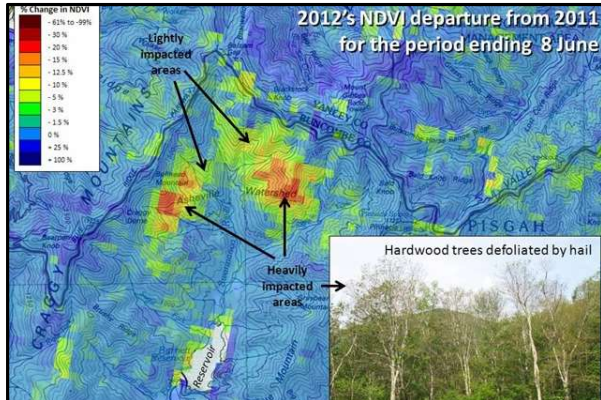


Presentation Title

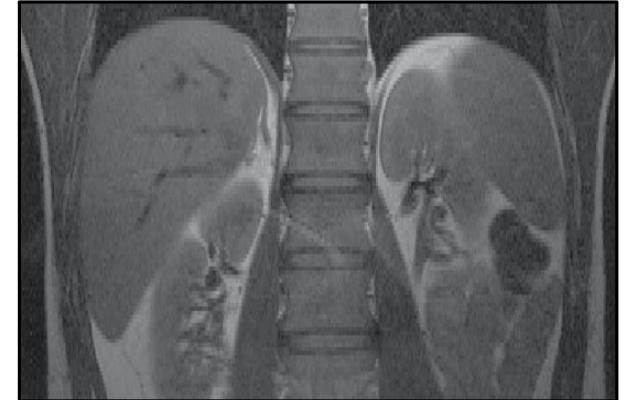
By Author

Date



NASA Develops Warning System for Detecting Forest Disturbances

Image Processing and Computer Vision Applications



Beth Israel Medical Center Improves MRI Accuracy



CNH Develops Intelligent Filling System for Forage Harvesters

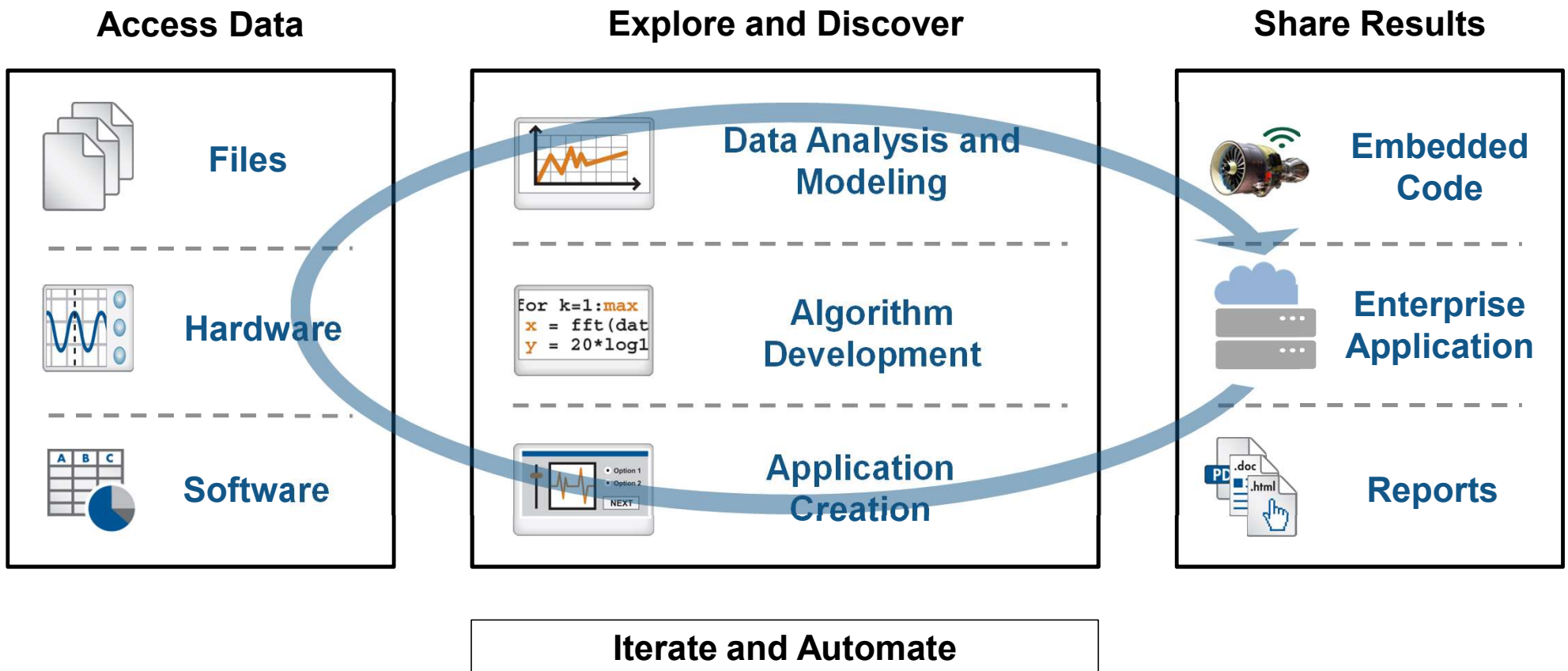


FLIR Accelerates Development of Thermal Imaging FPGA



