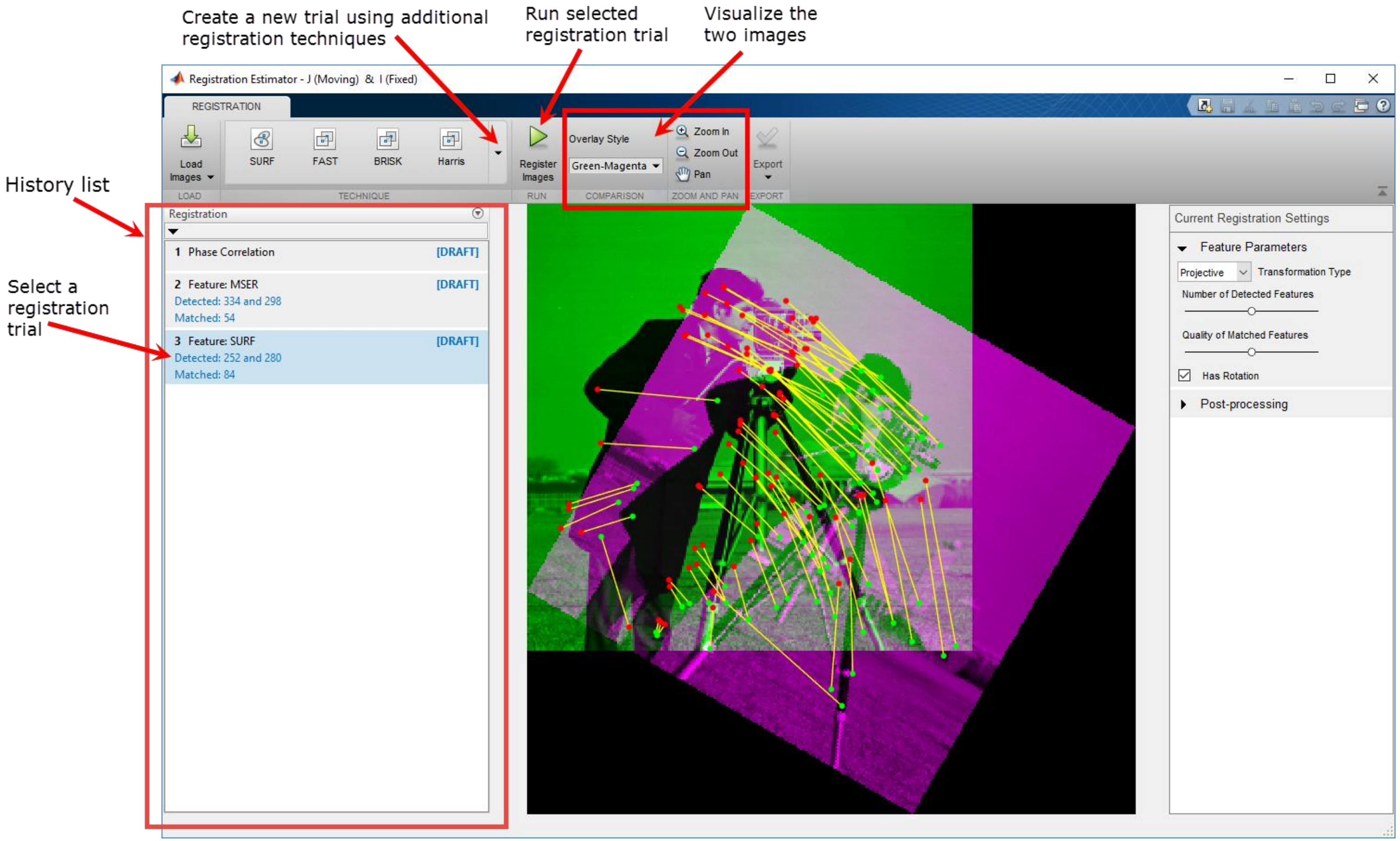


View 3-D volumetric data as a volume or as a set of slices

[Example: Explore 3-D Labeled Volumetric Data with Volume Viewer App]

Registration Estimator App

R2017a

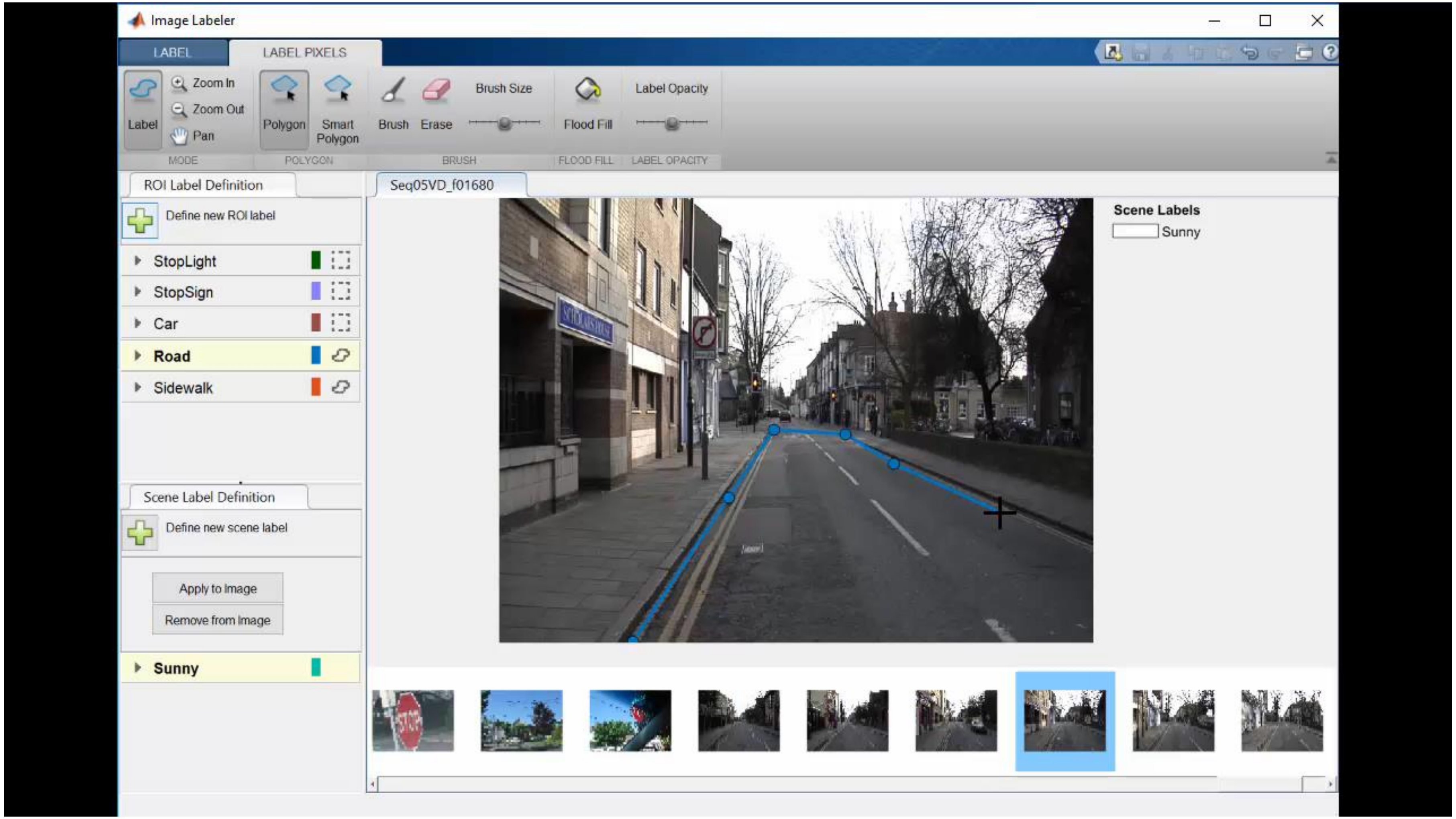


Explore different registration techniques to bring two images into alignment

[Example: Register Images Using the Registration Estimator App]

Video and Image Labeler

R2018a
R2018b

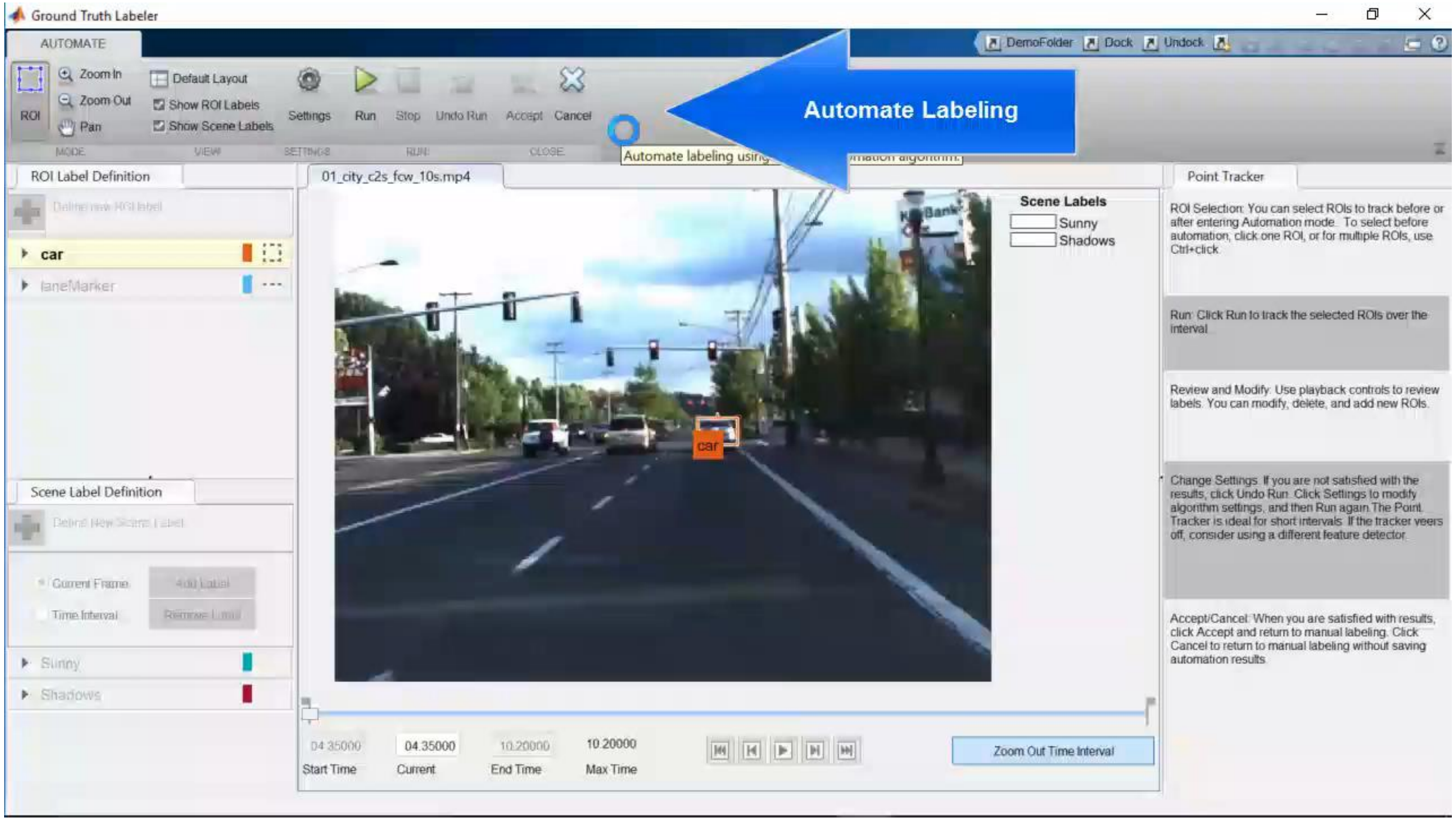


Label images and video using bounding boxes / at the pixel level

Example: Get Started with the Video Labeler

Automate Ground Truth Labeling

R2017a

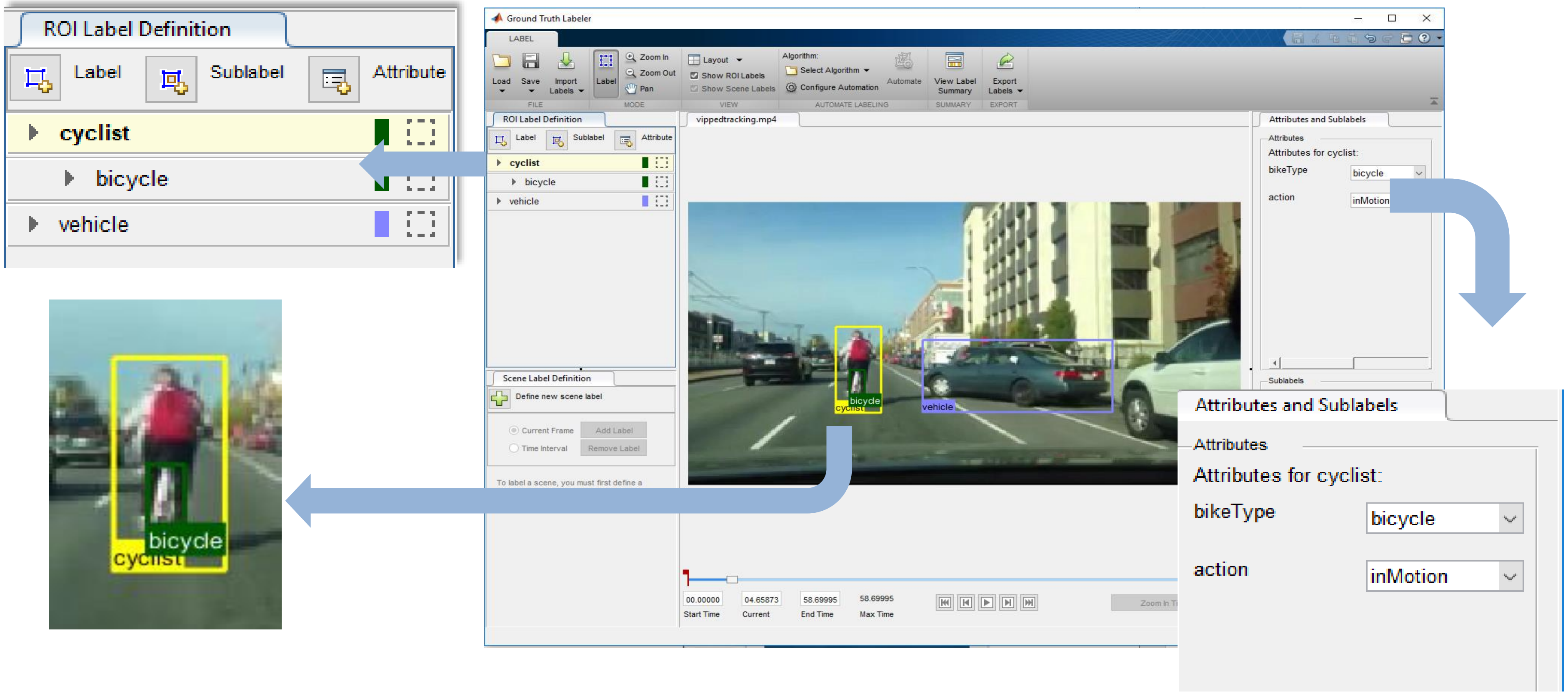


Run custom or pre-built detectors to identify objects in automated-driving applications

[Example: Create Automation Algorithm for Labeling]

R2018a

Attributes and Sublabels



Camera Calibrator App

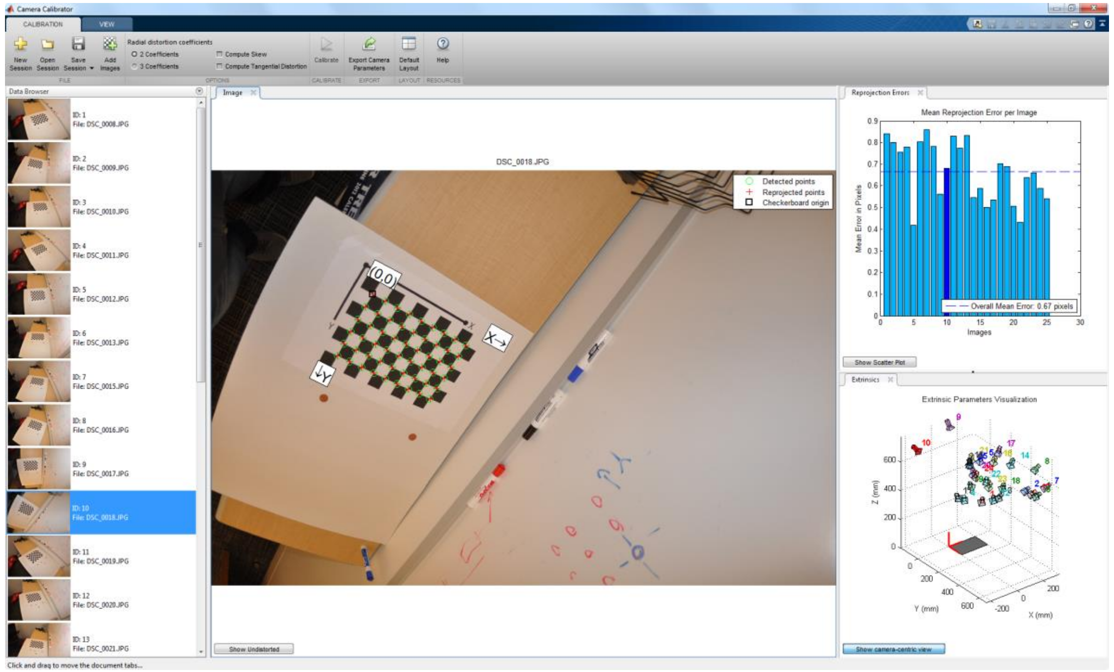
R2013b

- App for simplified workflow for estimating camera intrinsic and extrinsic parameters

- Extracts real world measurements from images and video

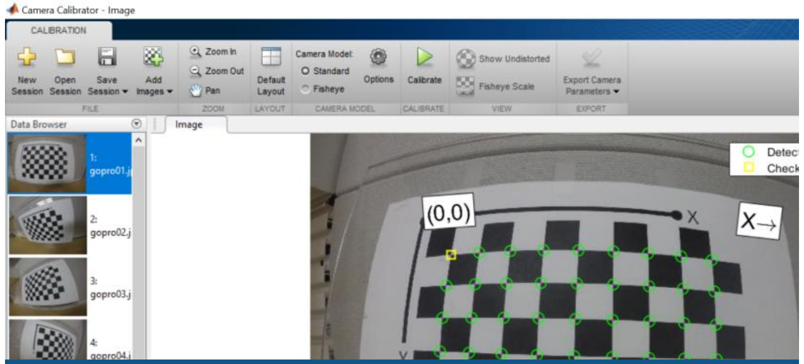
- Automatically detects checkerboard patterns

- Removes the effects of lens distortion from an image



R2018a

Camera Calibrator App – Fisheye Calibration



Fisheye Support in Camera Calibrator

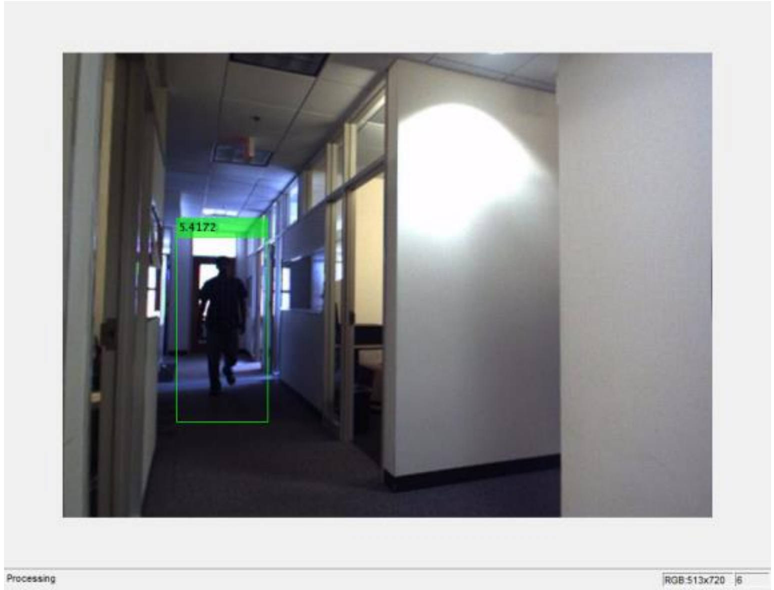
Undistort Fisheye Images

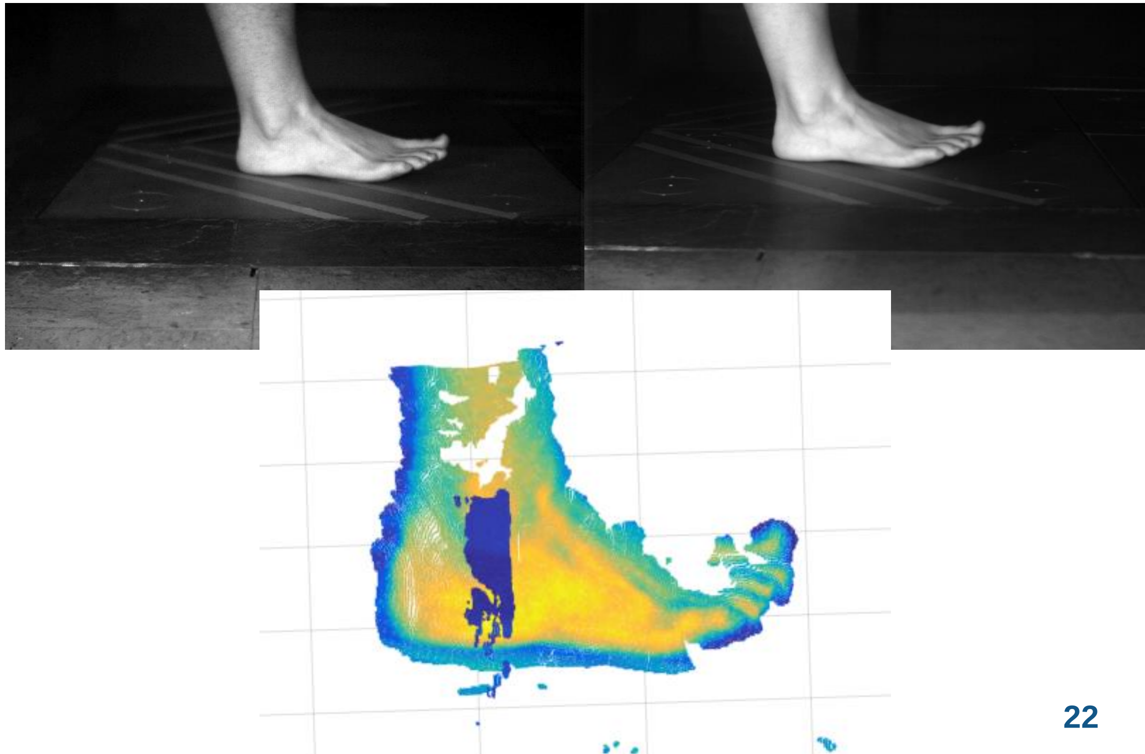
Interactively calibrate fisheye lenses using the Camera Calibrator app

Stereo Camera Calibration

R2014a

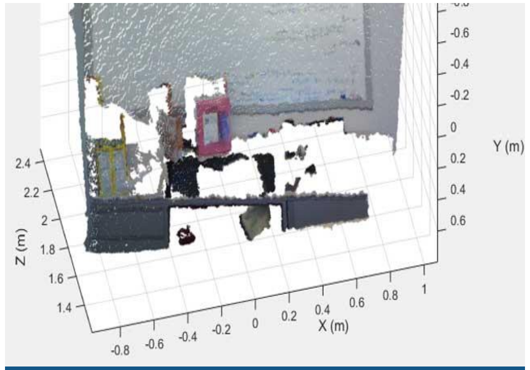
- Stereo calibration
- Semi-global disparity matching
  – Better results than block matching
- Estimate real distance to objects
- 3D scene reconstruction from disparity



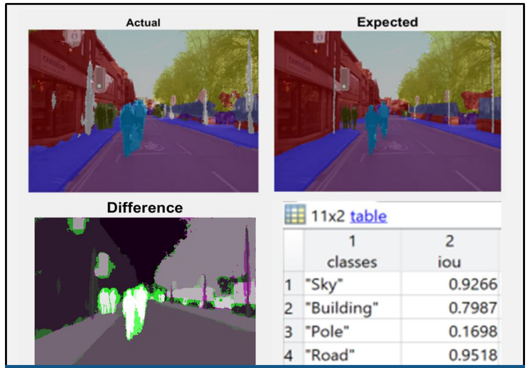


What’s New in Image Processing and Computer Vision?

Apps



Algorithms

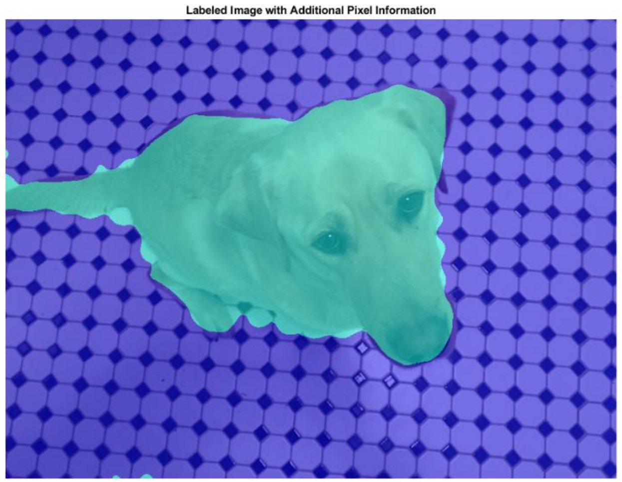


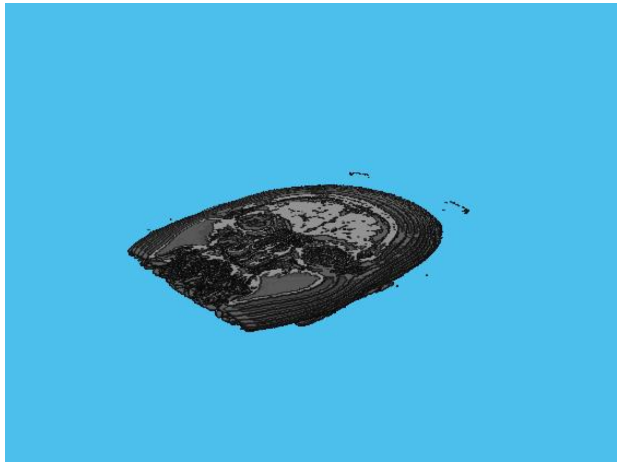
Deep Learning

Access

Image Segmentation

R2018b





imsegkmeans

imsegkmeans3

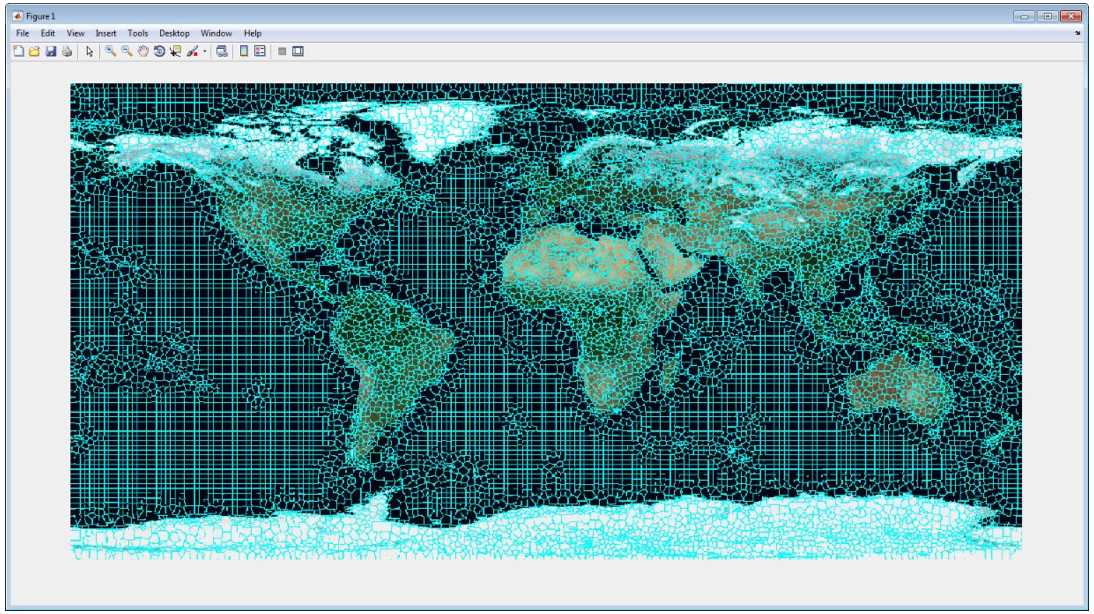
fibermetric

Segment 2-D images and N-D volumes using k-means clustering

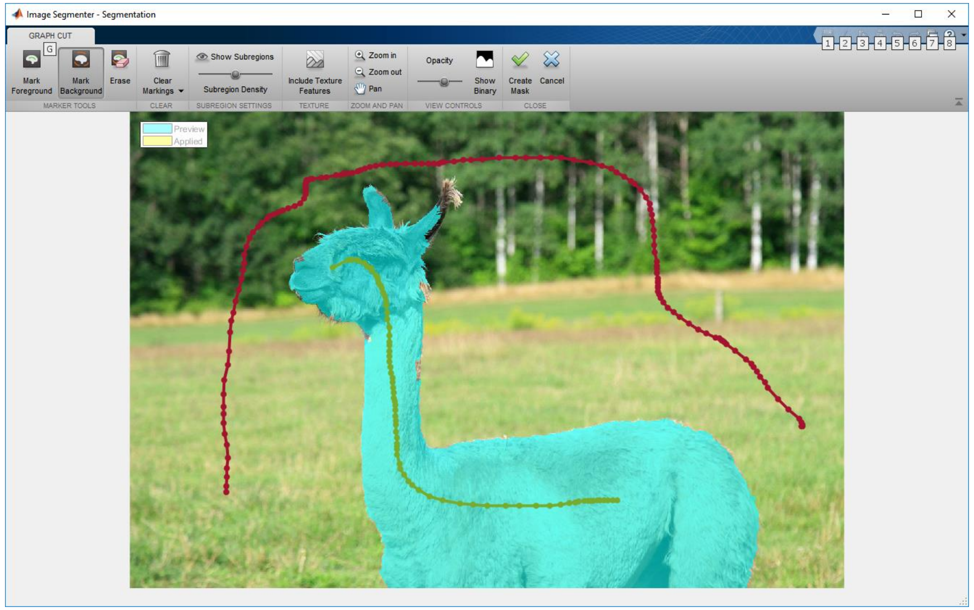
[Example: Color-Based Segmentation Using K-Means Clustering]

Superpixels

R2016a
R2016b



superpixels

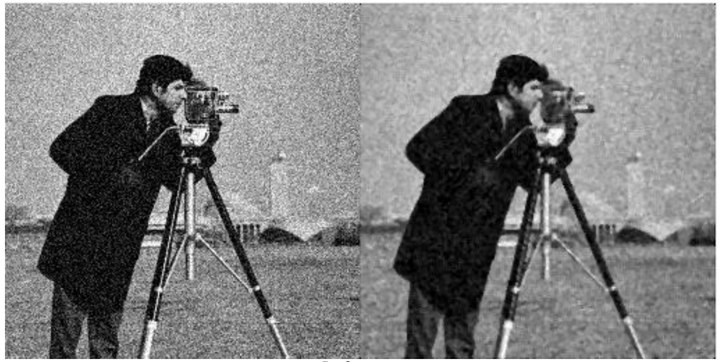


lazysnapping

- Group 2-D or 3-D image pixels into superpixels with similar values.
- Reduce complexity of image processing algorithms such as segmentation
- `lazysnapping`: graph cut segmentation

[Example: Plot Land Classification with Color Features and Superpixels]

Filtering

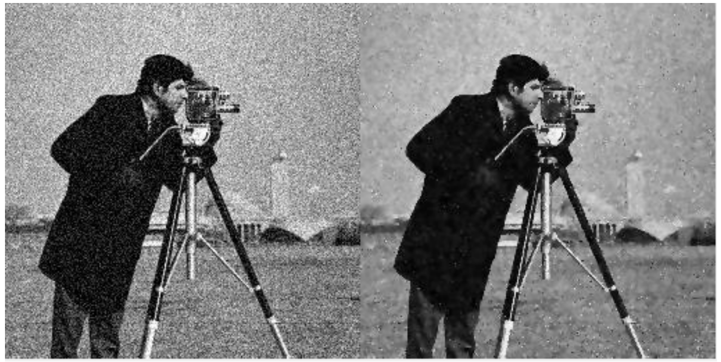


Median

Local Laplacian

Bilateral

Guided

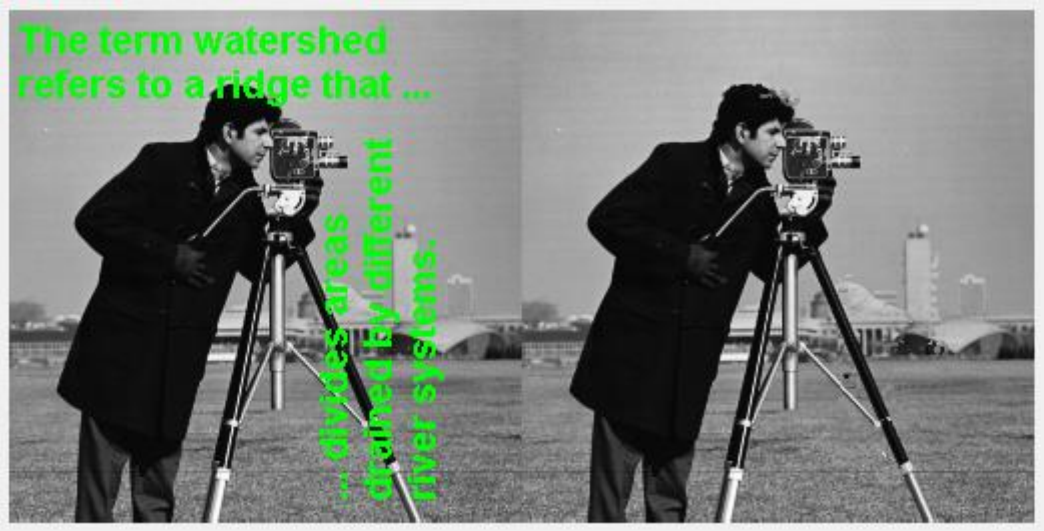


Diffusion

Non-Local Means

Image inpainting

- Fill damaged or missing regions of images
- inpaintCoherent for smaller defects
- inpaintExemplar for larger regions



Autogenerated via MATLAB Report Generator®

3-D Image Processing Functions

Image Import and Conversion

- dicomread
- dicomreadVolume
- imbinarize
- niftiinfo
- niftiwrite
- niftiread

Image Arithmetic

- imabsdiff
- imadd
- imdivide
- immultiply
- imsubtract

Image Enhancement

- histeq
- imadjustn
- imboxfilt3
- imfilter
- imgaussfilt3
- imhistmatchn
- integralBoxFilter3
- integralImage3
- medfilt3

Morphology

- bwareaopen
- bwconncomp
- imbothat
- imclose
- imdilate
- imerode
- imopen
- imreconstruct
- imregionalmax
- imregionalmin
- imtophat
- watershed

Image Analysis

- activecontour
- bfscore
- bwselect3
- dice
- edge3
- gradientweight
- graydiffweight
- imgradient3
- imgradientxyz
- imhist
- imsegfmm
- jaccard
- regionprops
- superpixels3

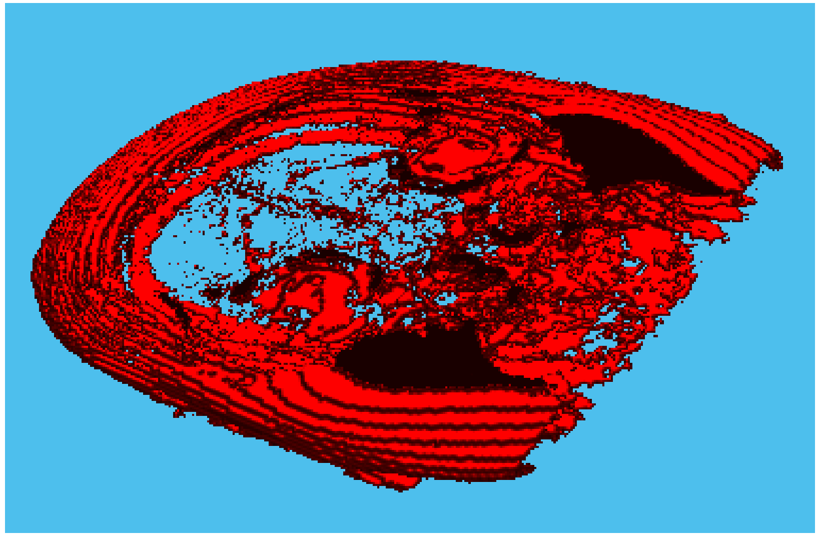
Geometric Transformations and Image Registration

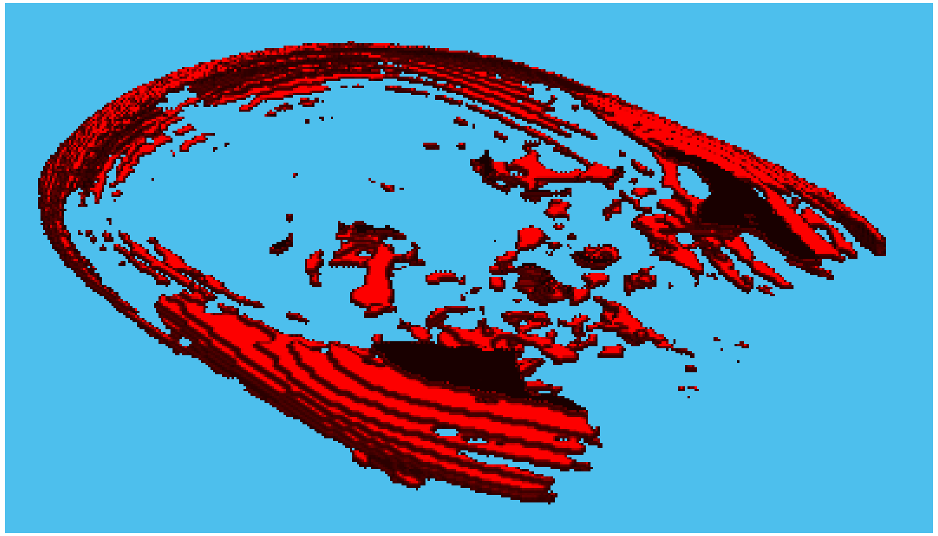
- imregister
- imregdemons
- imresize3
- imrotate3
- imwarp

Over 65 functions supported

3-D Morphology

R2018a





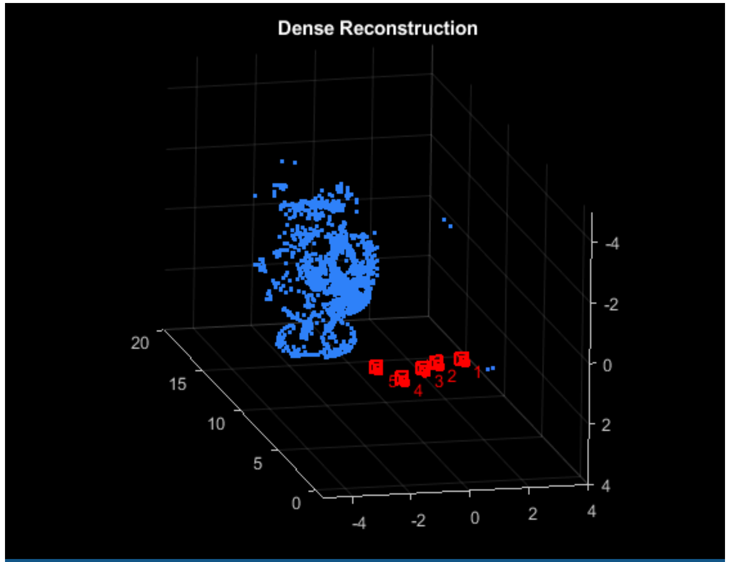
```
BW2 = bwmorph3(BW1,'majority');
```

Perform morphological operations on 3-D volumes
Perform skeletonization on all objects in 2-D image or 3-D binary volume

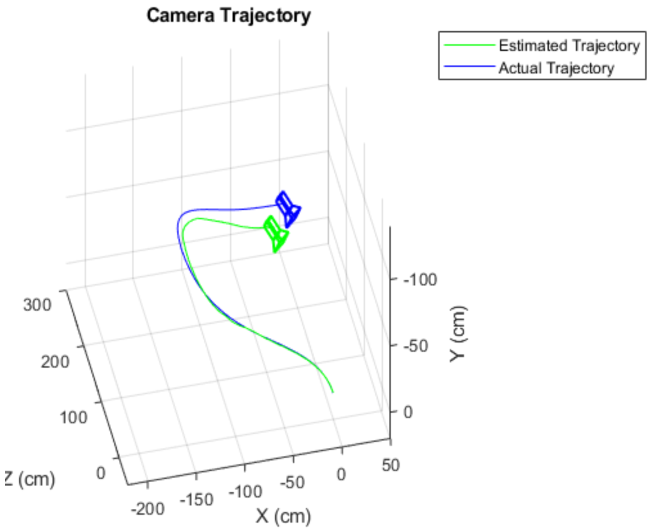
Example: bwmorph3

3-D Vision: Multi View Geometry

R2016a
R2016b



Structure from Motion



Visual Odometry

- Feature-based SfM and visual odometry workflow
- Bundle adjustment
- Multi-view triangulation

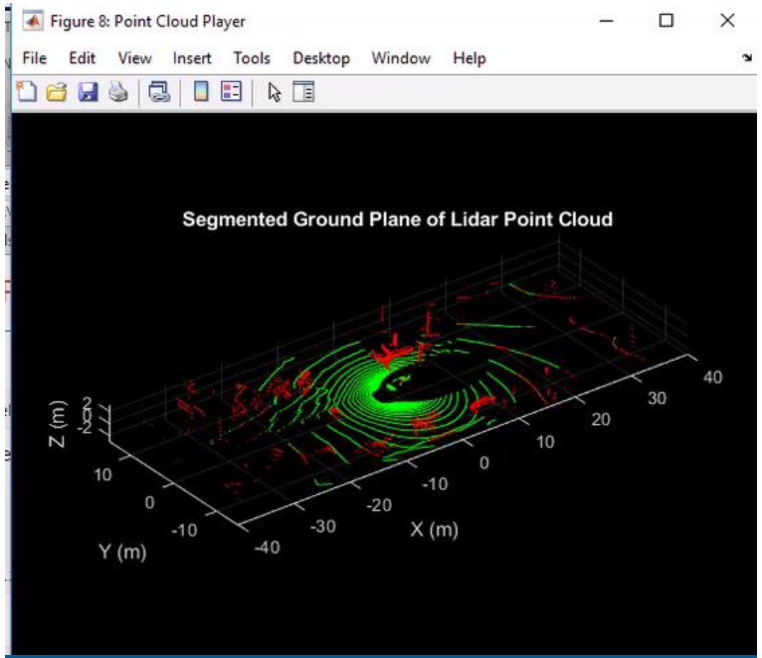
[Example: Structure From Motion From Multiple Views]

Lidar and Point Cloud Processing

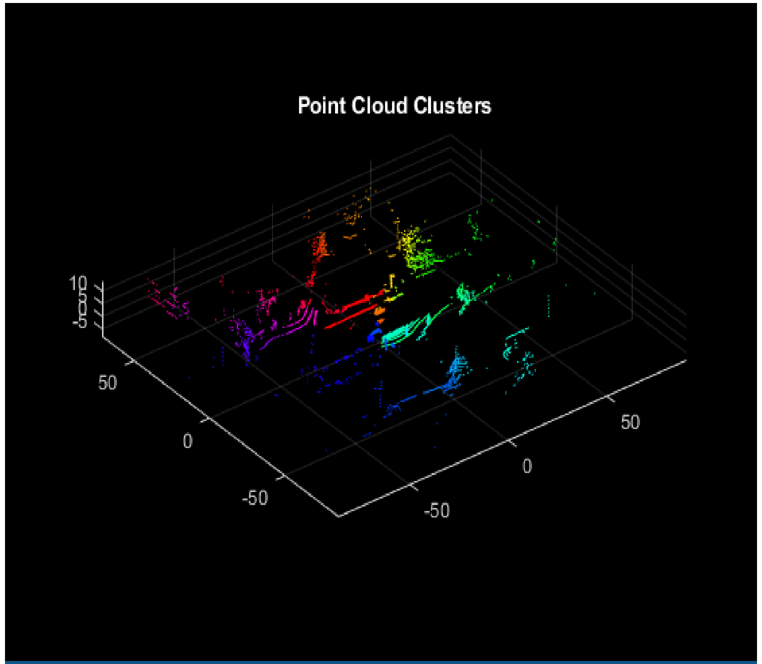
R2018b

Velodyne File Reader

```
% Read PCAP file recorded from HDL32E model sensor
veloReader = velodyneFileReader('lidarData.pcap','HDL32E');
```



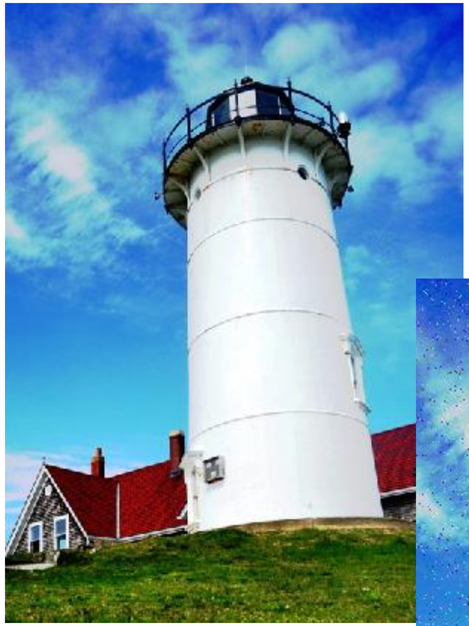
Ground Segmentation

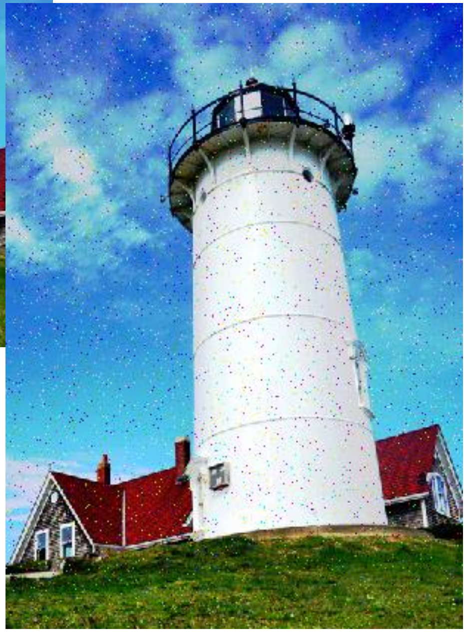


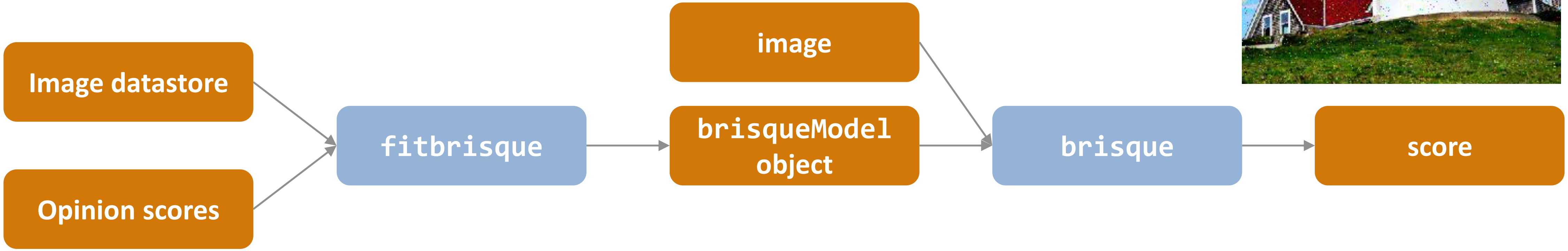
Clustering

Image Quality Metrics

Function	Measures
imabsdiff	Absolute difference
immse	Mean squared error
ssim	Structural similarity metric
psnr	Peak signal-to-noise
No-Reference Techniques (R2017b onwards)	
niqe	Naturalness Image Quality Evaluator
brisque	Blind Reference-less Image Spatial Quality Evaluator
piqe	Perception-based Image Quality Evaluator



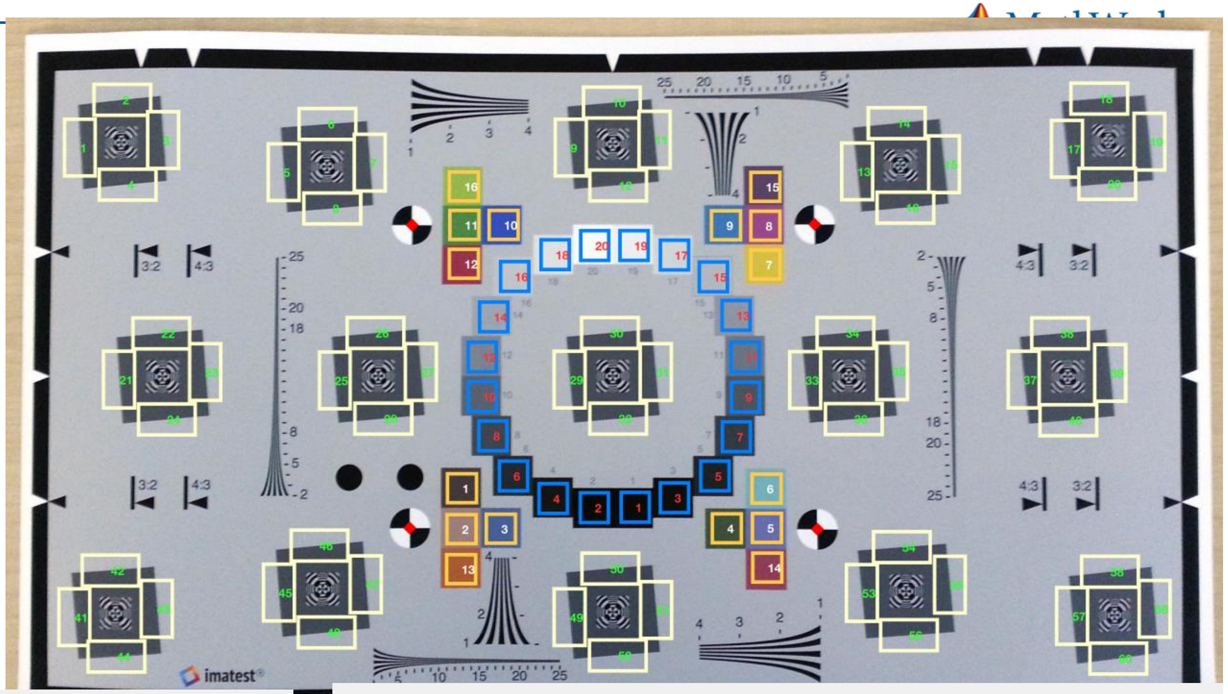


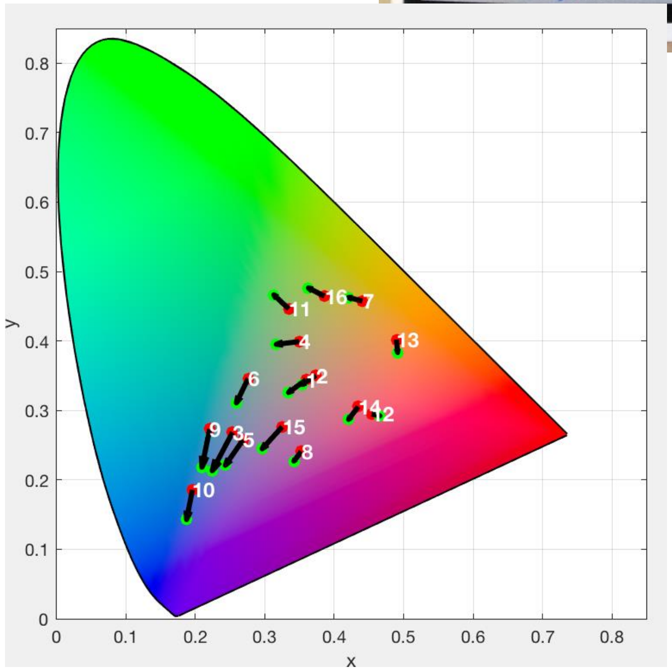


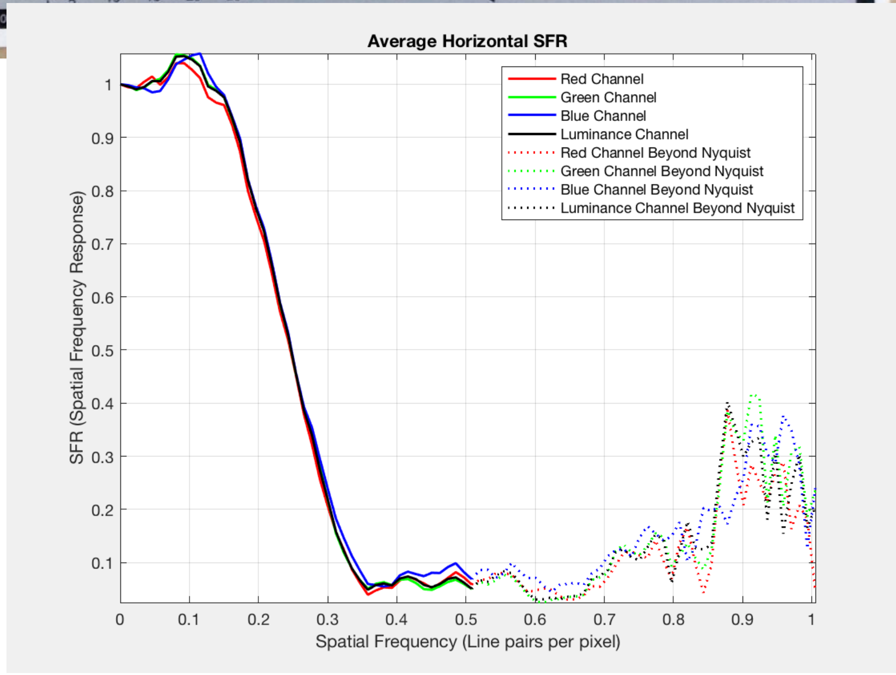
[Example: Train and Use a No-Reference Quality Assessment Model]