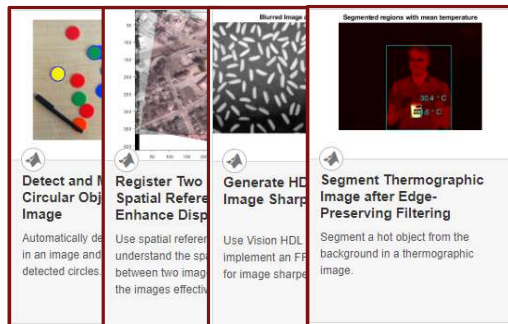
Veoneer (Autoliv) Builds Radar Sensor using LiDAR-Based Verification

Image Processing Workflow



Why Use MATLAB®?

Ease of Use and Thorough Documentation



(...)

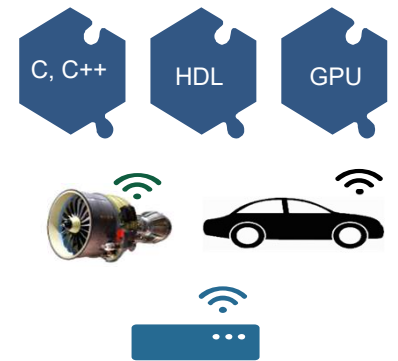
Rapid Prototyping and Algorithm Development



Code Generation for Embedded Deployment

MATLAB Code

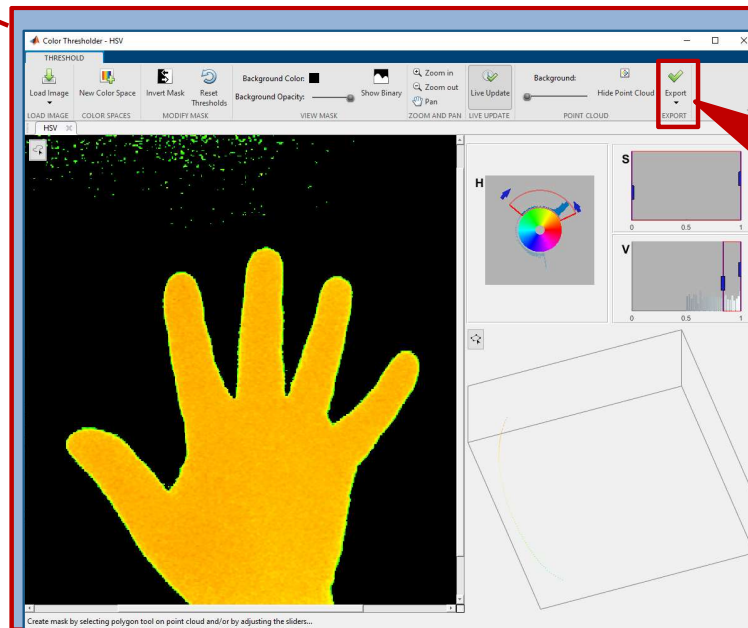
Embedded Hardware



Need Technical Help?