Test Chart Measurements

- Use imatest eSFR chart (ISO 12233)
- Automatically detect chart
- Measure:
  - Sharpness
  - Noise
  - Illuminant
  - Color accuracy
  - Chromatic aberration
- Access to raw data







Illuminant Estimation and Correction

IPT has added a bunch of color characterization and correction functions in recent releases

- `illumpca`
- `illumwhite`
- `illumgray`
- `chromadapt`
- `lin2rgb`
- `rgb2lin`

Estimate white point color from whole image using different models

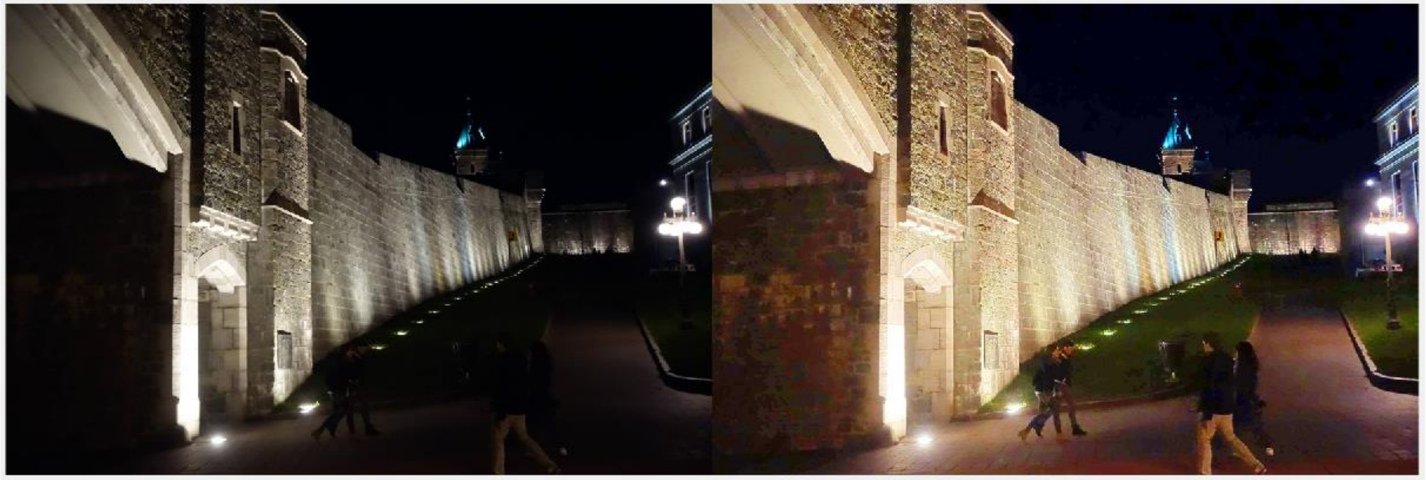
Change color using white point

Helper functions

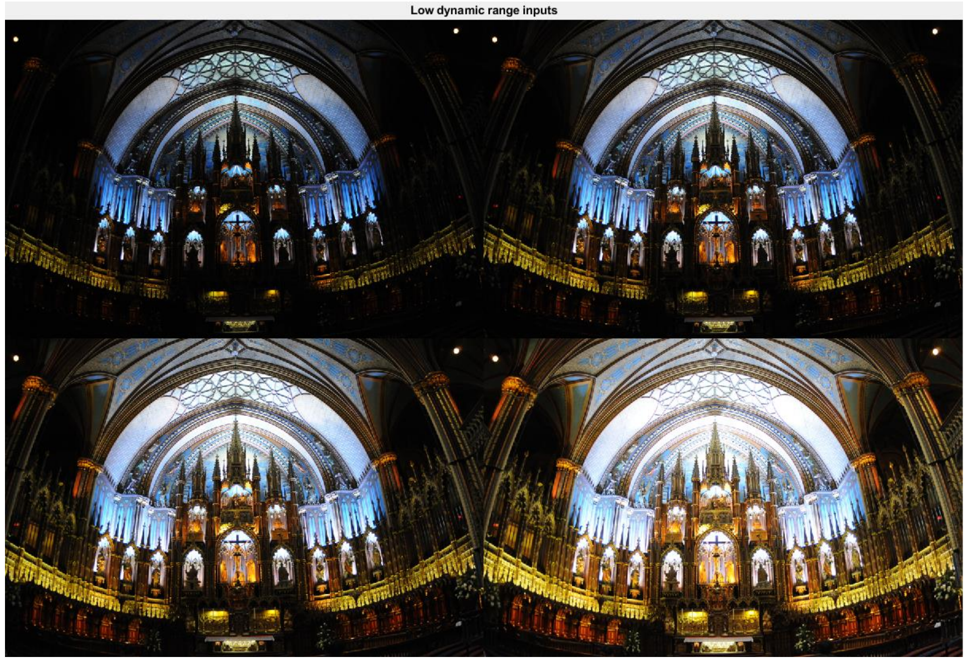
Haze reduction & Low-light enhancement

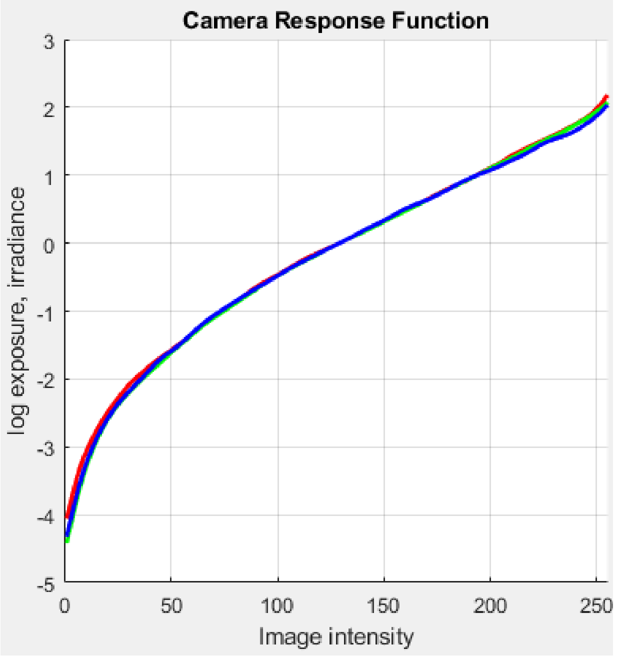
- `imreducehaze`: use "dark channel prior" algorithm to reduce haze

- Same algorithm can brighten dark images with `imlocalbrighten`



HDR





Low dynamic range (LDR) images

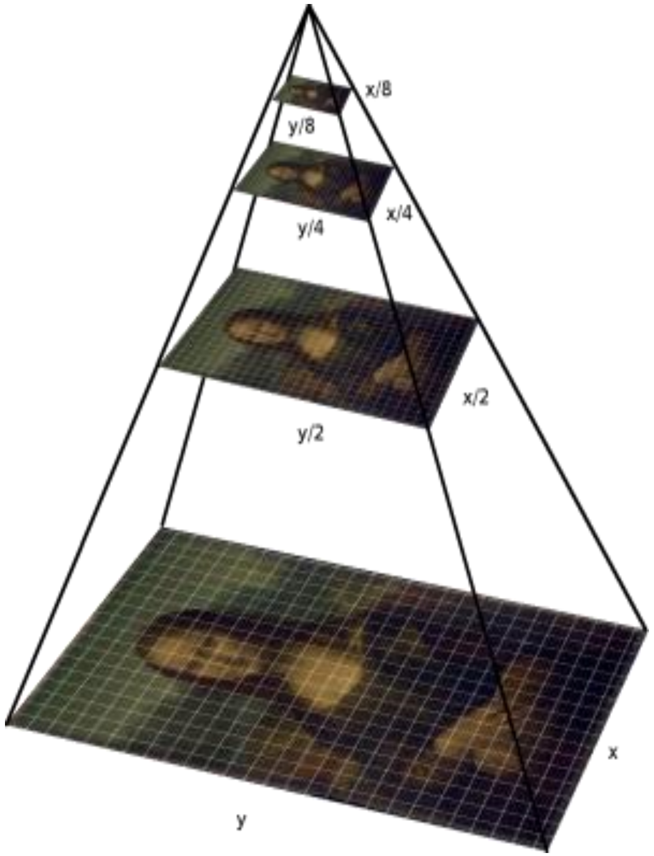
Registration

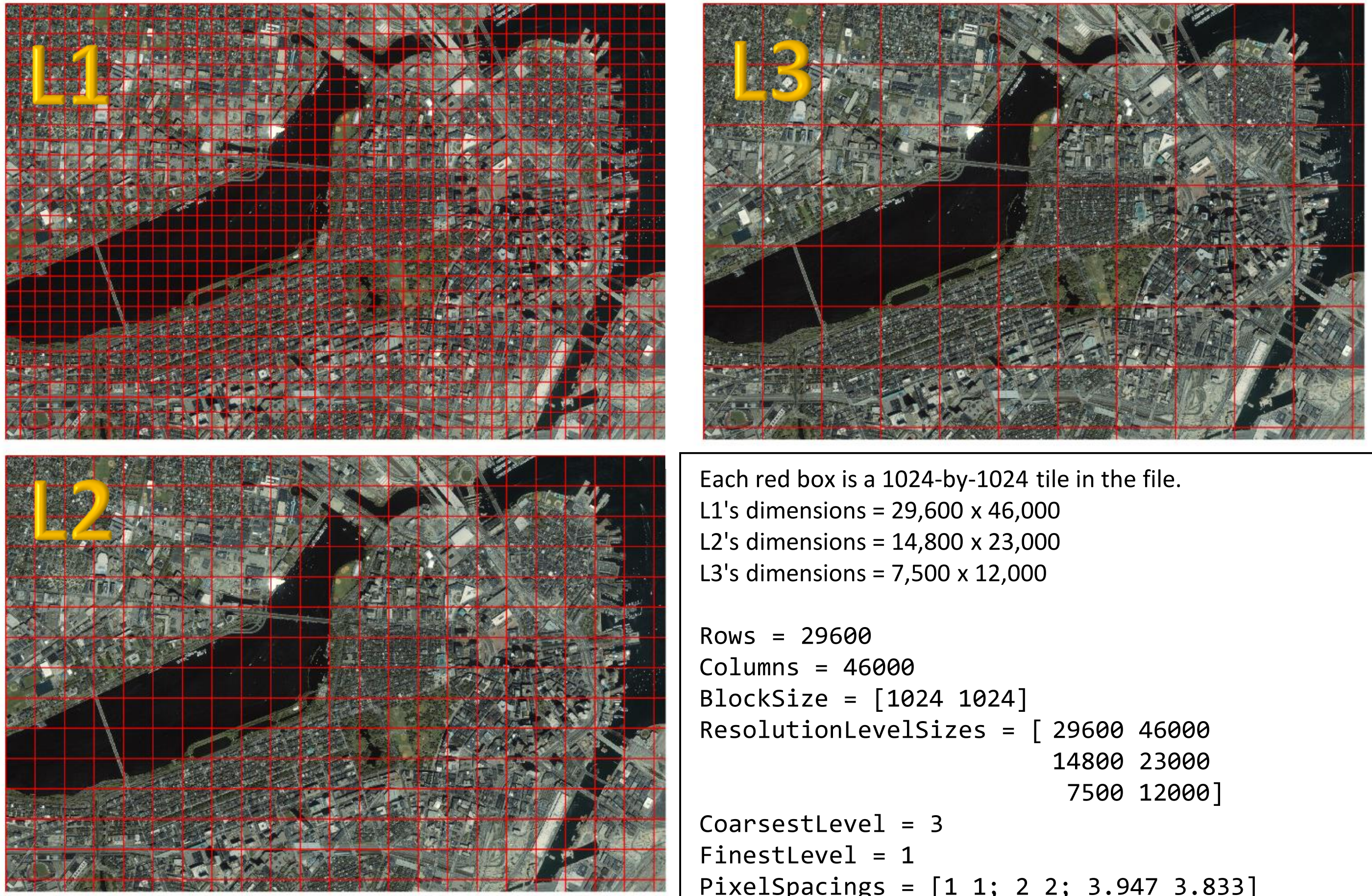
Camera response estimation

LDR → HDR

Tone mapping for display

bigImage : Multiresolution and tiled imagery





Each red box is a 1024-by-1024 tile in the file.
L1's dimensions = 29,600 x 46,000
L2's dimensions = 14,800 x 23,000
L3's dimensions = 7,500 x 12,000

```
Rows = 29600
Columns = 46000
BlockSize = [1024 1024]
ResolutionLevelSizes = [ 29600 46000
                         14800 23000
                          7500 12000]
CoarsestLevel = 3
FinestLevel = 1
PixelSpacings = [1 1; 2 2; 3.947 3.833]
```

```
>> img = bigimage('myReallyBigFile.tif');
```

GPU Acceleration with NVIDIA

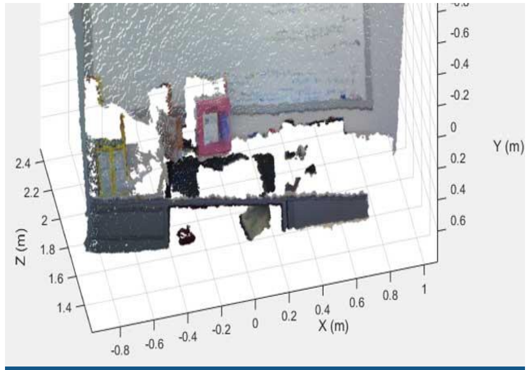
- Hundreds of MATLAB functions
- More than 50 of the most popular Image Processing functions


```
bwmorph      imhist
bwlookup     imnoise
corr2        imopen
edge         imresize
histeq       imrotate
imadjust     imshow
imbothat     imtophat
imclose      imwarp
imdilate     mean2
imerode      medfilt2
imfilter     padarray
imgradient   rgb2gray
```

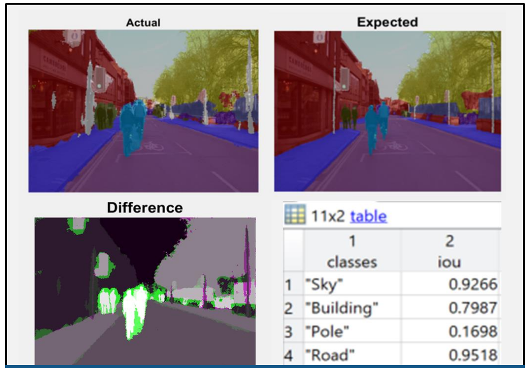
www.mathworks.com/products/parallel-computing/builtin-parallel-support.html

What’s New in Image Processing and Computer Vision?

Apps



Algorithms

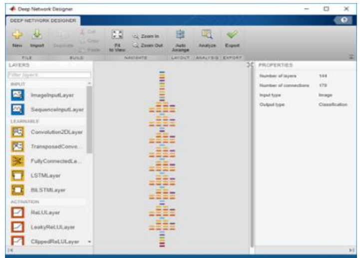


Deep Learning

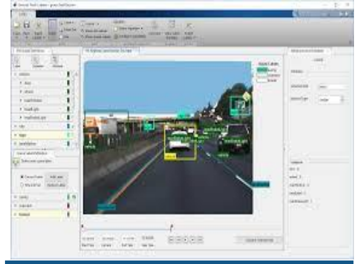
Access

Deep Learning for Image Processing and Computer Vision

ACCESS

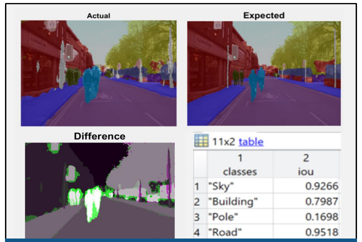


TRAINING INTERFACES

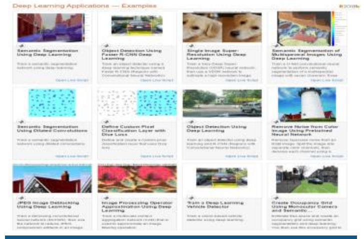


LABELING

PRE-PROCESSING

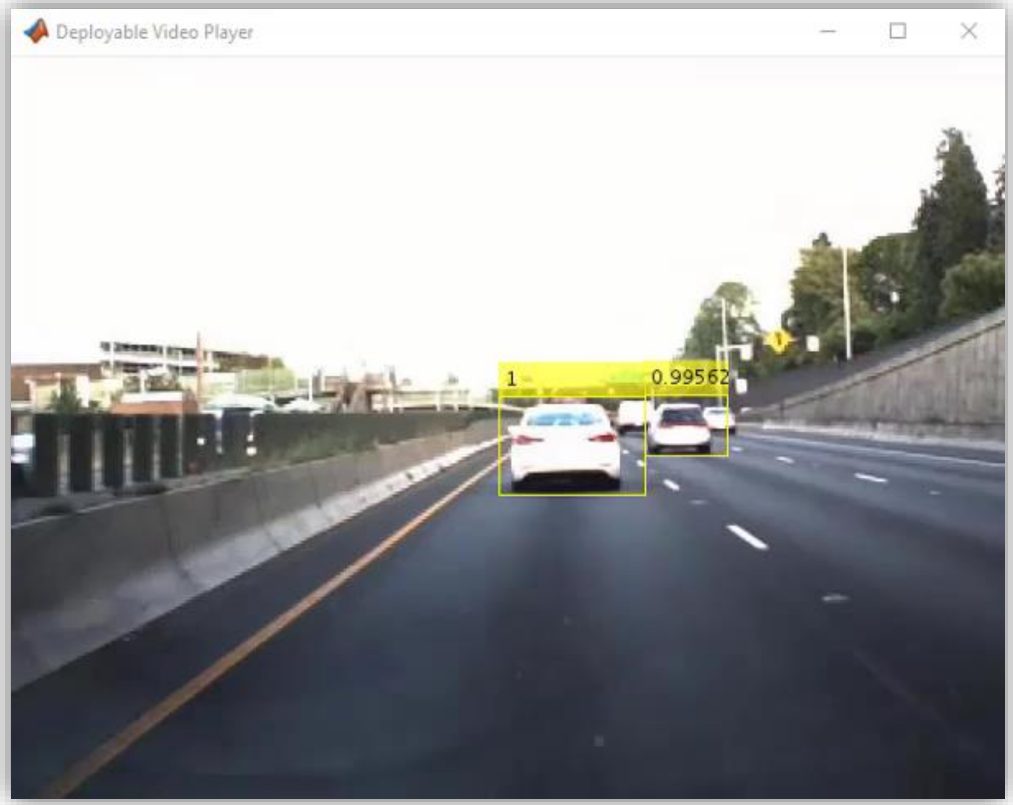


EVALUATION TOOLS



EXAMPLES

CODE GENERATION



Object Detection

R-CNN

Fast R-CNN

Faster R-CNN

YOLO v2

Semantic Segmentation

SegNet

FC

U-Net

Computer Vision

Train

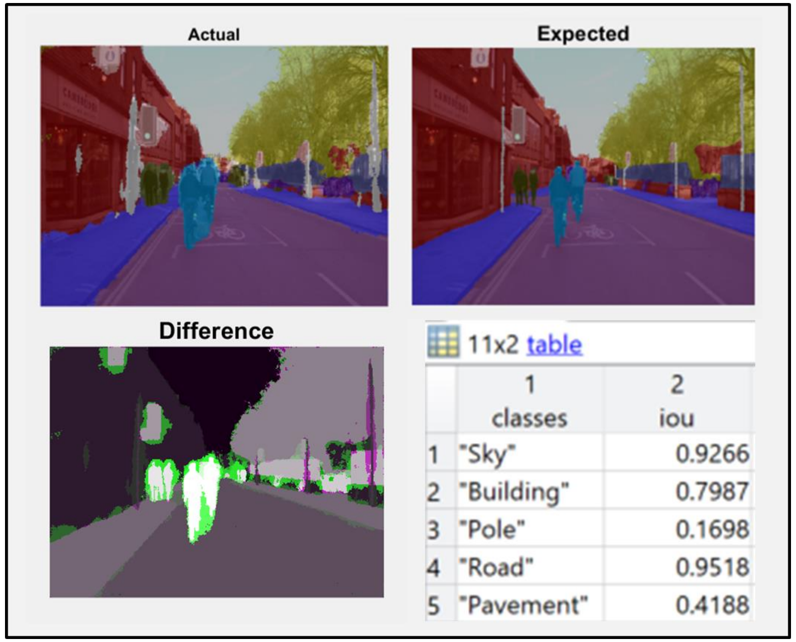
SegNet

FC

U-Net

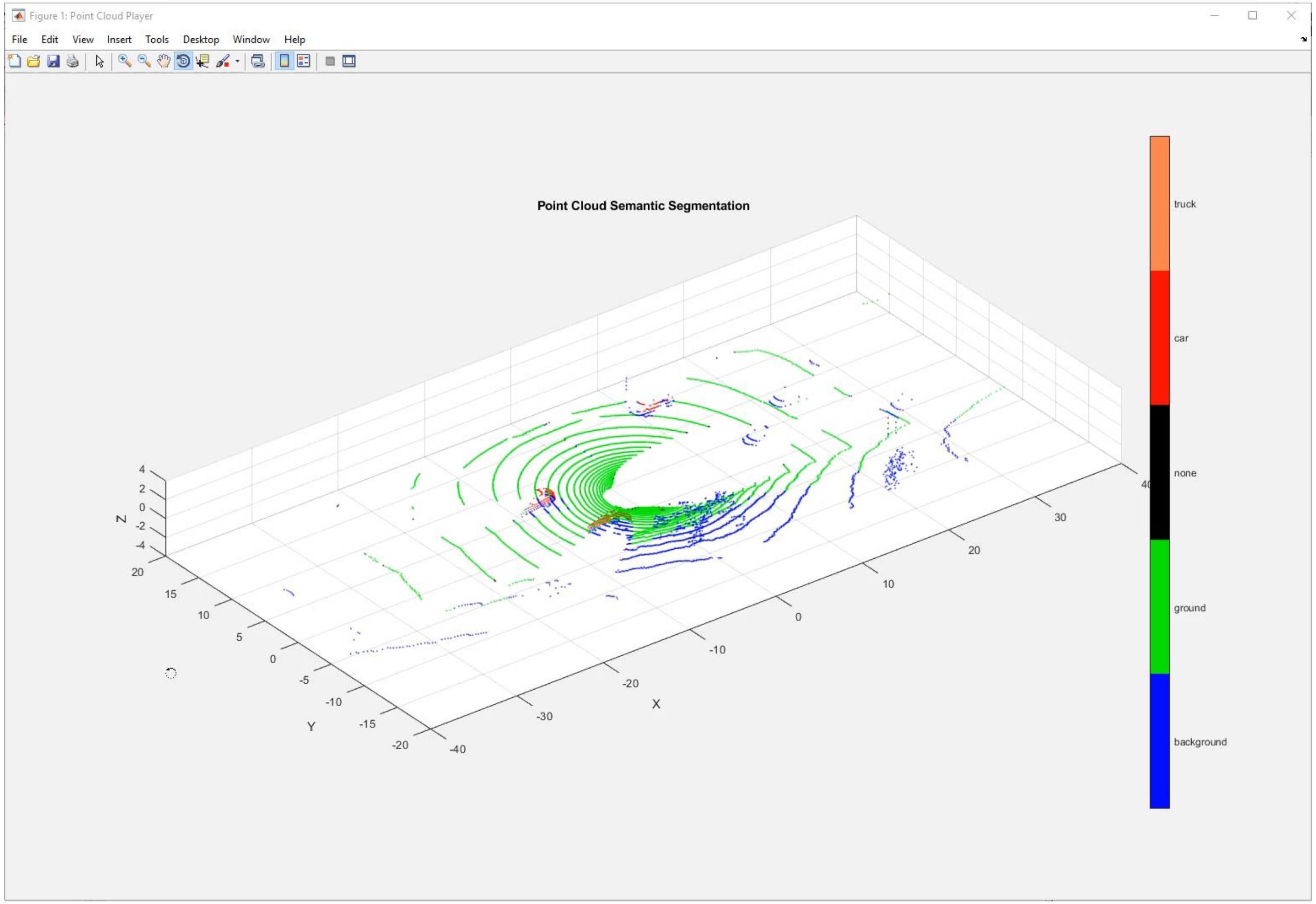
Automatically create network architectures

Evaluate precision of semantic segmentation network

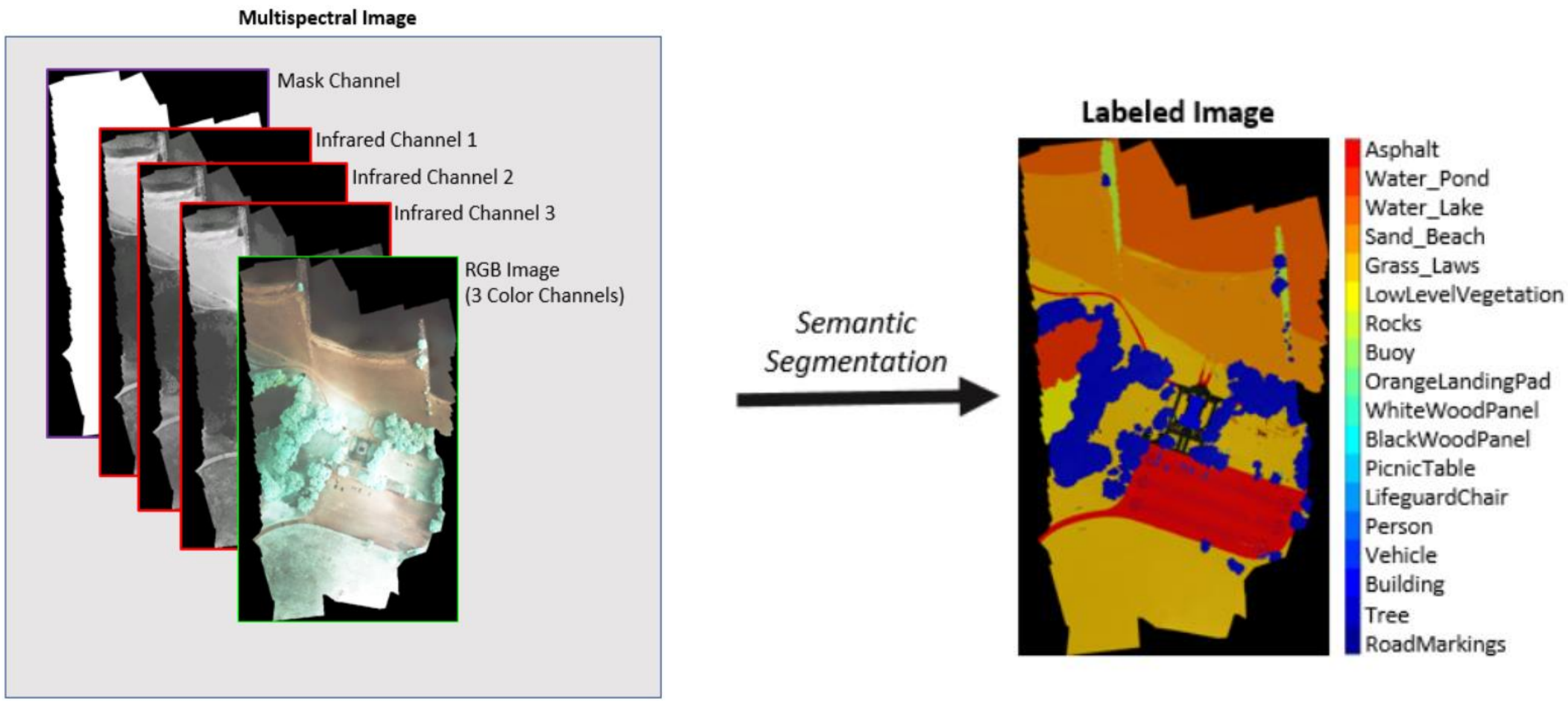


	1 classes	2 iou
1	"Sky"	0.9266
2	"Building"	0.7987
3	"Pole"	0.1698
4	"Road"	0.9518
5	"Pavement"	0.4188

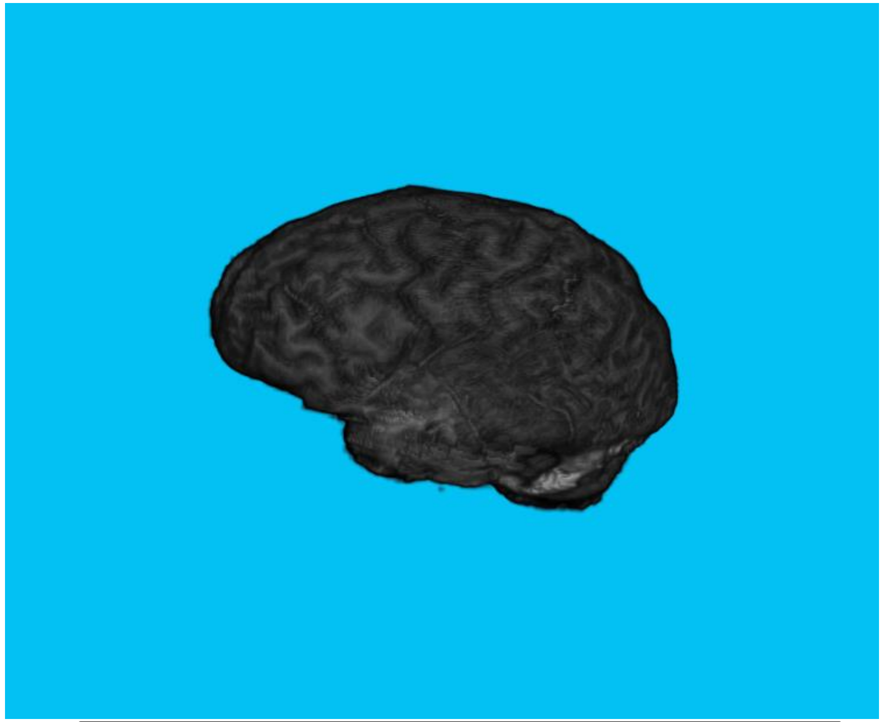
Semantic Segmentation of Lidar Data



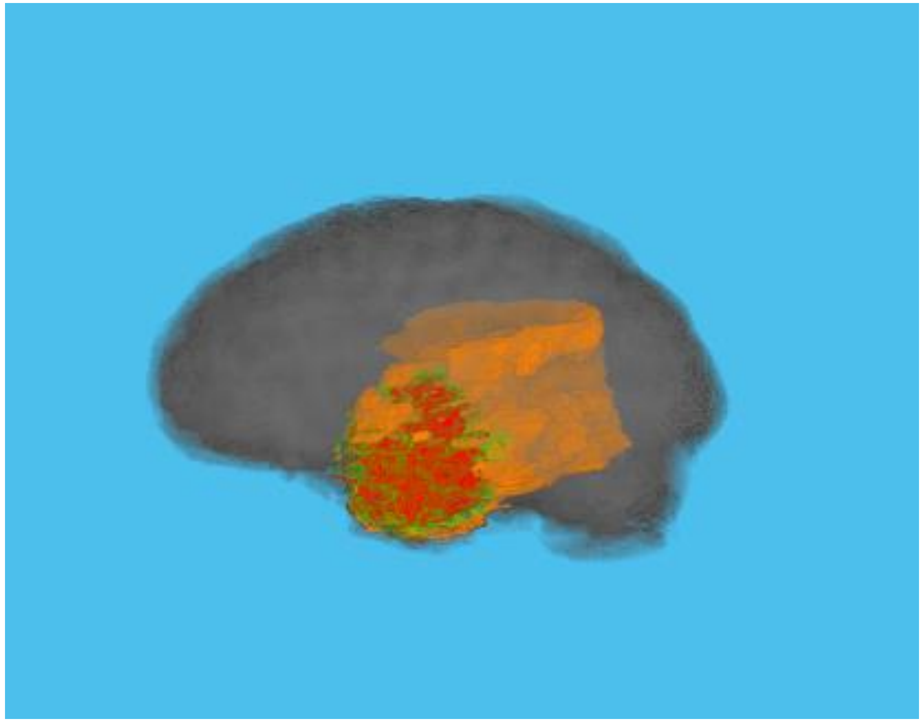
Semantic Segmentation of Hyperspectral Images



3D Volume Support for Deep Learning



Train 3D CNN to detect brain tumors in MRI scan



3-D Deep Learning Support in R2019a

- image3dInputLayer
- convolution3dLayer
- maxPooling3dLayer
- avgPooling3dLayer
- transposedConv3dLayer
- concatenationLayer
- reluLayer
- leakyReluLayer
- clippedReluLayer
- softmaxLayer
- fullyConnectedLayer
- batchNormalizationLayer
- dropoutLayer
- additionLayer
- classificationLayer
- regressionLayer
- pixelClassificationLayer
- semanticSeg
- evaluateSemanticSeg
- ImageDatastore
- PixelLabelDatastore
- randomPatchExtractionDatastore

Deep Learning for Image Processing

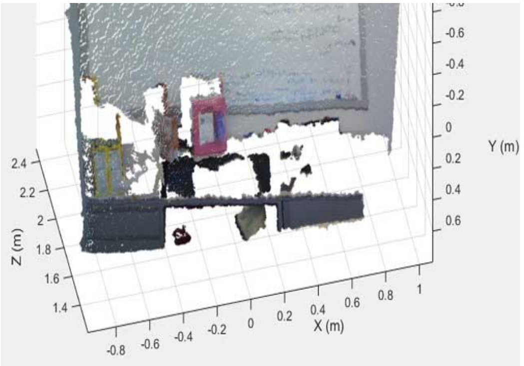
R2017b

Denoise Image using a pretrained network or a network you created

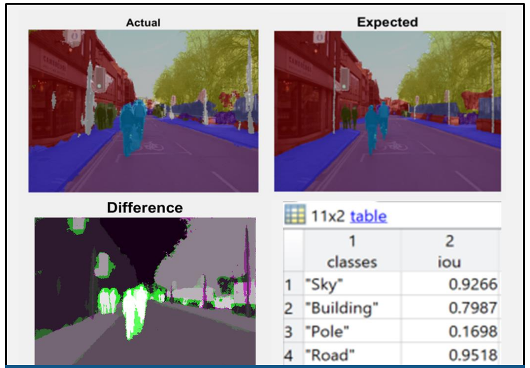
Example: Remove Noise from Color Image Using Pretrained Neural Network

What’s New in Image Processing and Computer Vision?

Apps



Algorithms



Deep Learning

Access

OpenCV Interface

- Bring OpenCV C/C++ code into MATLAB using MEX
- Ships with OpenCV-binaries (v3.4.0 as of 2018b)
- Large library of data type conversions
- Several examples to help get started

Install and Use Computer Vision Toolbox OpenCV Interface R2019a

Use the OpenCV Interface files to integrate your OpenCV C++ code into MATLAB® and build MEX-files that call OpenCV functions. The support package also contains graphics processing unit (GPU) support.

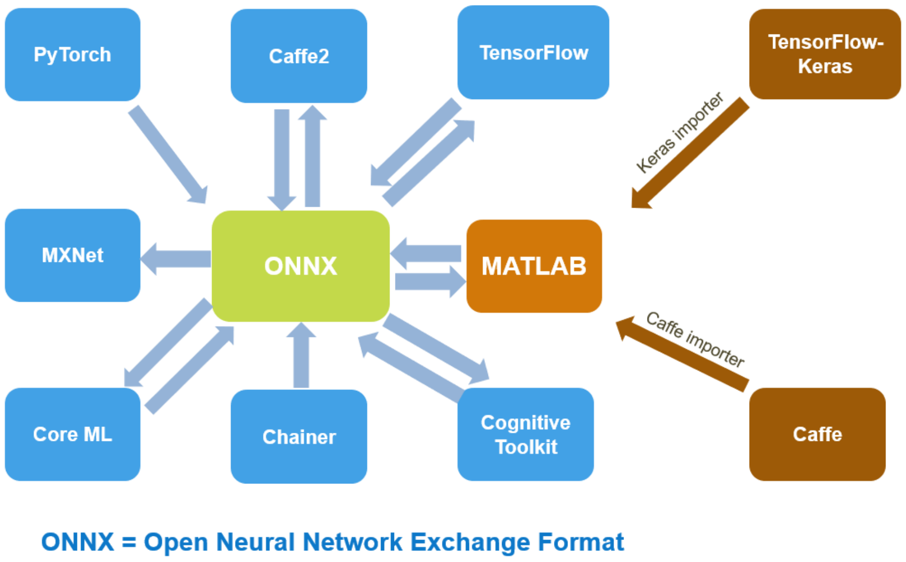
- Installation
- Support Package Contents
- Create MEX-File from OpenCV C++ file
- Use the OpenCV Interface C++ API
- Create Your Own OpenCV MEX-files
- Run OpenCV Examples

Installation

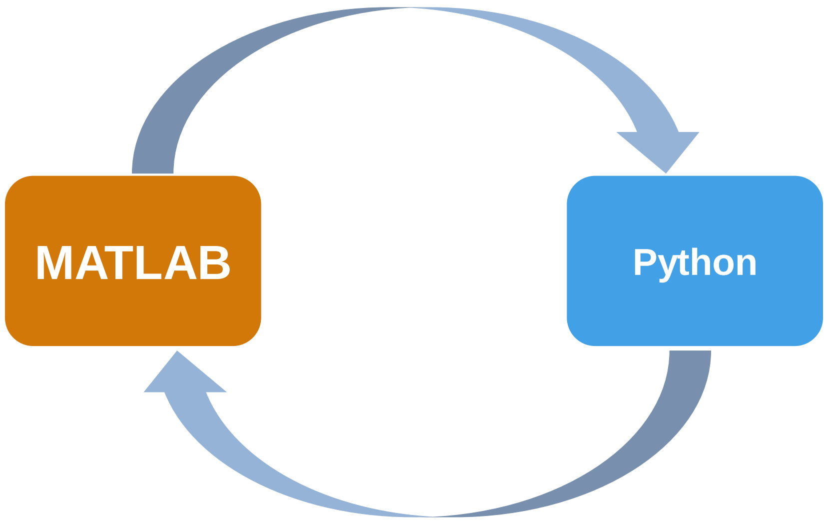
After you install third-party support files, you can use the data with the Computer Vision Toolbox™ product. Use one of two ways to install the Add-on support files.

- Select **Get Add-ons** from the **Add-ons** drop-down menu from the MATLAB desktop. The Add-on files are in the "MathWorks Features" section.
- Type `visionSupportPackages` in a MATLAB Command Window and follow the prompts.

Two Ways to Work with TensorFlow, Caffe, and PyTorch



Model Exchange



Co-execution (Python and C++)

Call Other Languages from MATLAB

- C/C++ code

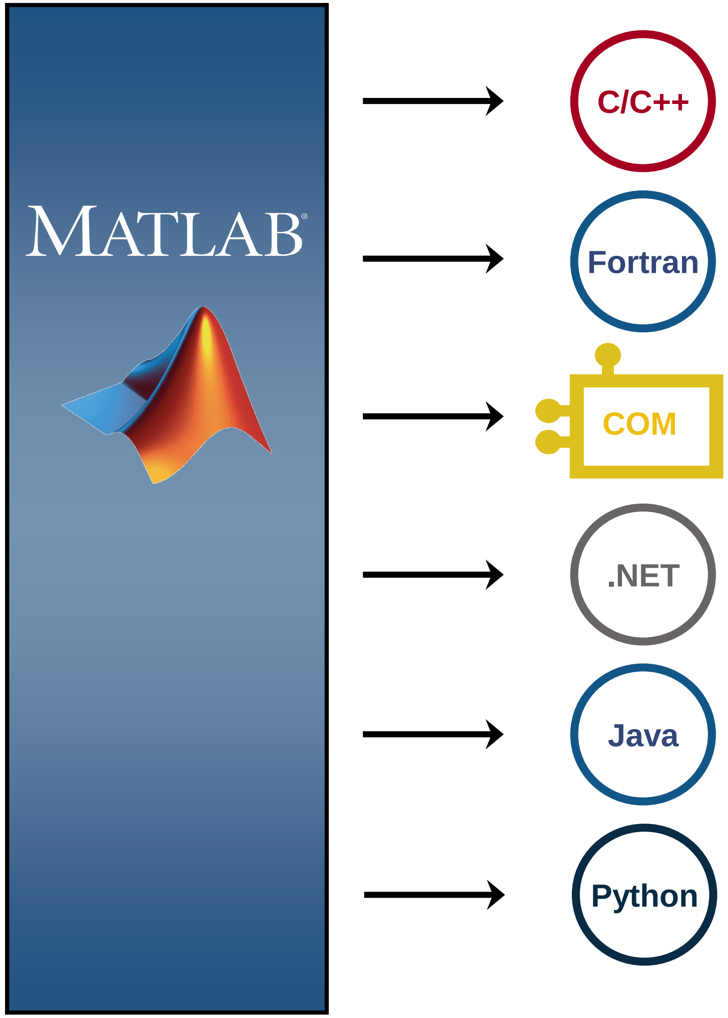
```
>> mex foo.c
>> mex foo.cpp
>> coder.ceval(‘foo’,arg1)
```

- C/C++ libraries

```
>> calllib(foolib, foo, arg1)
>> clib.foo(arg1)
```

- Python (support for CPython versions 2.7, 3.5, 3.6, and 3.7)

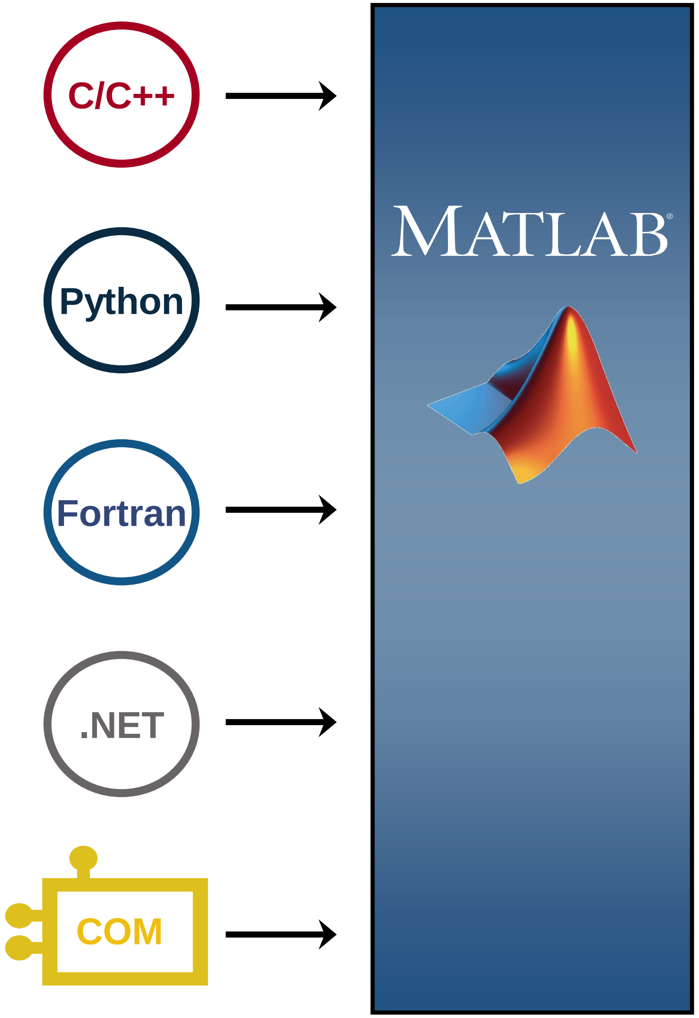
```
>> py.foolib.foo(arg1)
```



Call MATLAB from Other Languages

- MATLAB Engine API
  - C/C++
  - Python
  - Fortran

- Automation server
  - COM
  - .NET



Anomaly and Defect Detection

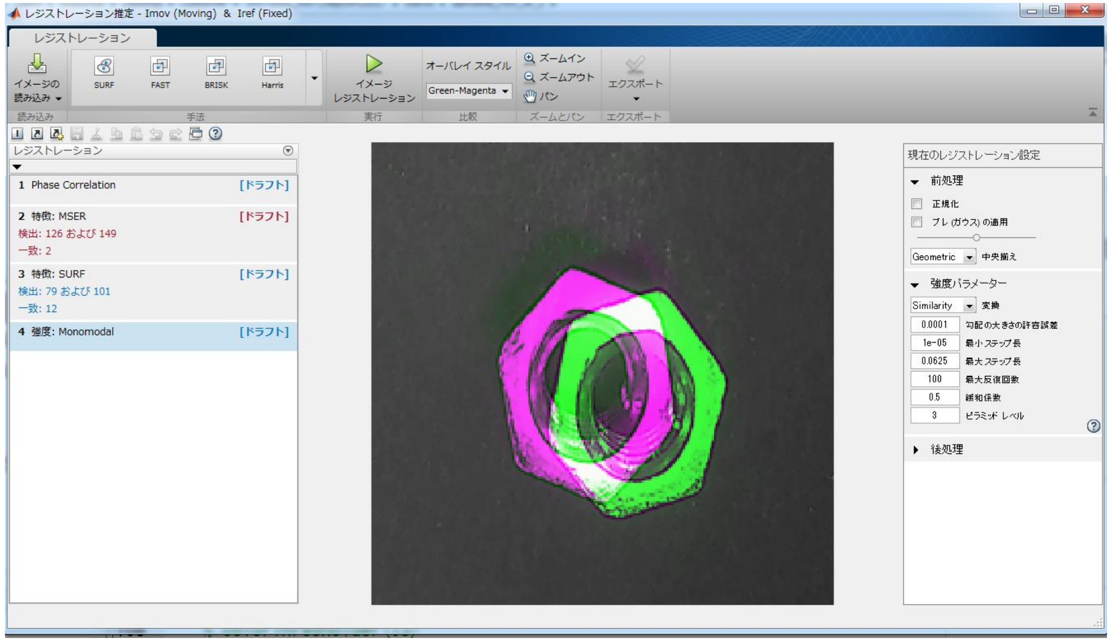
How MathWorks tools can help with defect detection

1. Preprocessing tools
2. Traditional techniques
3. Defect detection using AlexNet
4. Anomaly detection using a “Convolutional Auto Encoder” (CAE)

Preprocessing

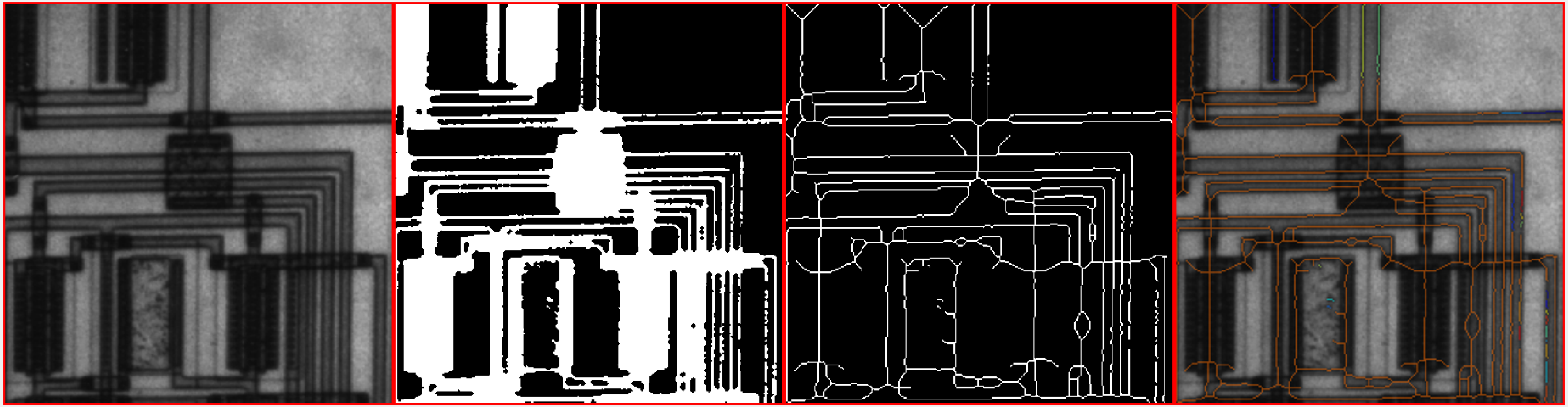
Image Processing Toolbox has the fundamentals

- Registration
  - Intensity, features, phase correlation, …
- Intensity adjustment
  - `stretchlim`, `histeq`, `adapthisteq`
- Denoising
  - Edge-preserving filters



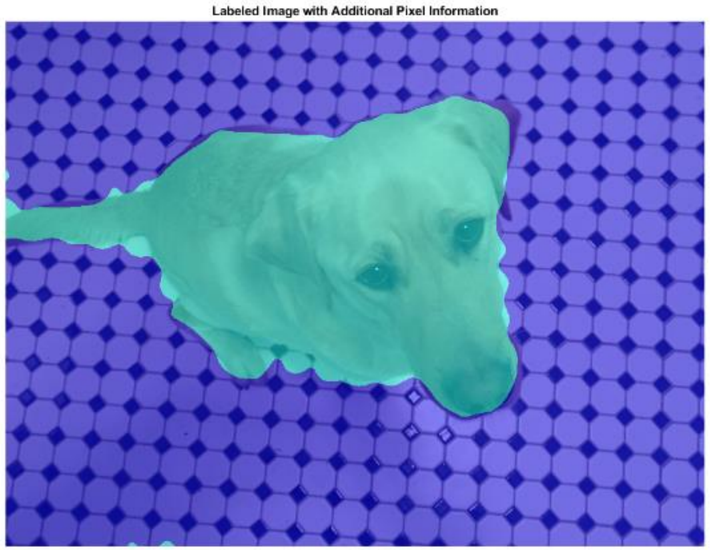
Segmentation: Thresholding and morphology

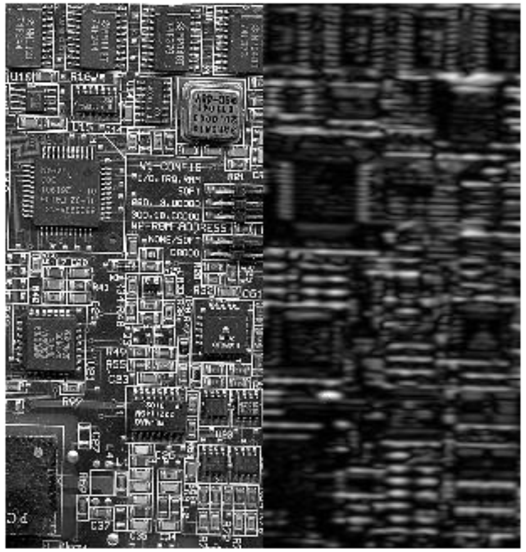
Original image | thresholding + `imclose` | `bwmorph` | `bwlabel` + `labeloverlay`



Segmentation: Advanced techniques

- Simple N-class clustering: `imsegkmeans`
- Find “threadlike” objects: `fibermetric`
- Texture analysis: `imgaborfilt`





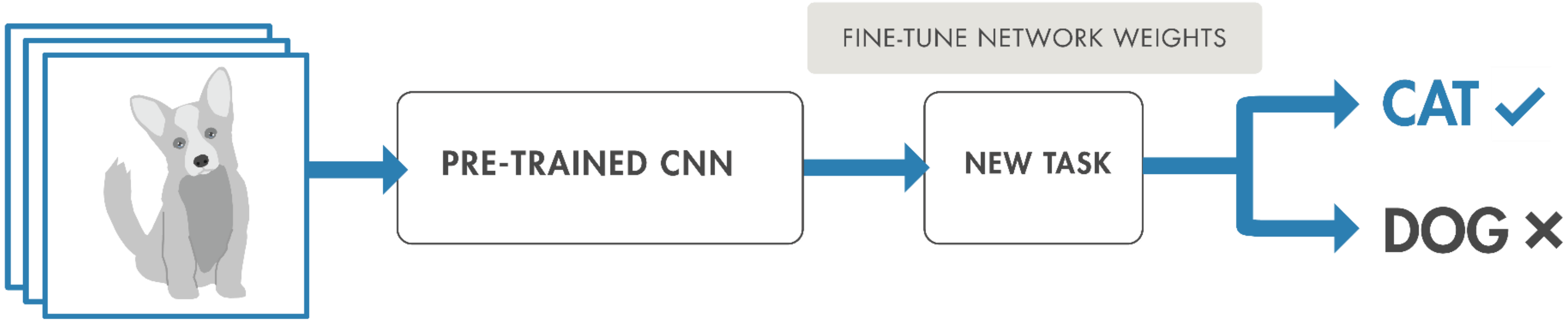
Finding defects with machine learning

1. AlexNet and a 1-class SVM

2. Anomaly detection using a “Convolutional Auto Encoder” (CAE)

Defect detection using AlexNet: General supervised learning

Fine-tune a pre-trained model (transfer learning)