- Technical Support
- Application Engineers

Apps Accelerate Workflows



Color Thresholder App

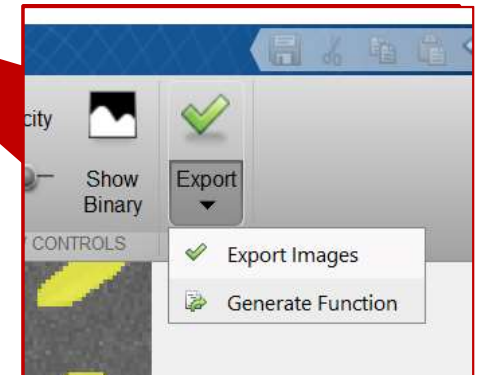


Image Processing Toolbox™ & Computer Vision Toolbox™

Import, Display, and
Exploration

Geometric Transform and
Image Registration

Image Filtering and
Enhancement

Image Segmentation and
Analysis

3D Volumetric Processing

Camera Calibration and
3D-Vision

Tracking and Motion
Estimation

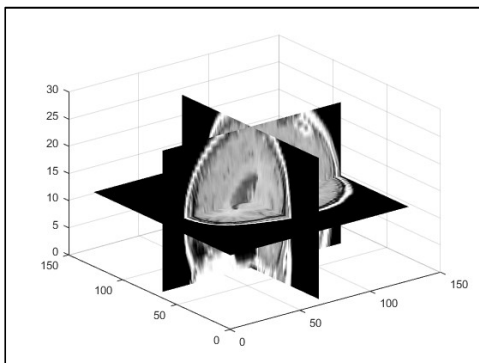
Feature Detection and
Extraction

LiDAR and Point Cloud
Processing

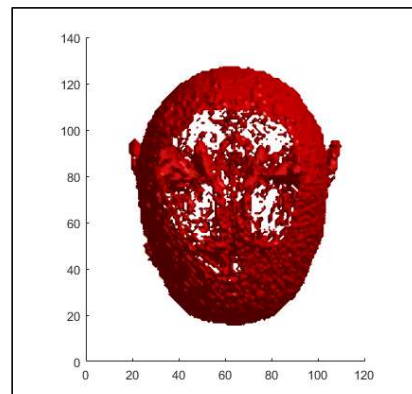
Deep Learning

3-D Image Processing Functions and Apps

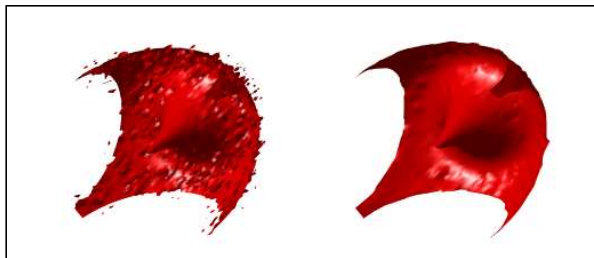
Over 65 functions supported for 3-D volumetric data