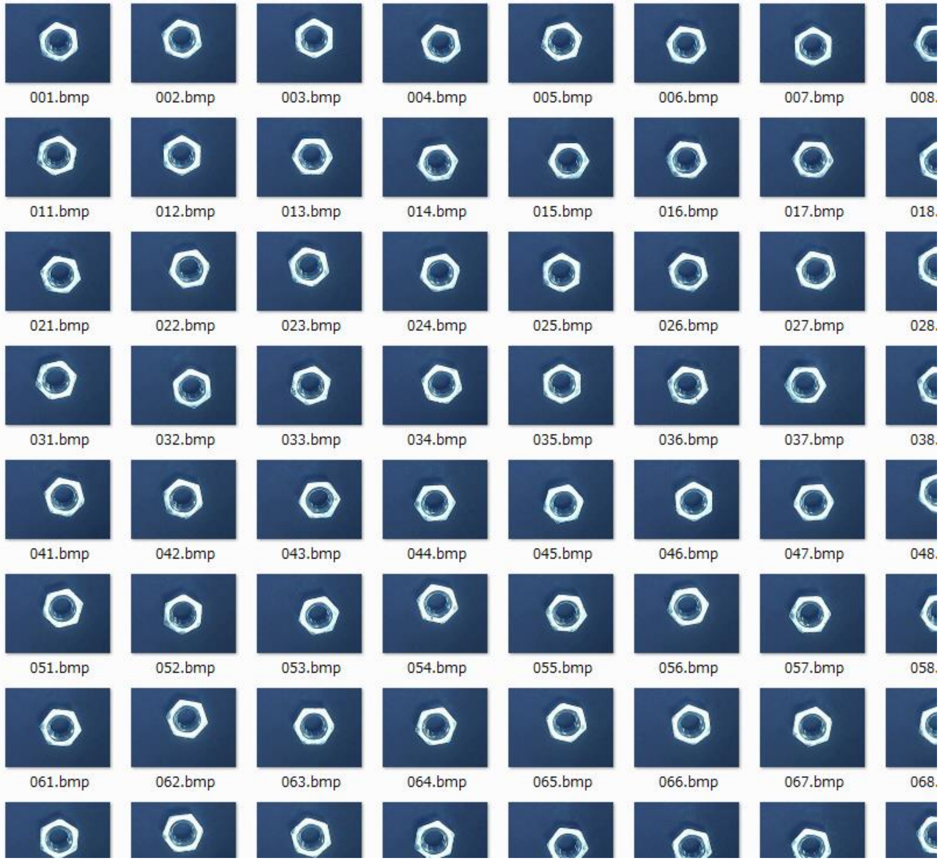
- High Accuracy on Image classification
- Transfer Learning is easy to use

- Prepare image data with labels
- same number of images for each category

Defect detection using AlexNet: The task

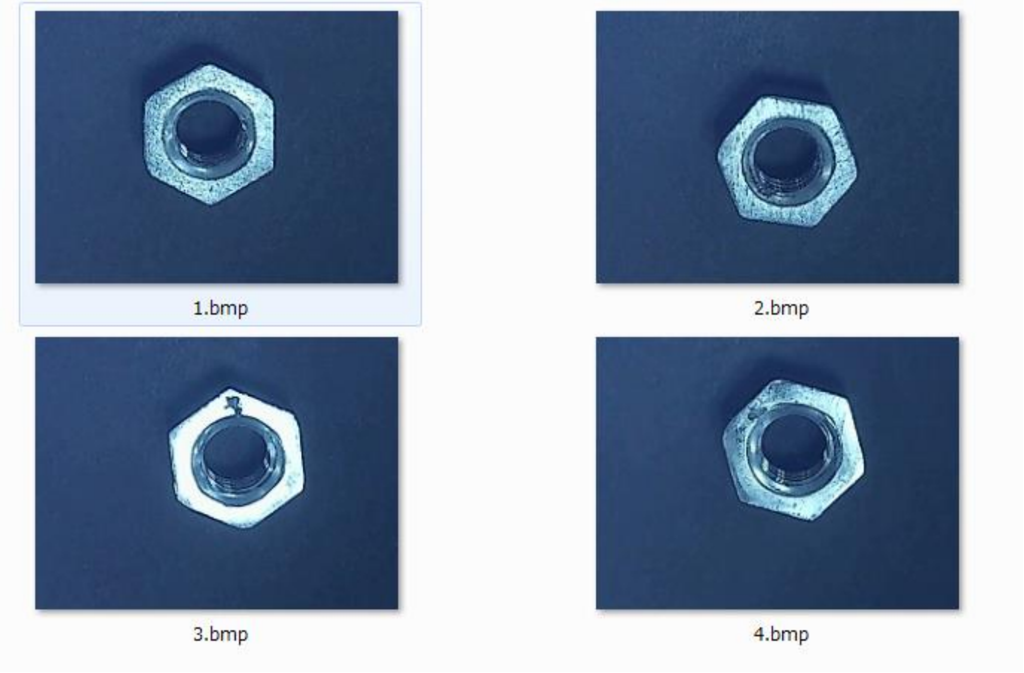
Deep Learning: Image anomaly detection for production line ~find scratched units~

100 hexagon nuts



Task: Find 4 defective units in 100 test images.

4 Defective units

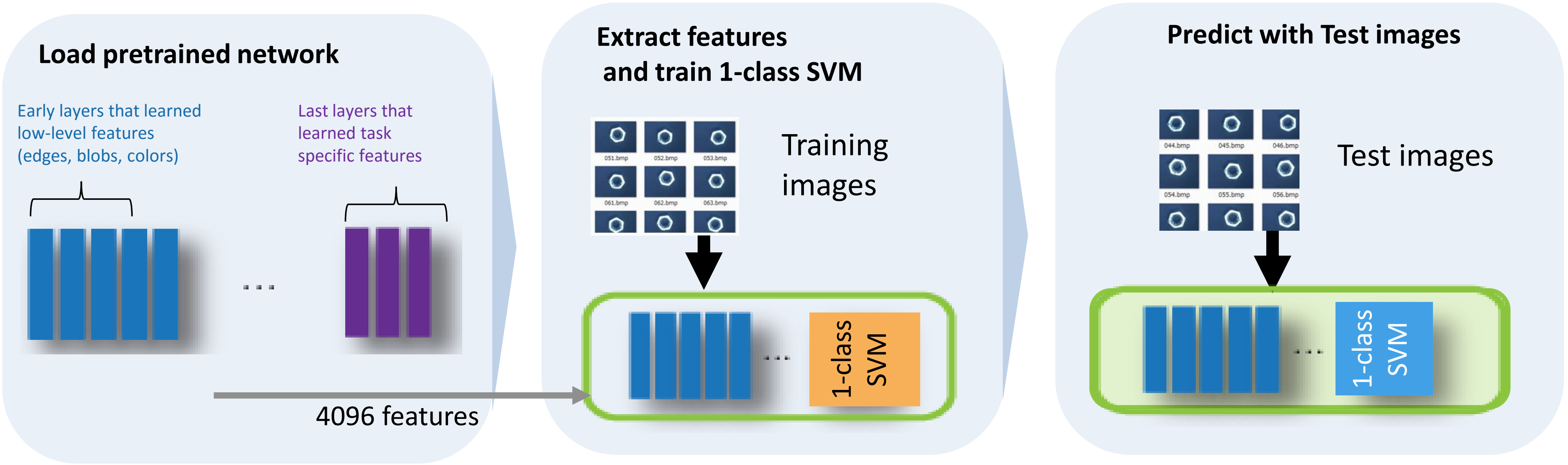


Challenge:

Number of defective units is very small.

Difficult to apply supervised learning to this task.

Defect detection using AlexNet: Hybrid CNN+SVM approach

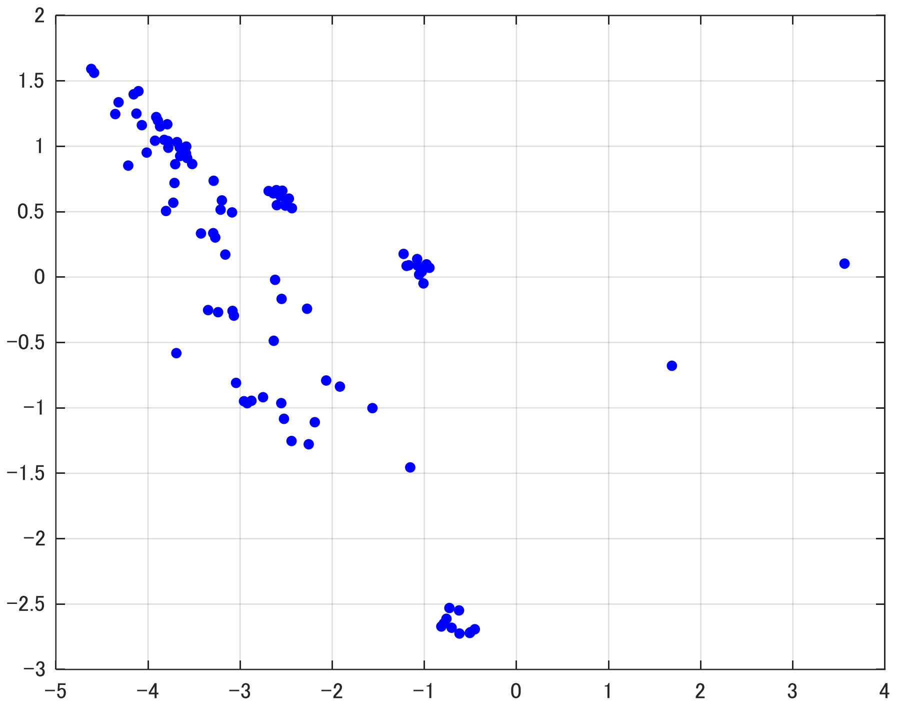


・Load pretrained AlexNet

・Extract features with pre-trained model（AlexNet）
・Train 1-class SVM with 100 images.
・Unsupervised training
・Only normal images are required for training

・Predict
・List sorted by predicted score
・Find 4 defective units from 100 test images

Defect detection using AlexNet: 1-class SVM

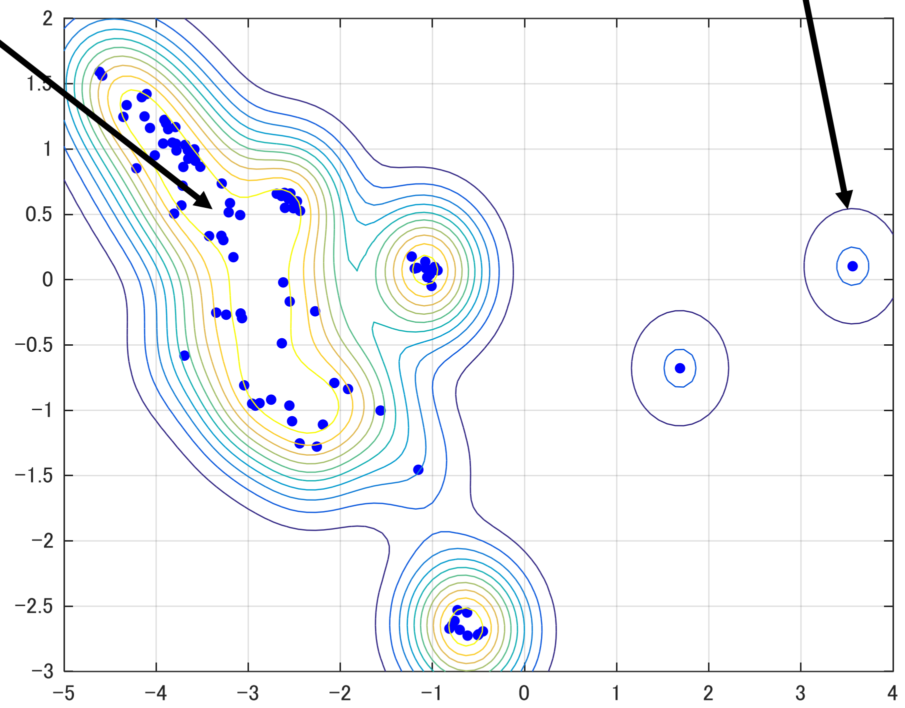


Feature distribution

1-Class SVM

High score

Low score

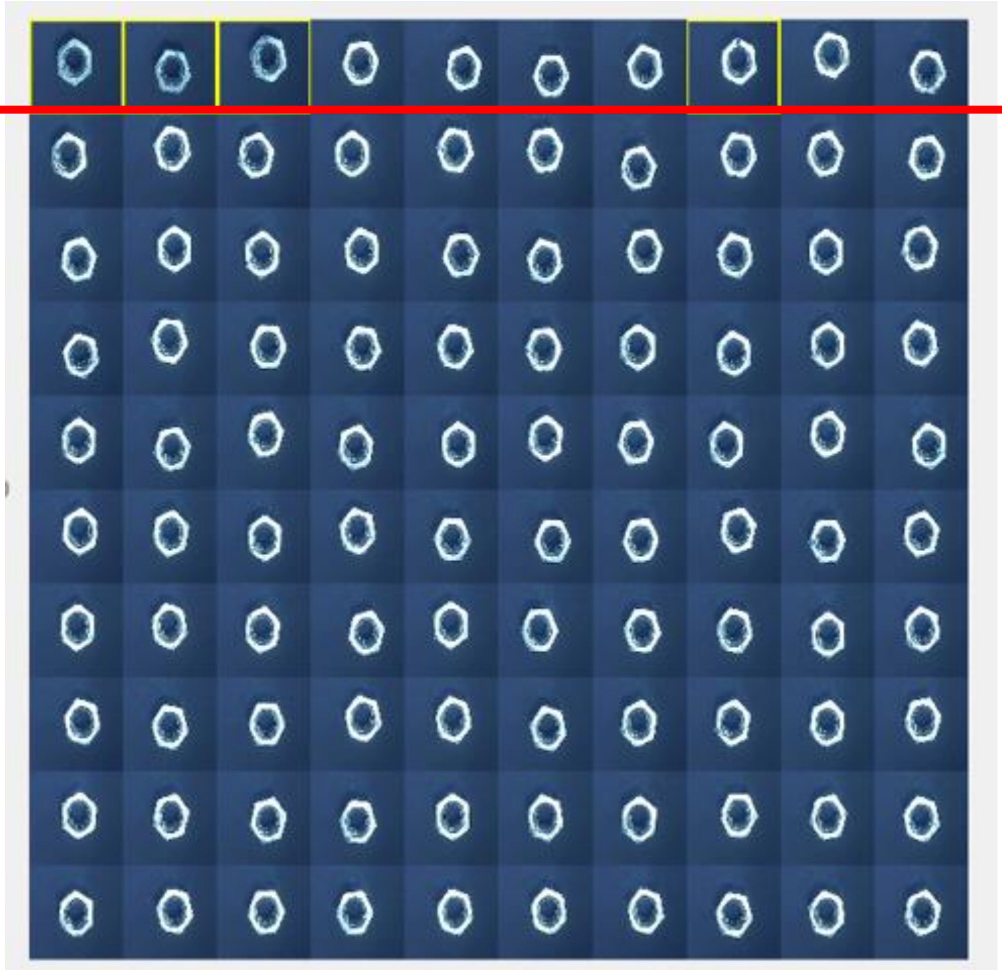


Unsupervised training
high density -> High score
Low density -> Low score

Defect detection using AlexNet: Results

Place images in ascending order according to predicted score.

Low Score (Abnormal)



High Score (Normal)

Yellow box images are units people judge as defect.

*You can find all defective unit in the first line.
* Inspectors should check only 10 %.
Deep Learning could **reduce inspection costs**.

But. not perfect.
How can we achieve better result?

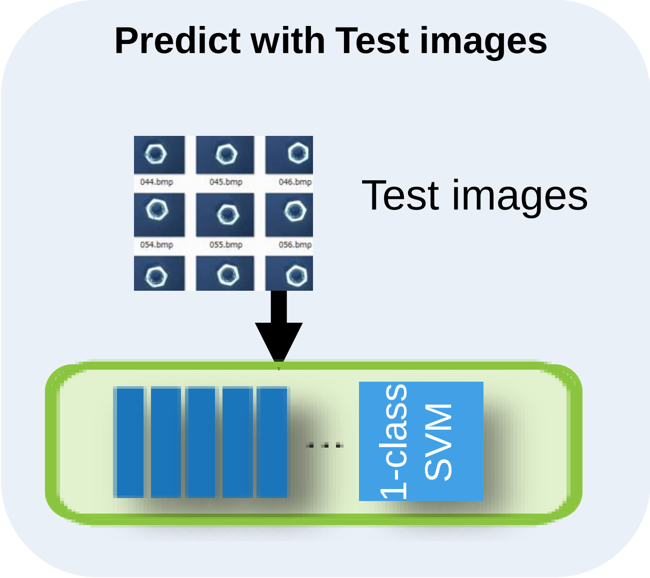
Defect detection using AlexNet: How to improve results

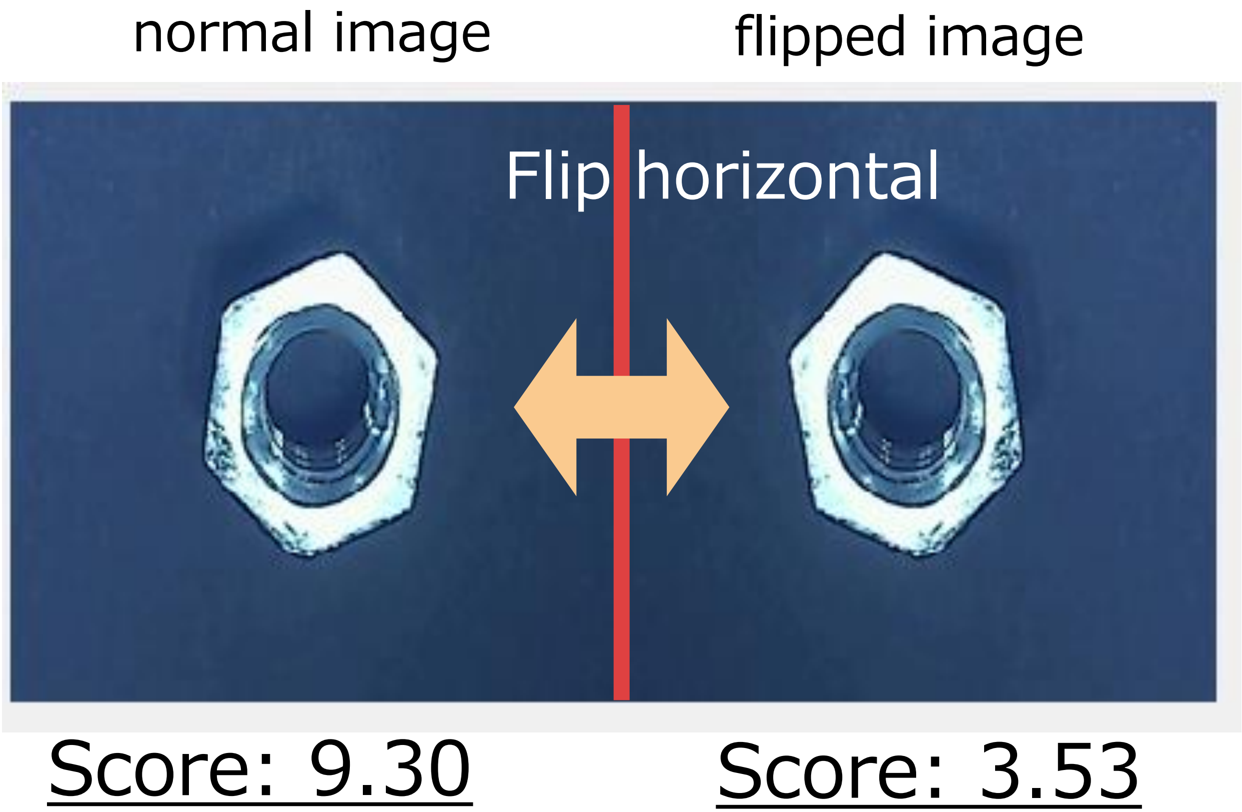
- Preprocessing / Image augmentation
- Increase the number of images
- Fine tune the hyperparameters of classifier(SVM)
- Use different pretrained network to extract features (vgg,GoogleNet…)

It may work.
But it may NOT work.
You don’t know if it works or not until applying actually.

Defect detection using AlexNet: Improving results

Function for flipping horizontally : B = fliplr(A)





Flip horizontal decreased the score

The position of the nuts and Luminance gradient(shade) also affect the predicted score.
In shipping inspection,
-People understand the meaning of the task and focus on the surface of nut.
-Deep Learning model predict a result from the whole image.

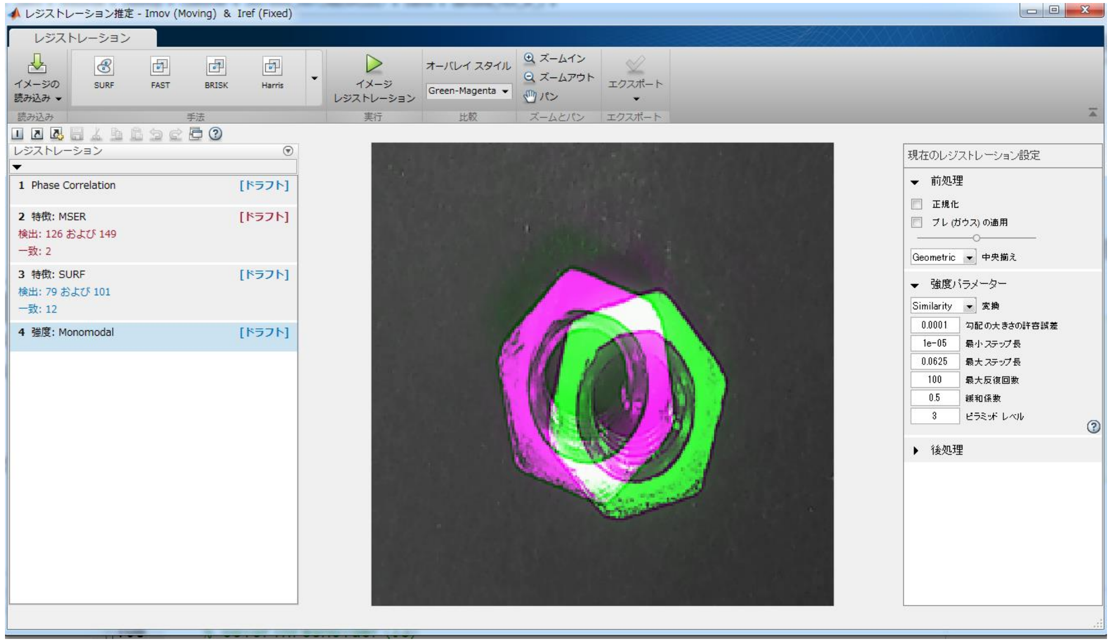
Defect detection using AlexNet: Preprocessing

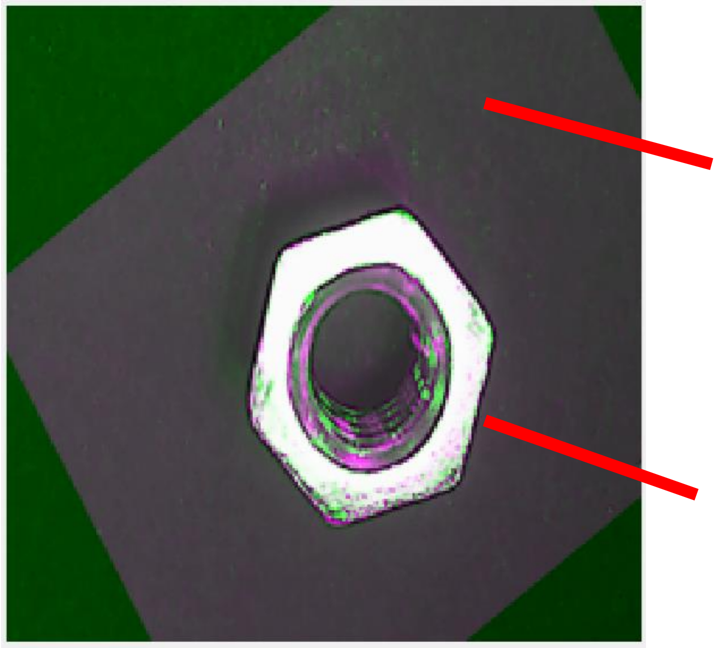
Let the model look at the surface.

-Match the image the same position and orientation as the reference image using registration

-Apply a mask to the background and set the value to zero

Registration Estimator App



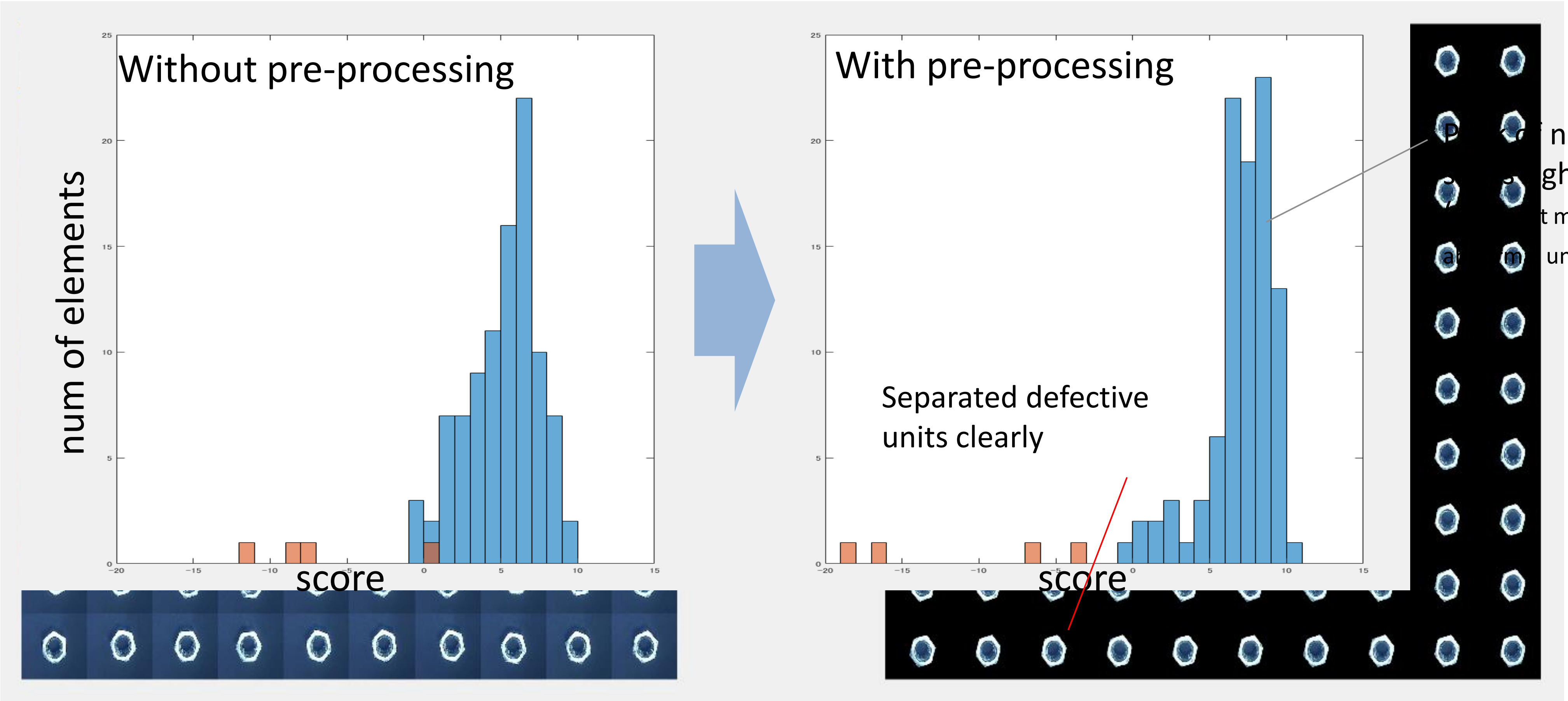


Defect detection using AlexNet: Results with preprocessing

Without pre-processing

Histogram of predicted scores

With pre-processing



-same position and orientation

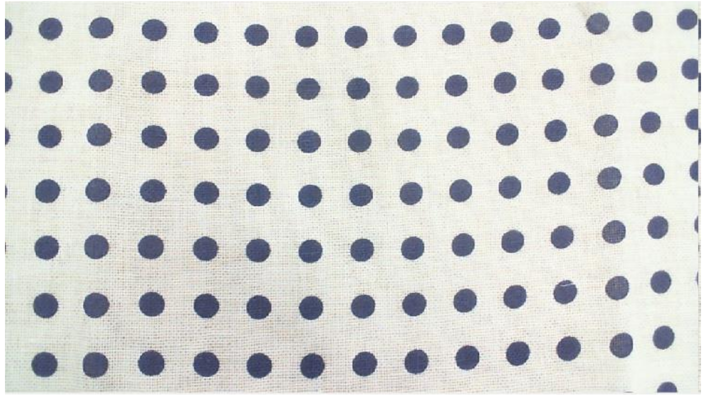
-All the value of background is zero

You can find all defect units in upper-left (better result)

Anomaly detection using an auto encoder

[Anomaly detection and localization using deep learning(CAE)]

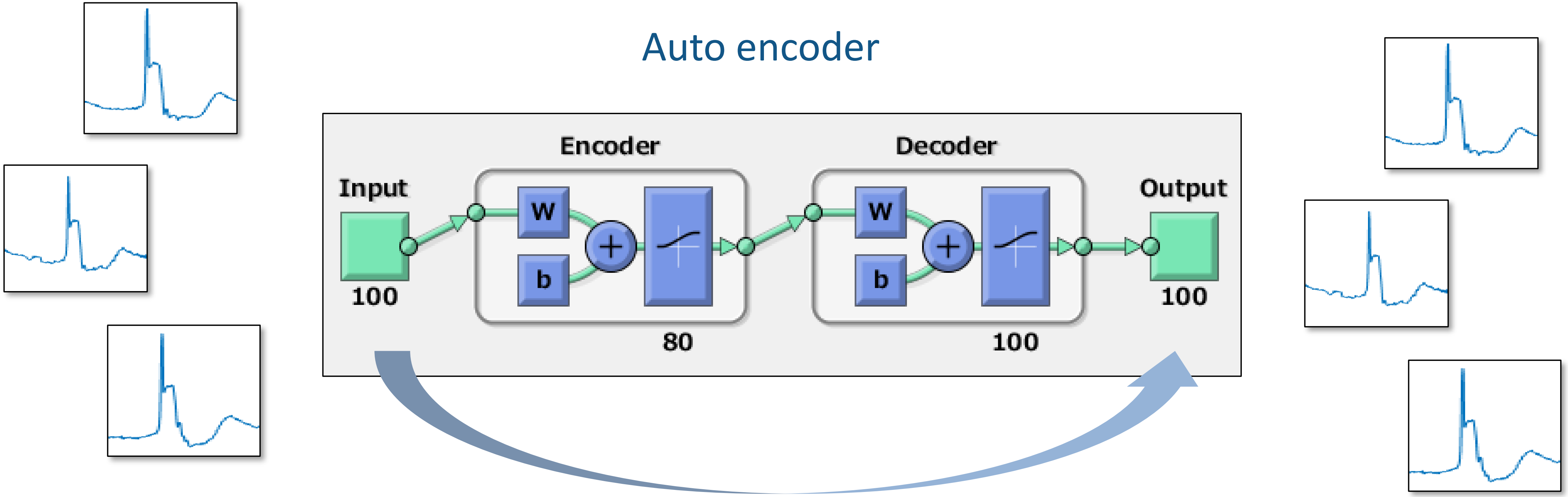
For chemical materials, wafer, clothing, and food materials, it is necessary to detect defects and impurities in normal products. However, it is a problem in automatic detection that you do not know what kind of abnormality will occur.



many kind of abnormality

In this demo, we will try to detect and localize different types of abnormalities by training DNN only with normal image data.

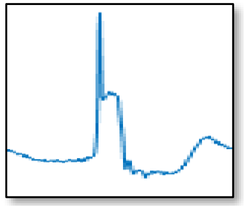
Anomaly detection using an auto encoder

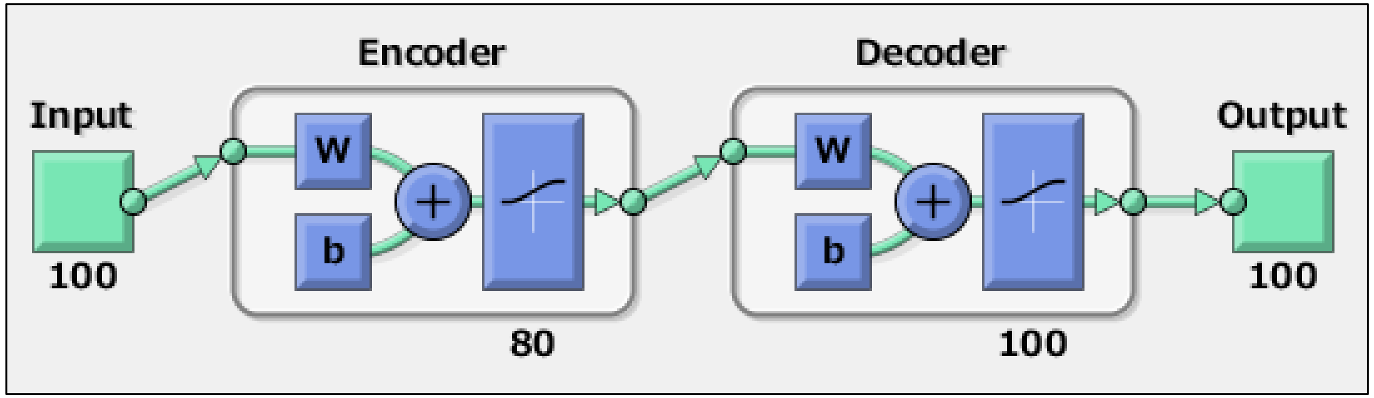


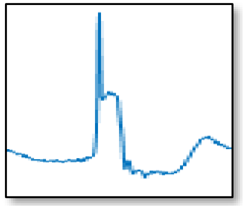
train network to play the original normal waveform

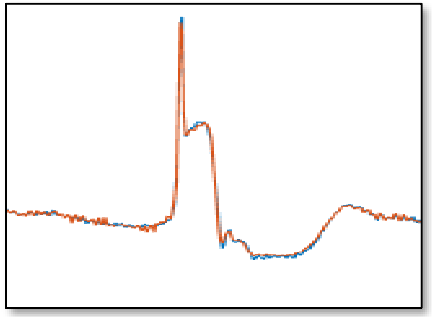
Anomaly detection using an auto encoder

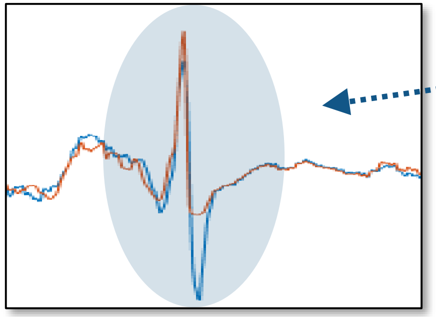
Trained network can reconstruct normal signal, but can’t do abnormal signal











Abnormal waveforms are not reconstructed correctly

Normal signal

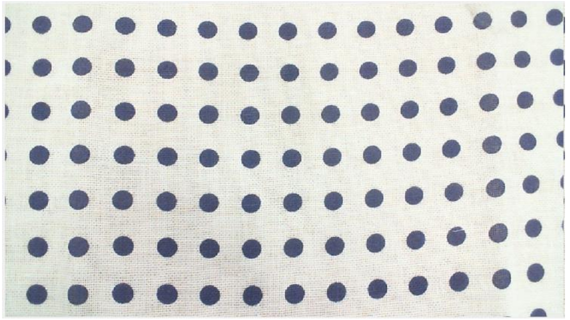
abnormal signal

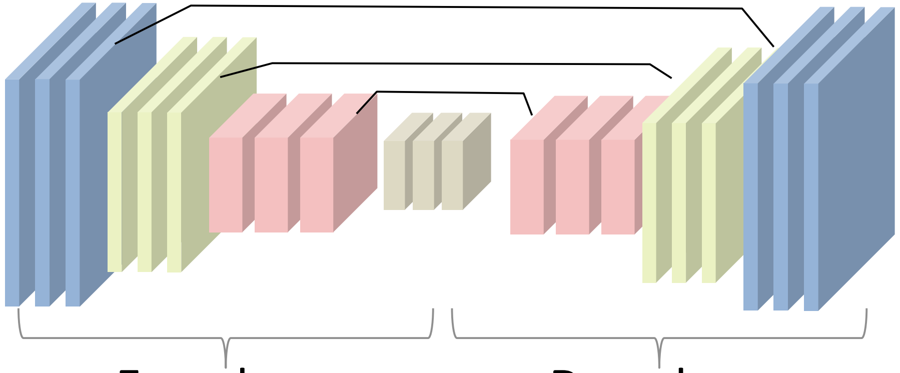
In this demo, apply this concept into image data

Anomaly detection using an auto encoder: Training

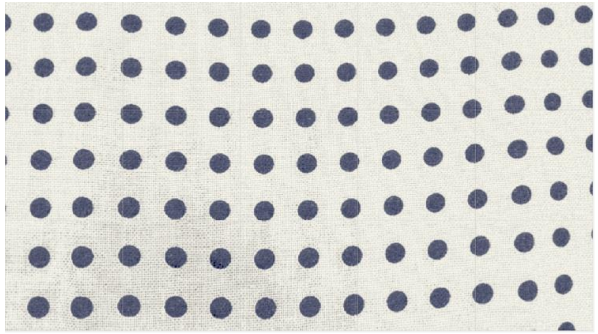
- Network has Encoder & Decoder part for Compression and decompression
- Train Network using normal images for both input and output labels

Input





Output label



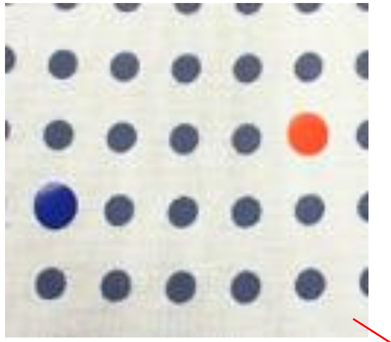
- Prepare network

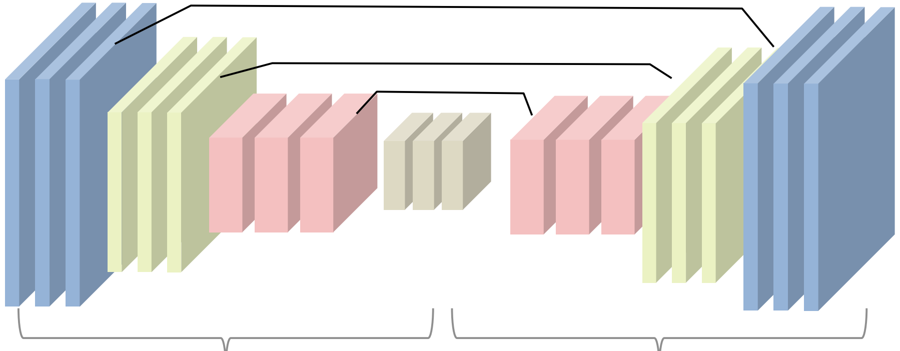
```
% Customize VGG16-based SegNet to output image
lgraph = segnetLayers(inputsize,numClasses,'vgg16');
lgraph = removeLayers(lgraph,{'softmax','pixelLabels'});
lgraph = addLayers(lgraph,regressionLayer('Name','regressionLayer'));
lgraph = connectLayers(lgraph,'decoder1_relu_1','regressionLayer');
```

Anomaly detection using an auto encoder: Inference & localization

Input an abnormal image into the trained network

Input



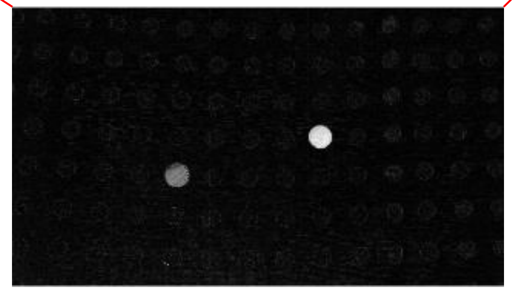


Encoder

Decoder

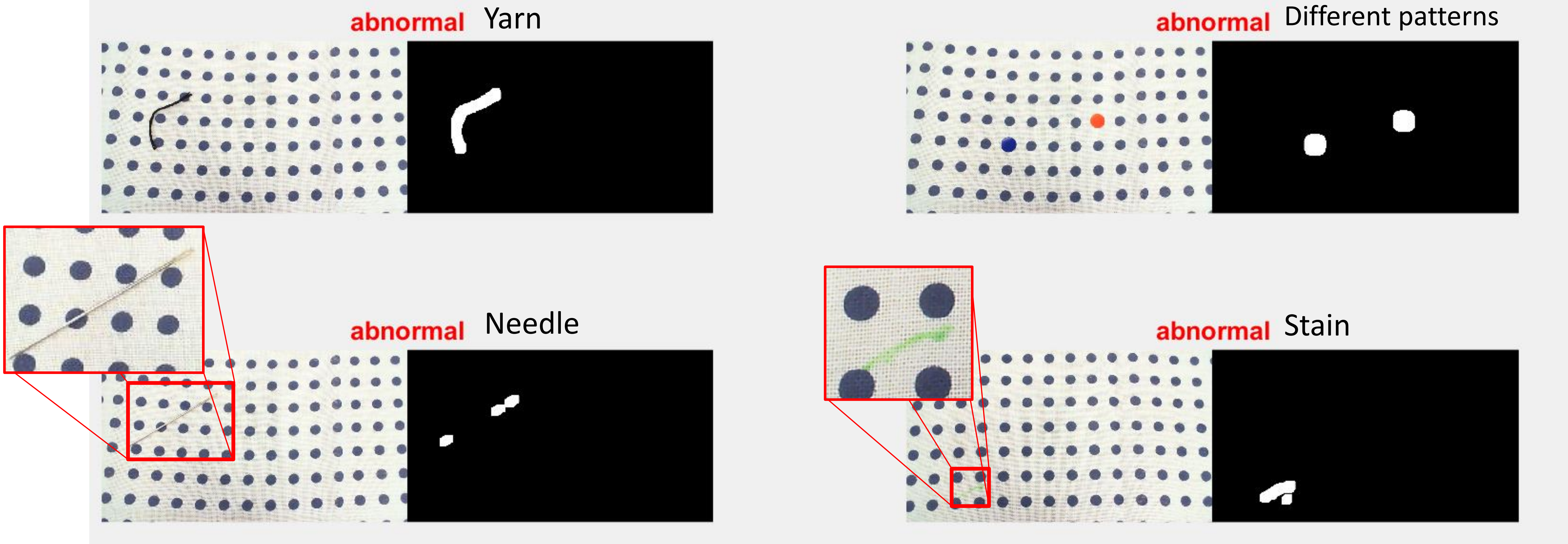
Predicted image

Calculate the difference between input and output



Concept:
Features that are not included in training data cannot be reconstructed successfully, and reconstruction errors show anomalies.

Anomaly detection using an auto encoder: Examples

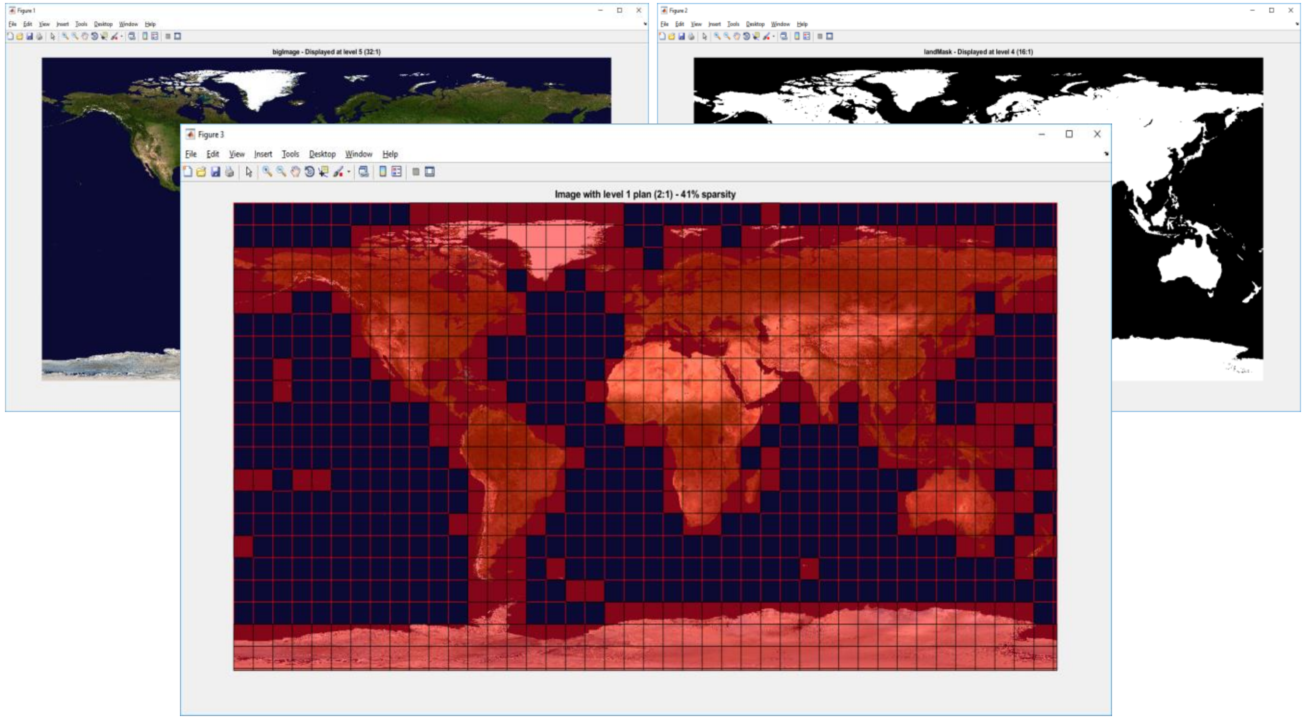


The trained network can detect and localize many kind of anomalies.

Working with Very Large Images in MATLAB

Overview

- Challenges
- bigimage object
- Visualization
- Processing
- Training for deep learning



Challenges

- Images can be too big to process all at once in RAM.
- Disk swapping is very slow.
- Chunking files on disk into smaller blocks is cumbersome.
- Working with multiresolution images is not straightforward.
- Visualizing large images can be very slow.

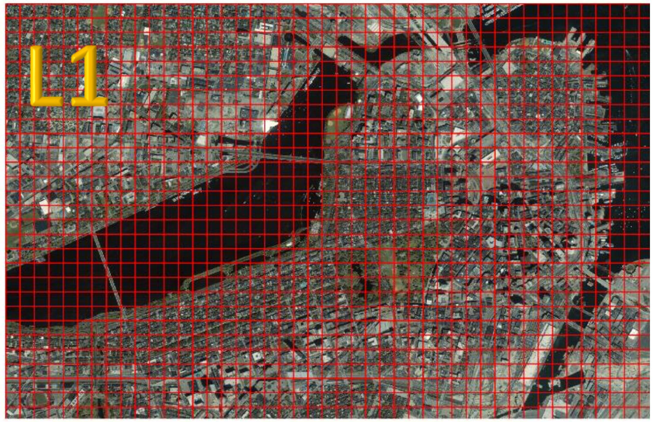
How big is "big?"

- We designed `bigimage` to handle **any** size image.
- It makes most sense for images that can't be processed in core RAM (or GPU).
- For example 10,000-by-10,000 or larger. (300 MB for RGB)
- It can handle infinitely large images.
- We routinely test with 90K-by-90K RGB TIFF images. (1.1 GB file, 24 GB memory)

(How big is your data? What file formats do you use?)

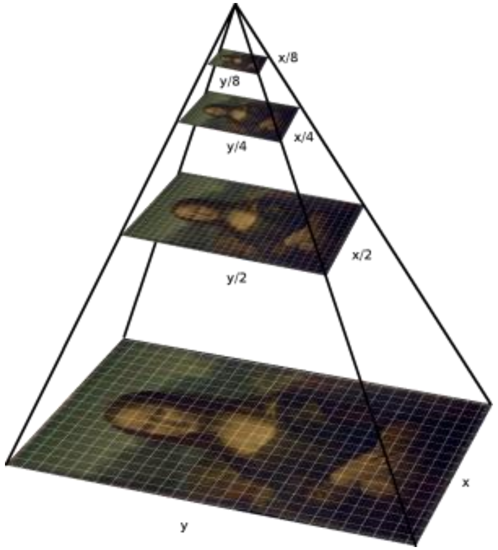
Multiresolution & tiled imagery

- One file contains multiple images of the same scene.
- Images are broken into blocks.
- Blocks cover different amounts of scene at different resolution levels. Block has same # of pixels.



Each red box is a 1024-by-1024 tile in the file.
L1's dimensions = 29,600 x 46,000
L2's dimensions = 14,800 x 23,000
L3's dimensions = 7,500 x 12,000

```
Rows = 29600
Columns = 46000
BlockSize = [1024 1024]
ResolutionLevelSizes = [ 29600 46000
                         14800 23000
                          7500 12000]
CoarsestLevel = 3
FinestLevel = 1
PixelSpacings = [1 1; 2 2; 3.947 3.833]
```



`bigimage`

- The `bigimage` class wraps around very large images.

```
>> img = bigimage('myReallyBigFile.tif');
```

- Only subregions of the image are read.
- The image pixels are not loaded all at once.
- Supports multiple resolution levels per image.

Visualization

- Use `bigimageshow` to quickly explore large images.

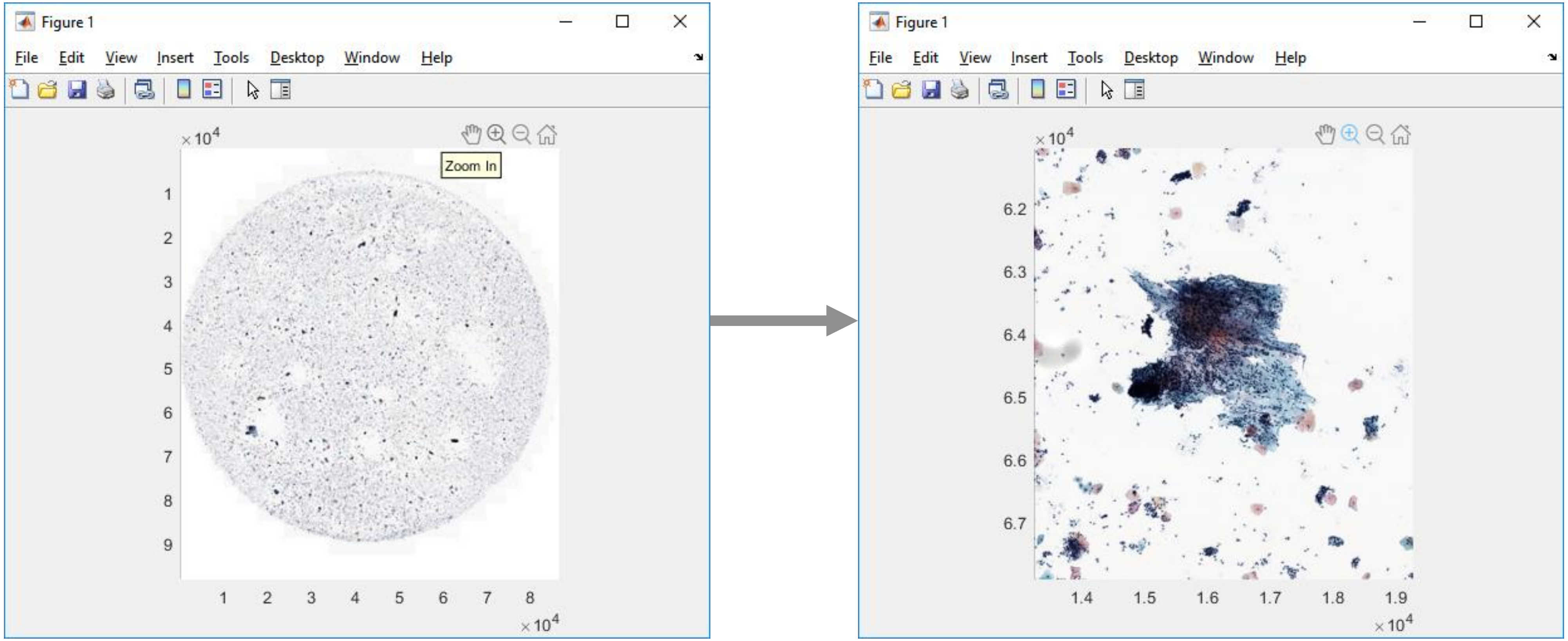


Image Processing: Key `bigimage` methods

- apply - Do image processing
- getRegion - Read an arbitrary region of the image
- getBlock - Read one specific block
- setBlock - Set the data for one block
- write - Write all of the data to a new image file (TIFF or directory)

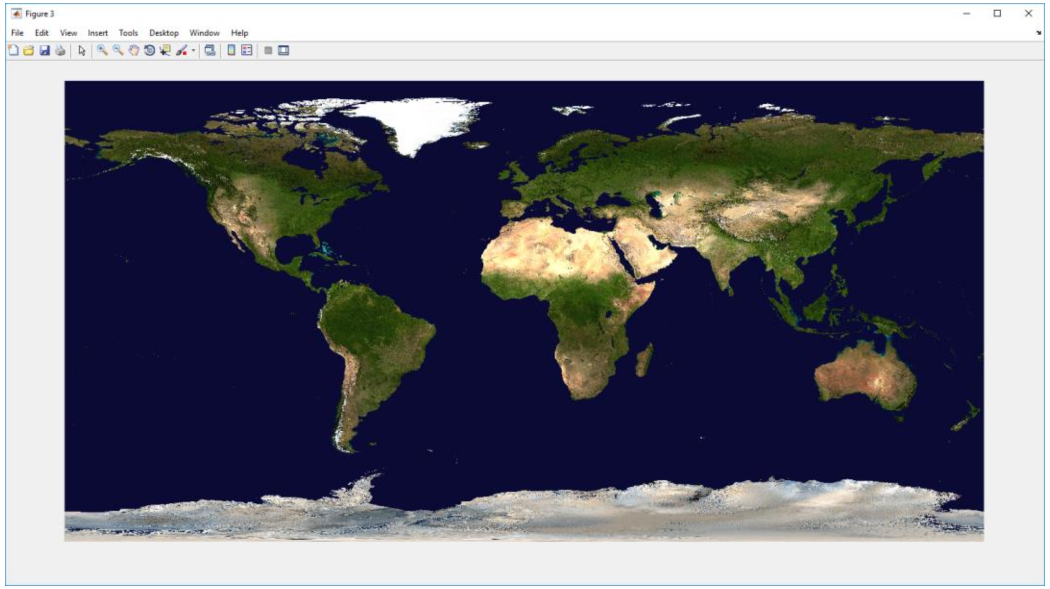
bigimage.apply()

- newObj = apply(obj, level, fcn)
- apply(obj, level, fcn, otherBigImages)
- [im1, im2, ..., imN, otherOutput] = apply(___)

Parameter	Values
UseParallel	TF
BlockSize	[rows,cols]
BorderSize	[m n] – Load *m* rows, *n* cols from neighbor blocks
PadMethod	kind
Mask	bigLogicalImage
InclusionThreshold	percent

Image processing example: Find all the sand in a satellite image

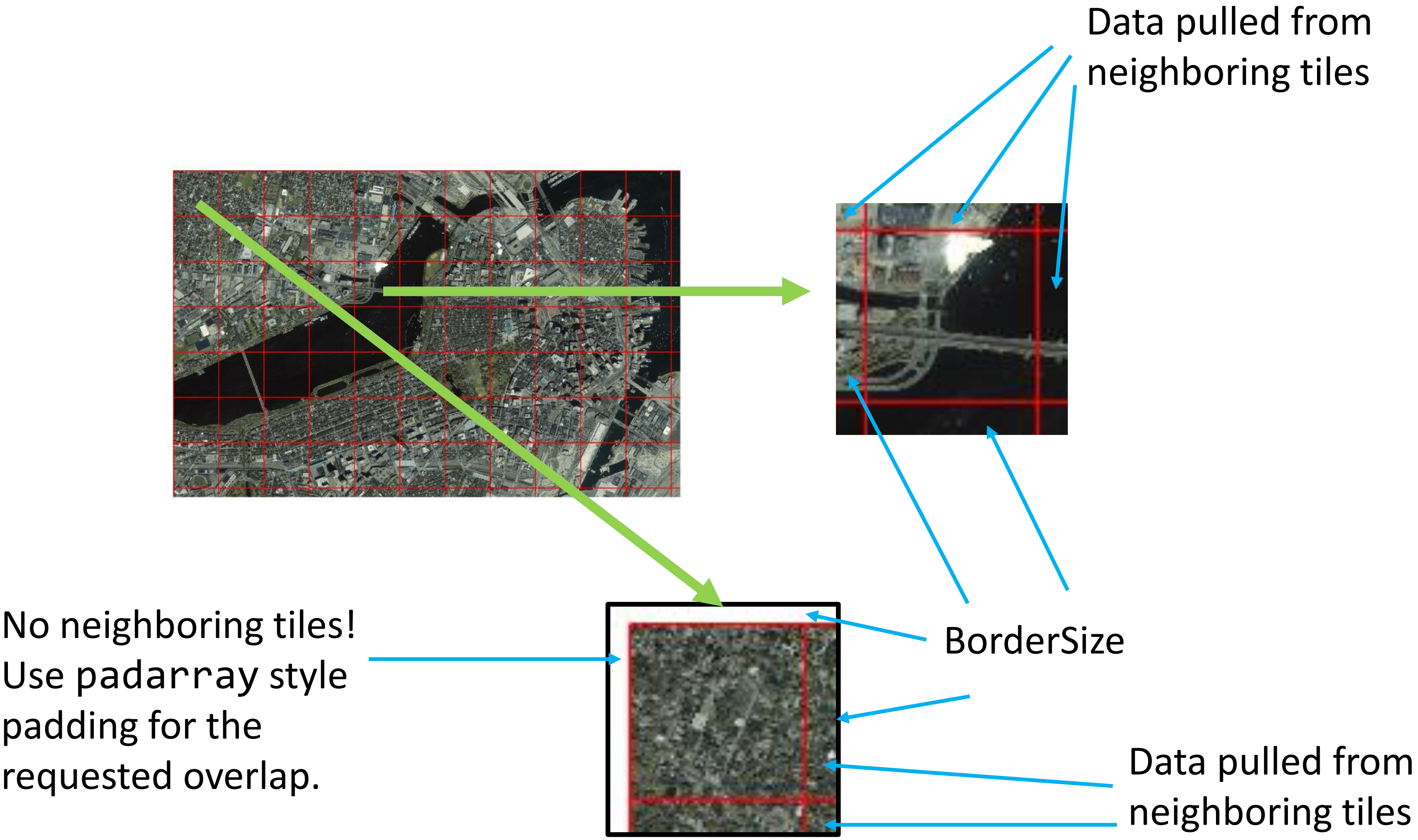
- Blue marble composite image = 43k x 22k RGB
- Too big to process efficiently in MATLAB.
- Use multi-resolution (pyramid) processing.



- Find the majority sandy regions based on color.

1. Instantiate a class that can load parts of the big file on-demand.
2. Examine whole image to find land regions using very low-res tiles.
3. Look at full-resolution tiles only of land region to classify pixels as sand.

bigimage: OverlapAmount, padding, and neighboring tiles



Training for deep learning

- Training a convolutional neural network (CNN) requires a lot of data.
- Very large images can be subsampled to increase training data.
- Use `bigimageDatastore` to get all blocks from a `bigimage` object.
- Pass this datastore to `trainNetwork`.

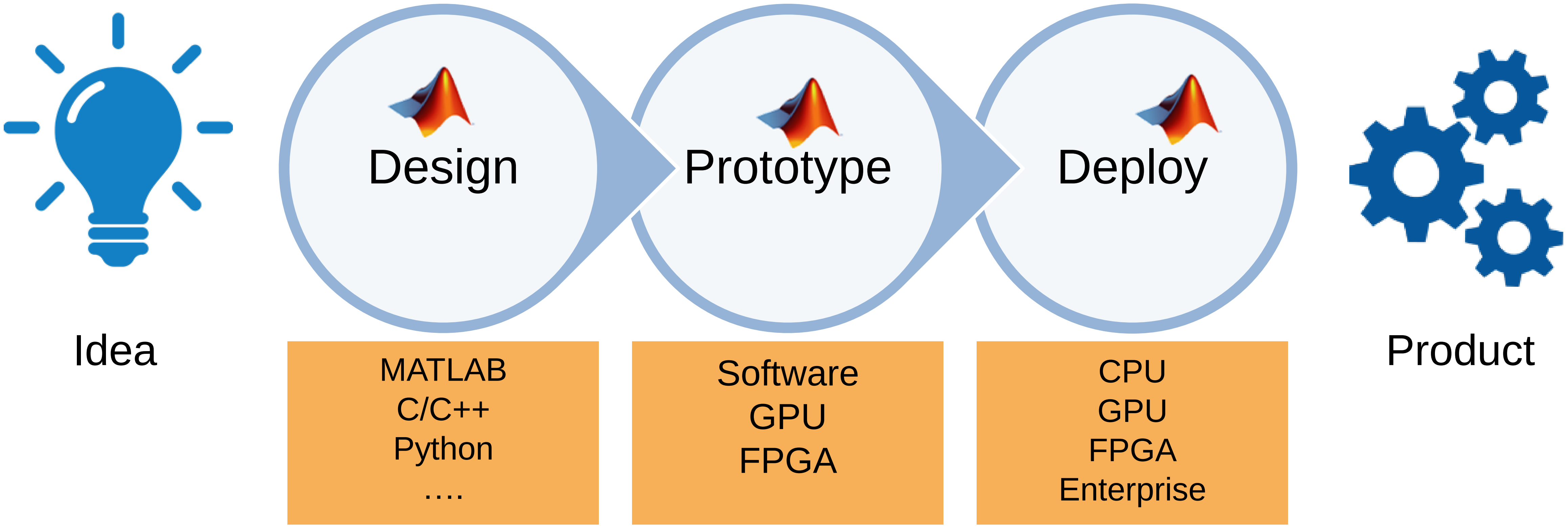
`bigimageDatastore`

`bigimageDatastore`

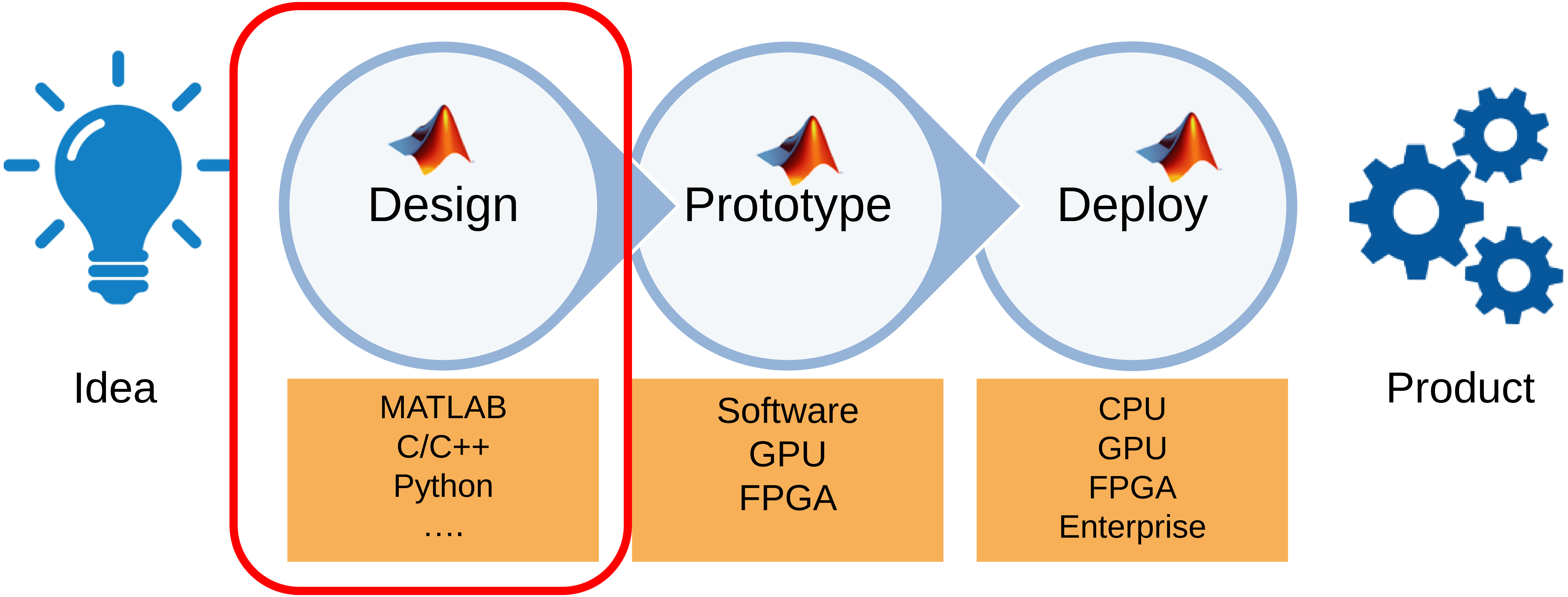
- Constructed with 1 or more `bigimage` objects
- `[blocks, info] = read(ds)` serves minibatches of blocks
- `readRelative(ds, info, direction)` gives a neighbor
- `shuffle`, `partition`, `combine`, `transform`, …

Deploying Image Processing Applications with MATLAB

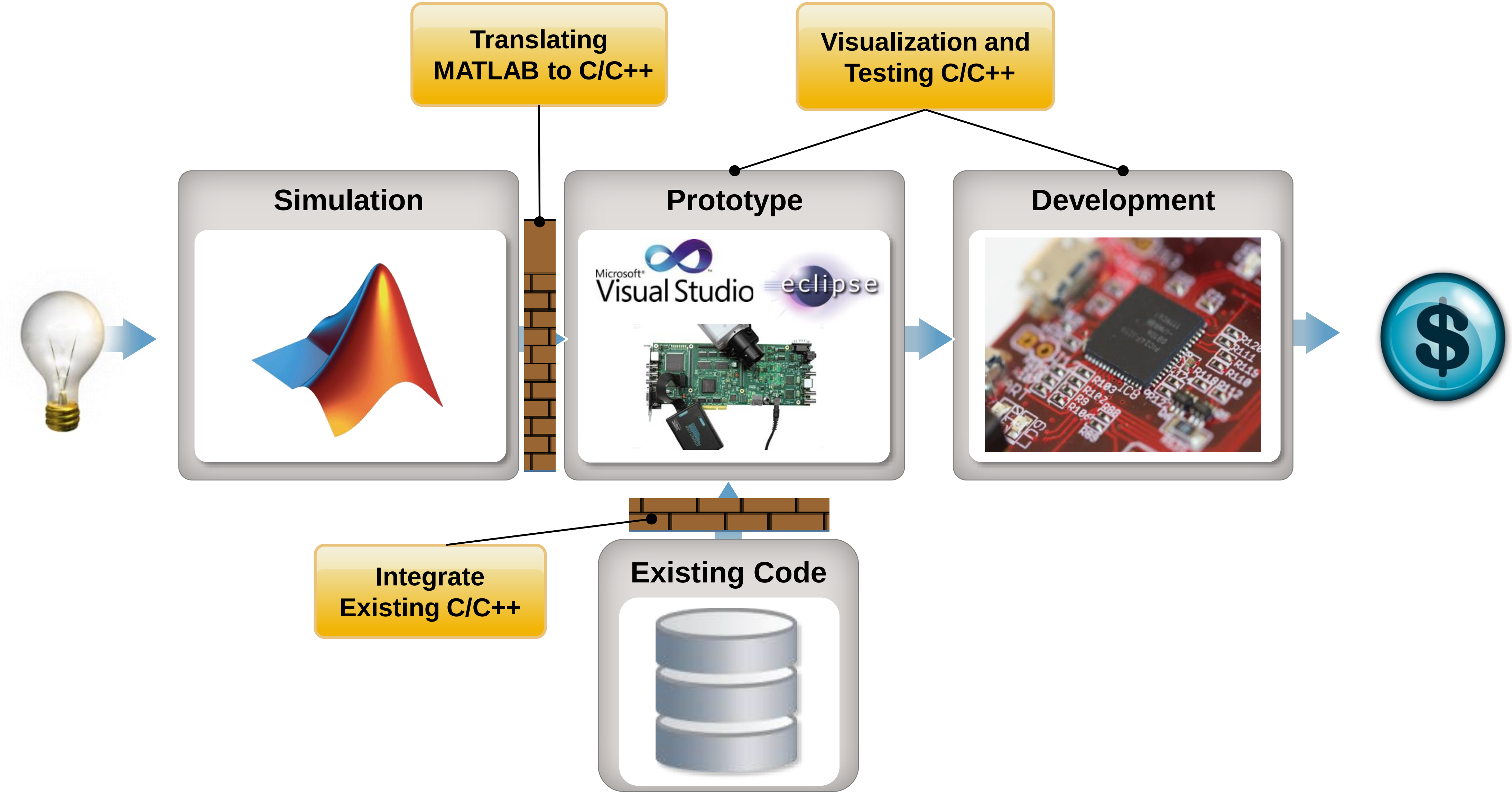
Image Processing Deployment Workflow



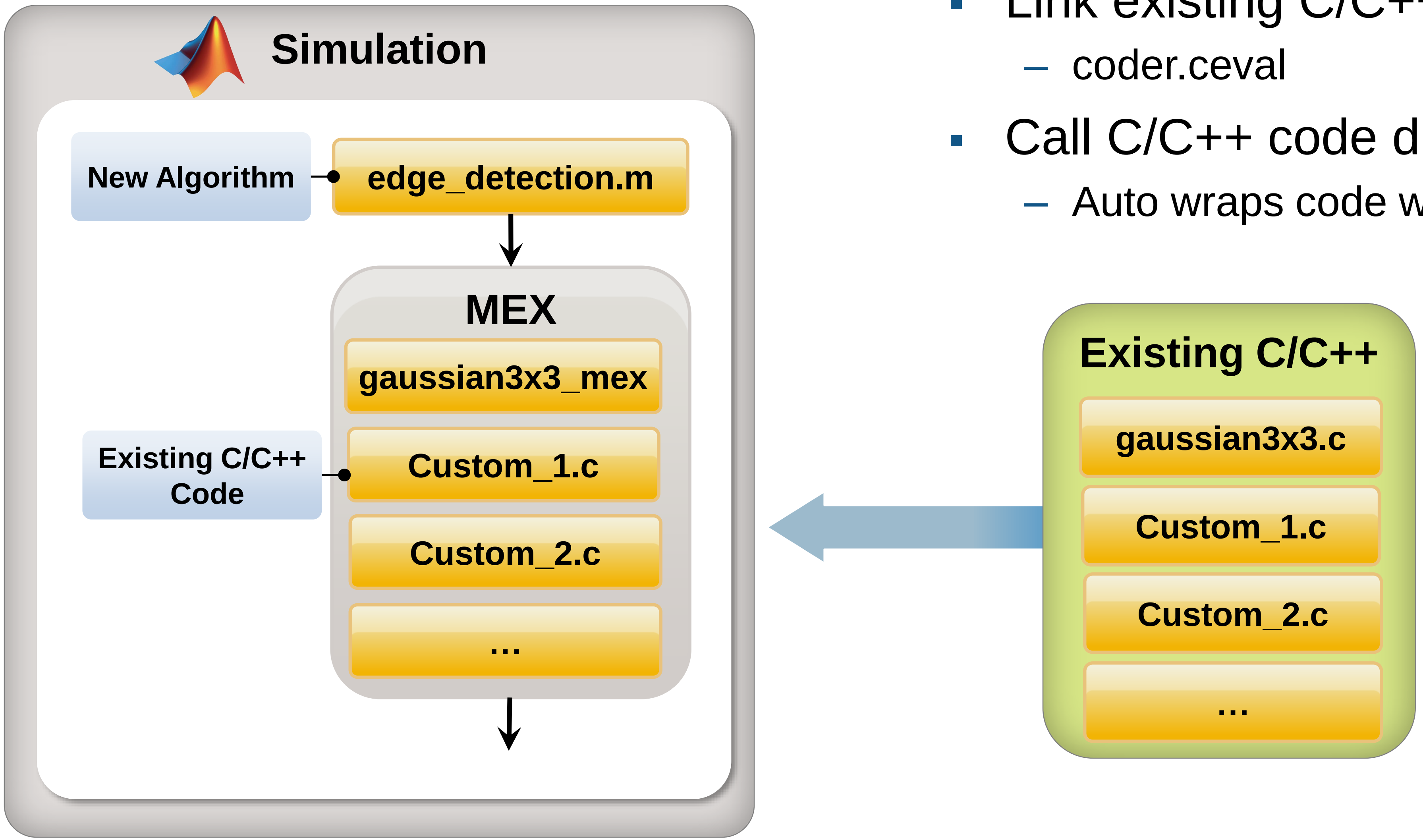
Designing an Image Processing Application



Typical Development Workflow



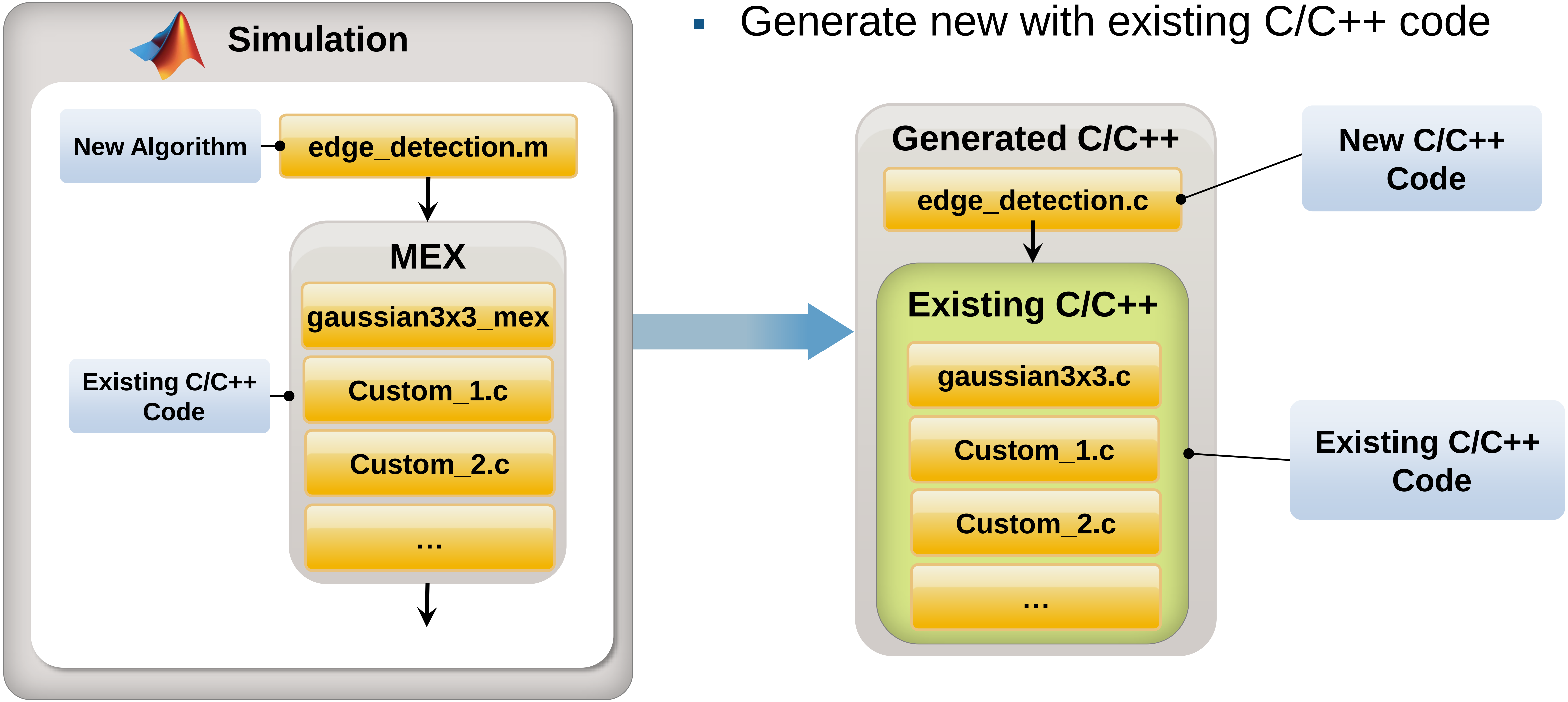
Integrating Existing Code in for Simulation



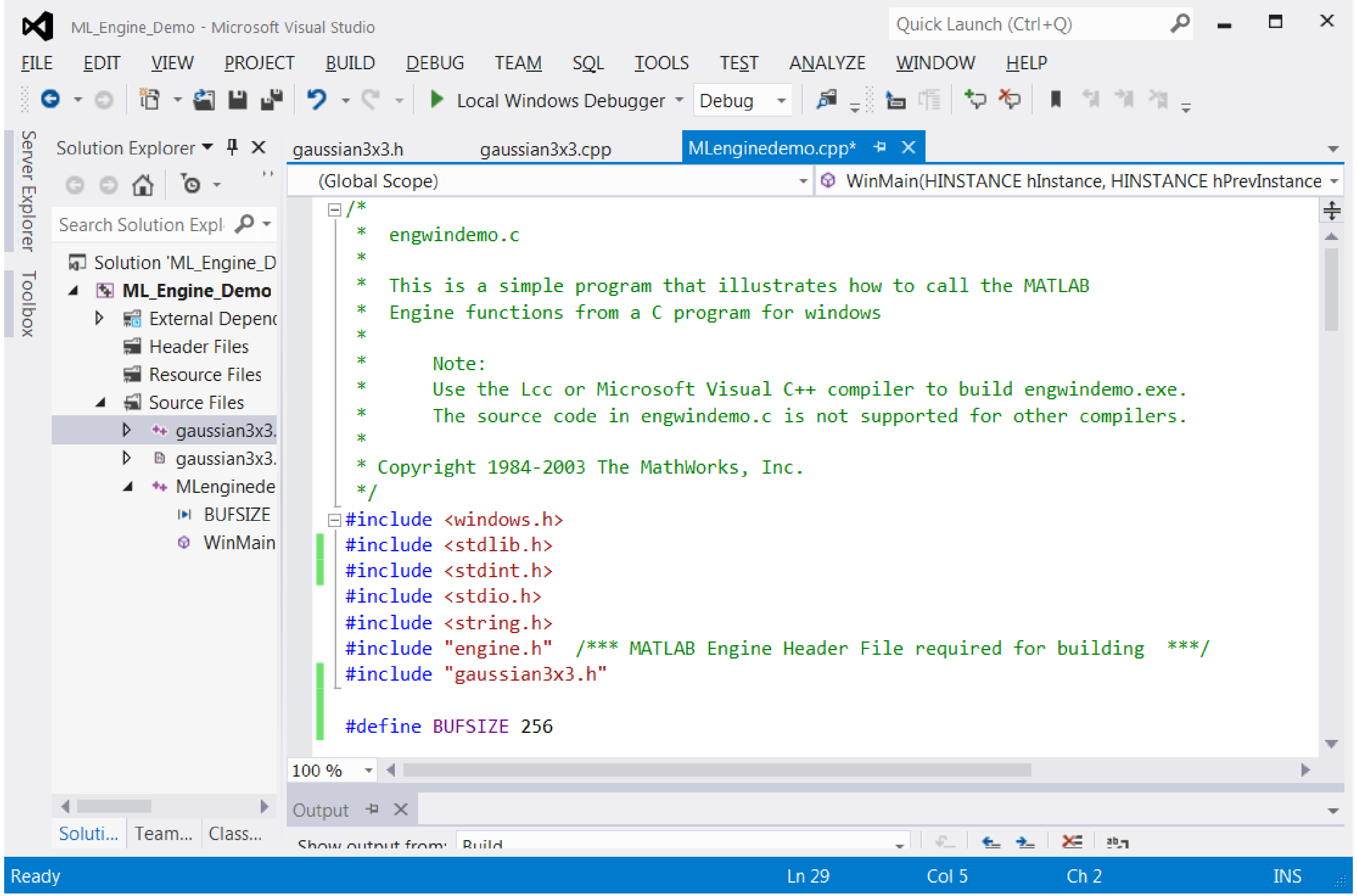
- Link existing C/C++ code to MATLAB
  - coder.ceval
- Call C/C++ code directly in MATLAB
  - Auto wraps code with MEX

Generate Code with Existing C/C++ Code for Prototyping

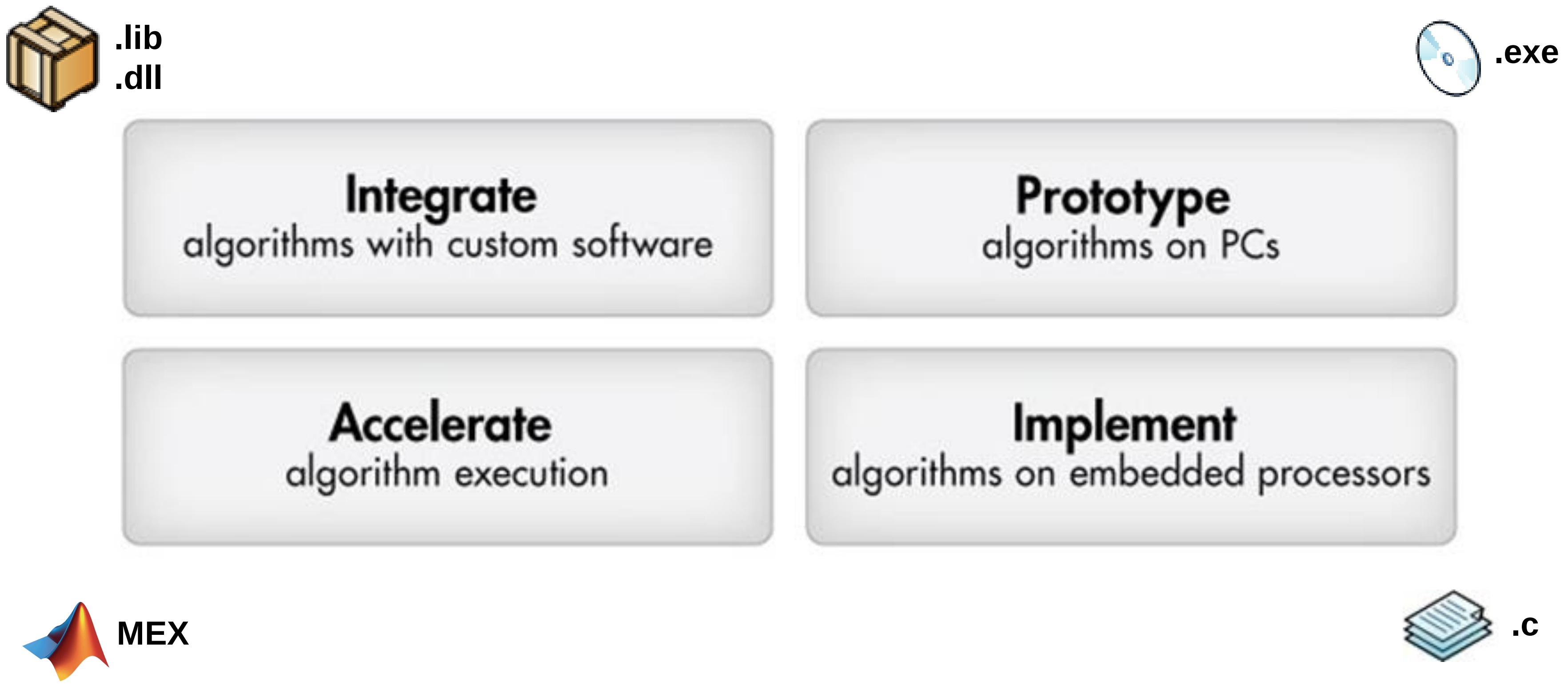
- Generate new with existing C/C++ code



Visualizing, Prototyping, and Testing C/C++ from Visual Studio or Eclipse



MATLAB Coder Use Cases



Prototyping an Image Processing Application

Idea

Design

MATLAB
C/C++
Python
….

Prototype

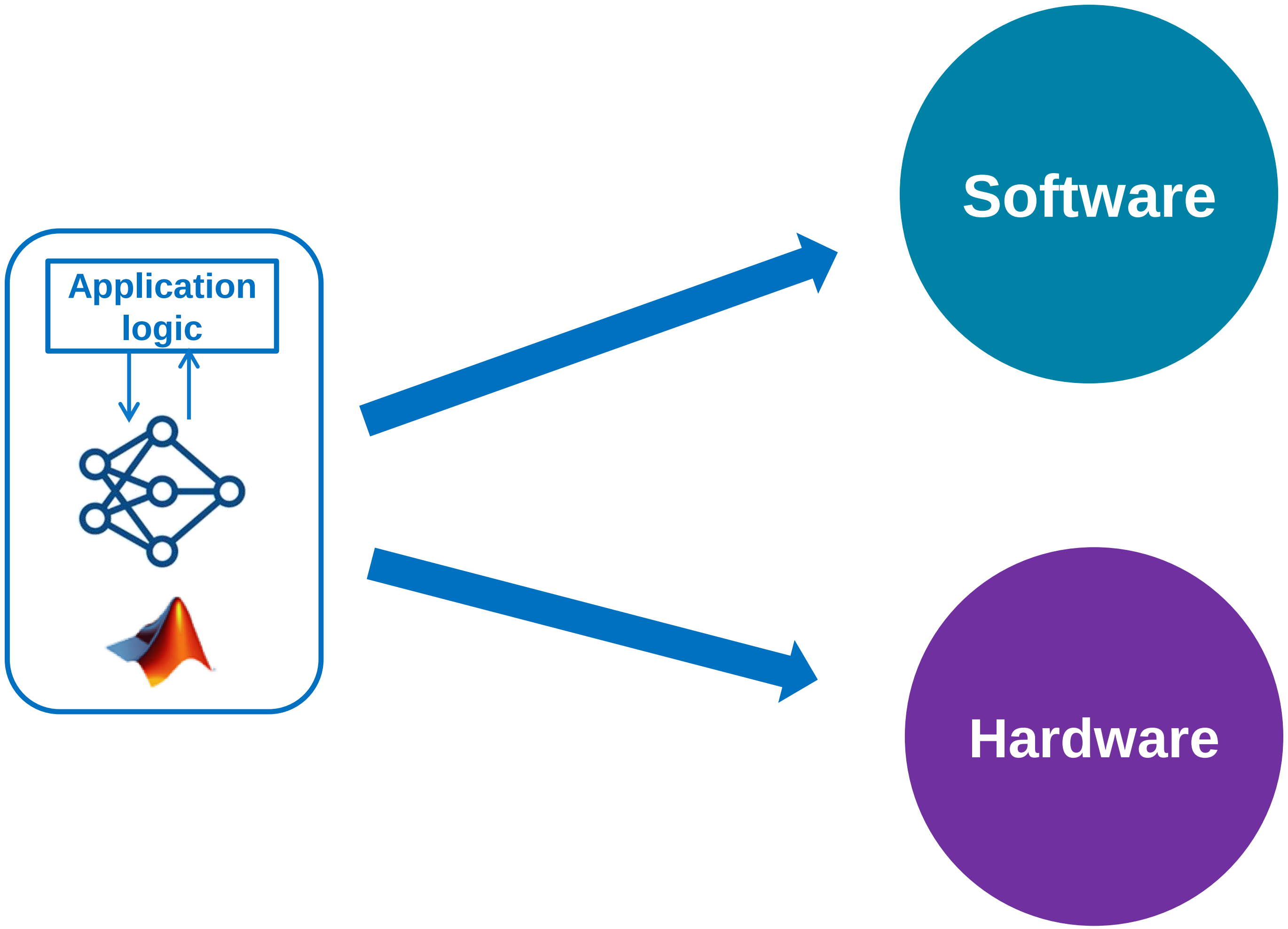
Software
GPU
FPGA

Deploy

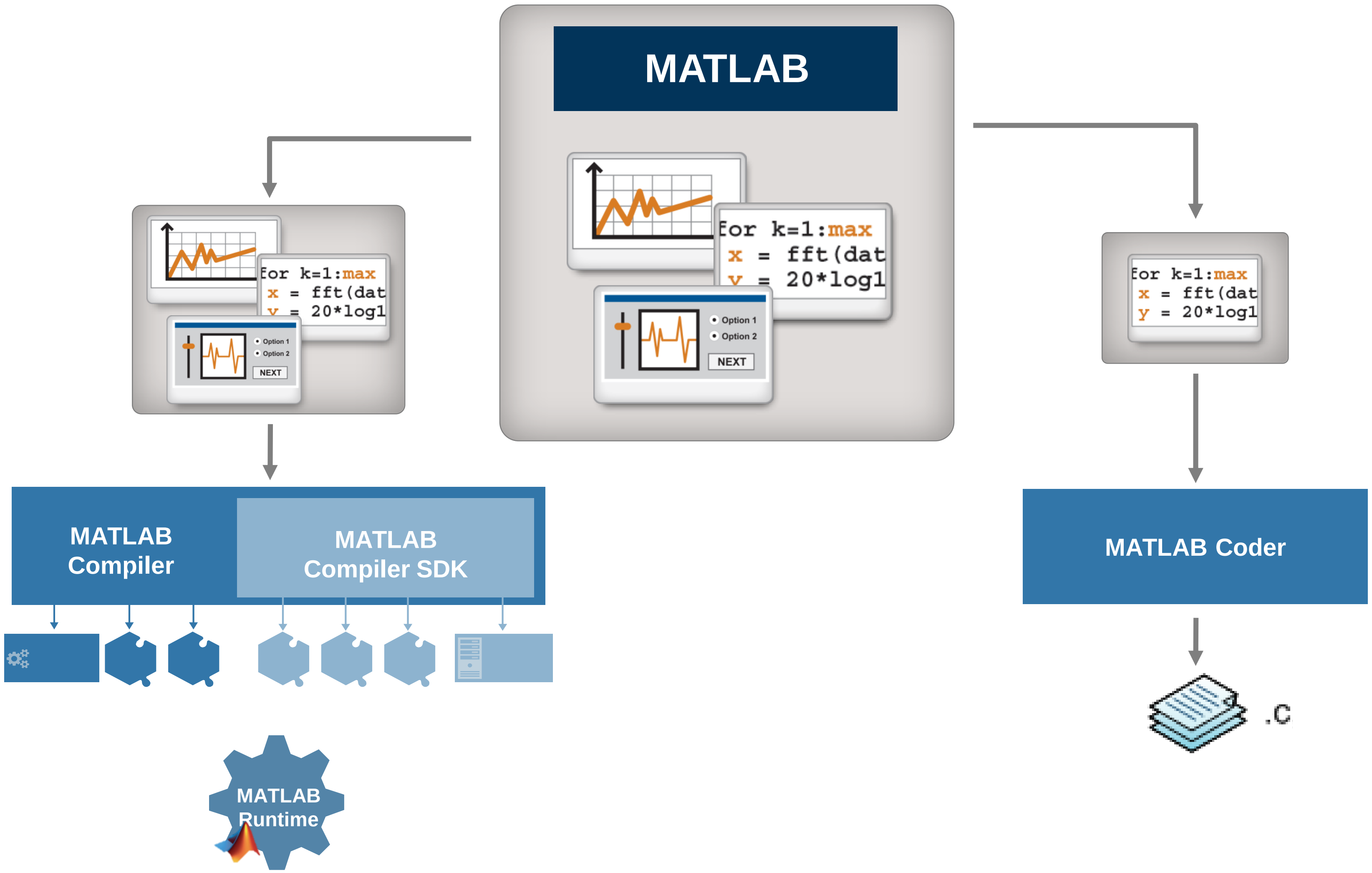
CPU
GPU
FPGA
Enterprise

Product

Prototype in Software or Hardware



Desktop Software Prototyping and Deployment



Hardware Support Packages

MathWorks® Products Solutions Academia Support Community Events

Hardware Support

Search Hardware Support | Hardware Support

Overview | Search Hardware Support | Request Hardware Support

Trial software | Contact sales

Refine by Product

MATLAB	1
Simulink	1
MATLAB Coder	1

Refine by Product Family and Category

MATLAB Product Family	1

Refine by Vendor

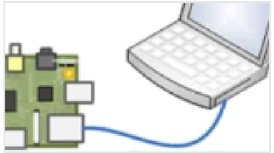
Raspberry Pi ×

Refine by Application

Control Systems	1
Digital Signal Processing	3
Embedded Systems	2
Image Processing and Computer Vision	×
Internet of Things	2
Power Electronics Control Design	1
Test and Measurement	1

FILTERED BY Image Processing and Com… × Raspberry Pi × Remove All ×

Results 1 - 3 of 3



Raspberry Pi Support from MATLAB

Acquire sensor and image data from your connected Raspberry Pi

Vendors: Raspberry Pi

Tags: Support Package Installer Enabled, Project-Based Learning, MathWorks Supported

Raspberry Pi Support from MATLAB Coder

Generate C code and deploy it as a standalone application on a Raspberry Pi

Vendors: Raspberry Pi

Tags: C/C++ Code Generation

Raspberry Pi Support from Simulink

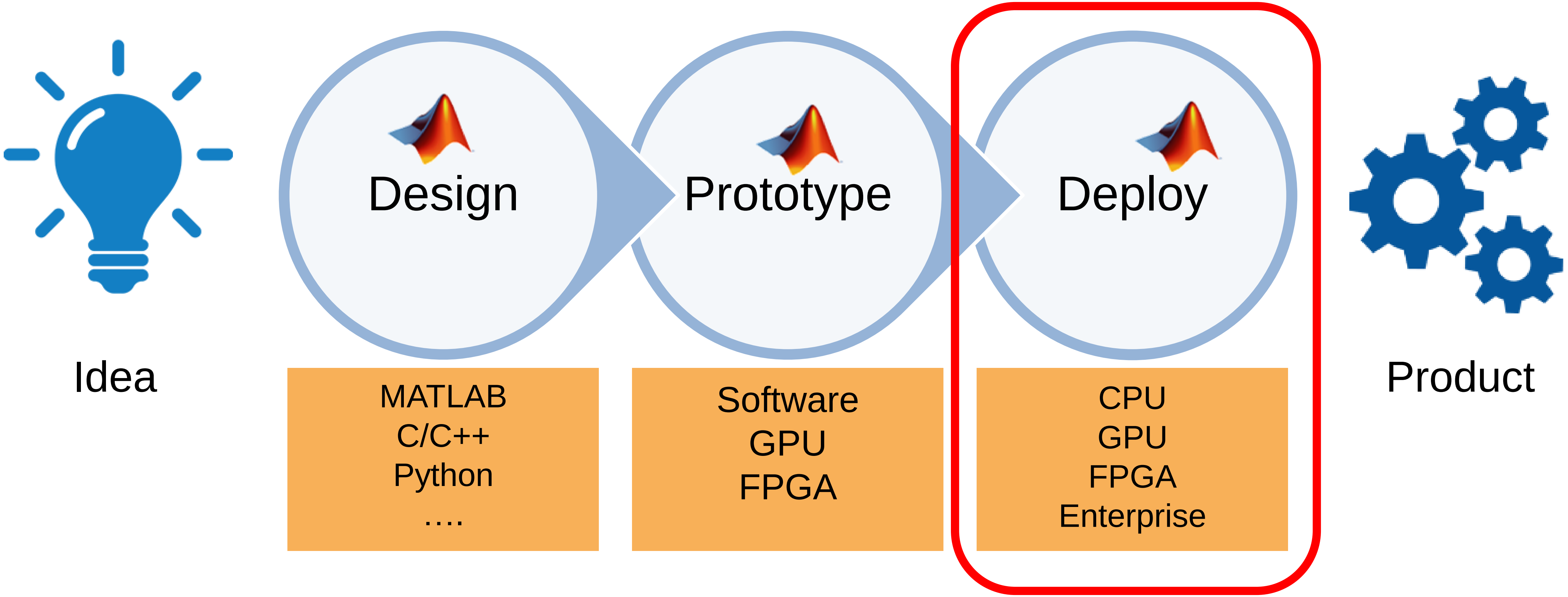
Run models on Raspberry Pi

Vendors: Raspberry Pi

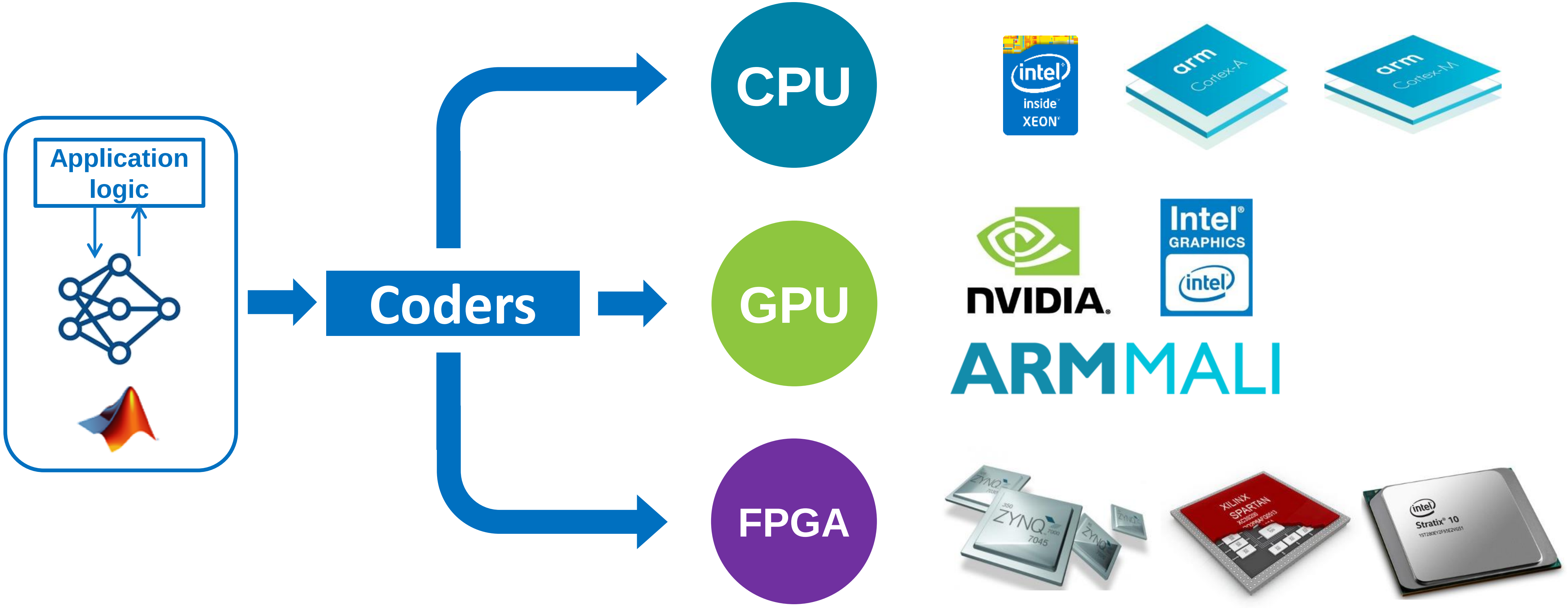
Tags: Support Package Installer Enabled, Run on Target Hardware, C/C++ Code Generation, Project-Based Learning, MathWorks Supported

Results 1 - 3 of 3

Deploying an Image Processing Application



Deploy to <u>any</u> processor with best-in-class performance



GPU Coder for Deployment

MATLAB

GPU Coder

Accelerated implementation of parallel algorithms on GPUs & CPUs

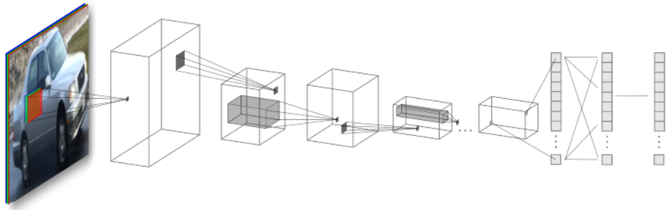
Intel MKL-DNN Library

NVIDIA CUDA C/C++

ARM Compute Library

Deep Neural Networks

Deep Learning, machine learning



5x faster than TensorFlow
2x faster than MXNet

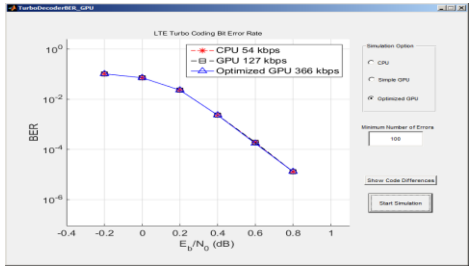
Image Processing and Computer Vision

Image filtering, feature detection/extraction

60x faster than CPUs for stereo disparity

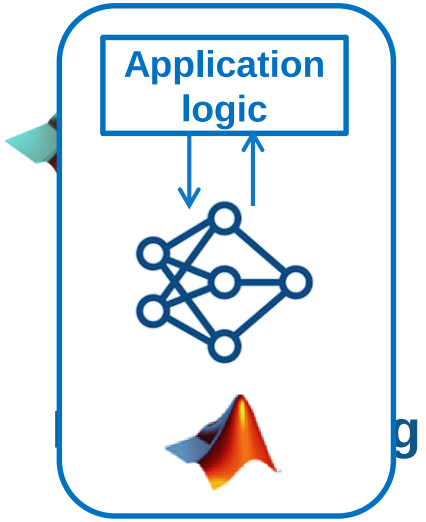
Signal Processing and Communications

FFT, filtering, cross correlation,



20x faster than CPUs for FFTs

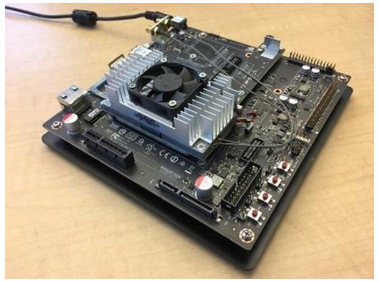
Deploying to CPUs



MATLAB Coder

Intel MKL-DNN Library

GPU Coder



NVIDIA TensorRT & cuDNN Library

MATLAB Coder

ARM Compute Library

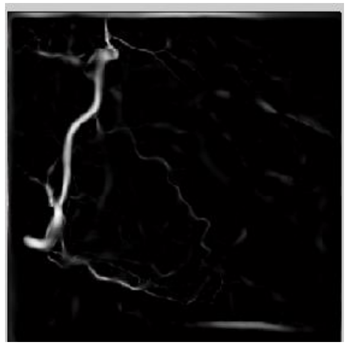
GPU Coder for Image Processing and Computer Vision

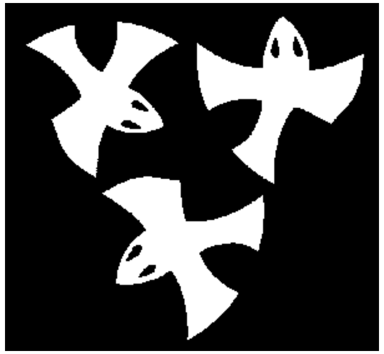
Fog removal

5x speedup

Frangi filter

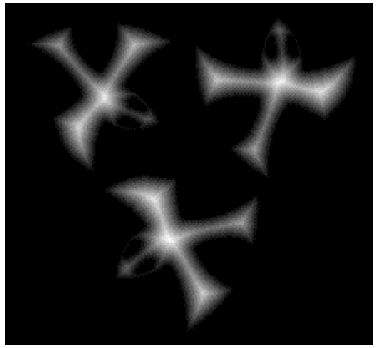
3x speedup





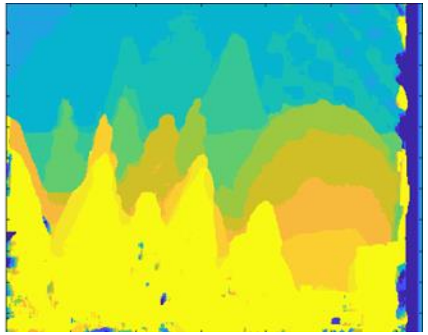
Distance transform

8x speedup



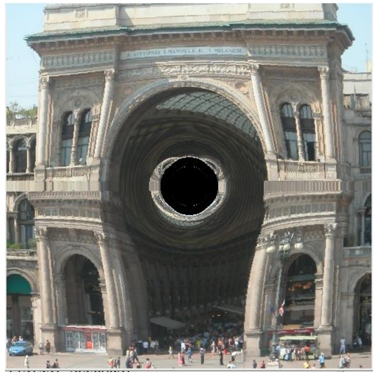
Stereo disparity

50x speedup



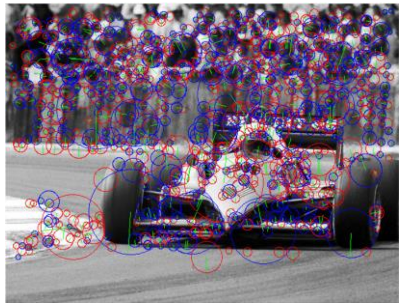
Ray tracing

18x speedup

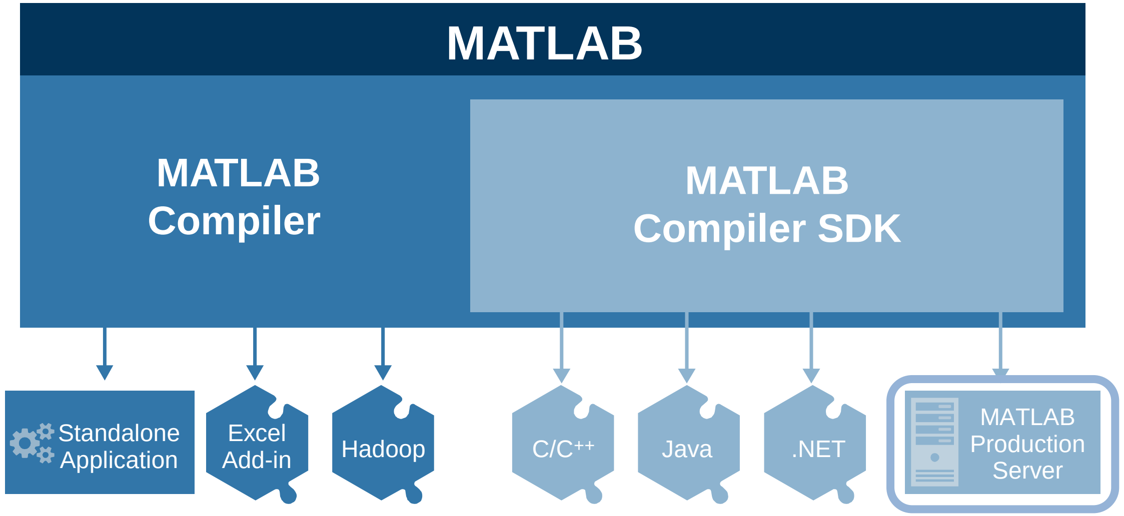


SURF feature extraction

700x speedup



Other Deployment Options



MATLAB Compiler for sharing MATLAB programs without integration programming

MATLAB Compiler SDK provides implementation and platform flexibility for software developers

MATLAB Production Server provides the most efficient development path for secure and scalable web and enterprise applications

THANK YOU!!