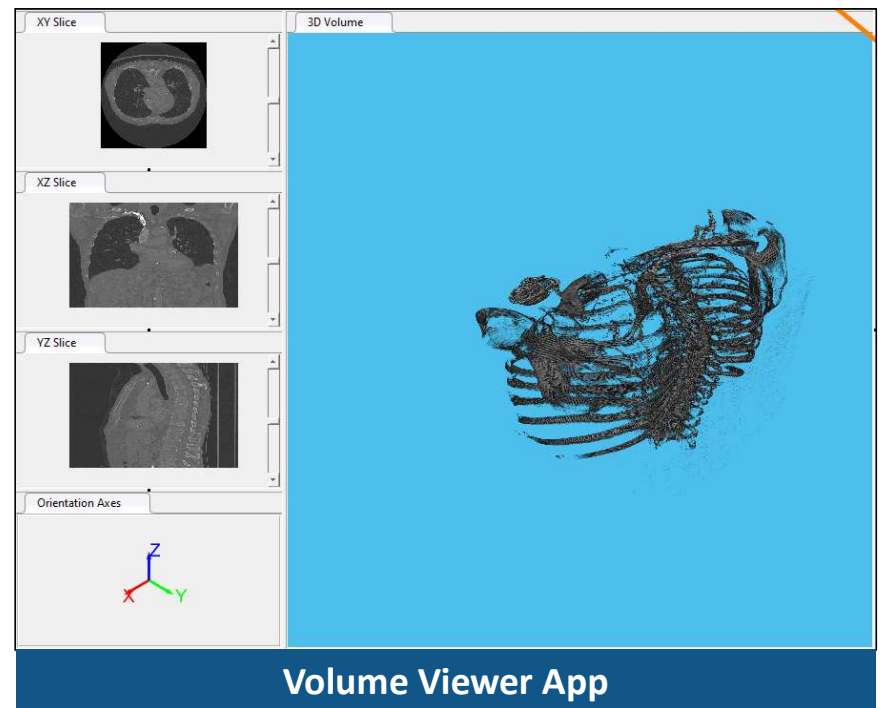
imresize3, imrotate3



activecontour

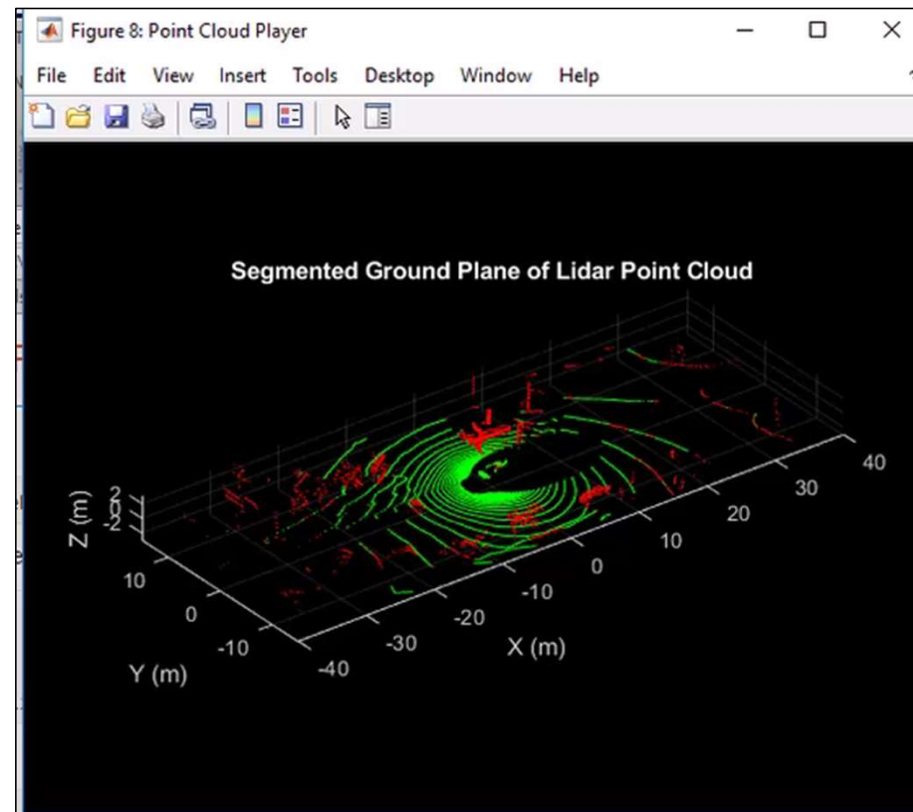


medfilt3



[Example: Segment Lungs from 3-D Chest Scan](#)

Lidar and Point Cloud Processing



[Example: Track Vehicles Using Lidar: From Point Cloud to Track List](#)

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](#)

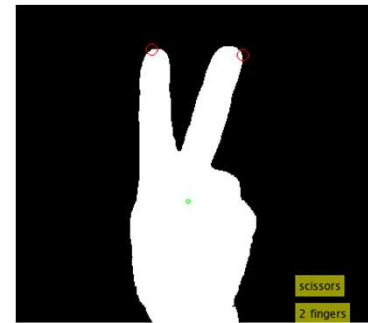
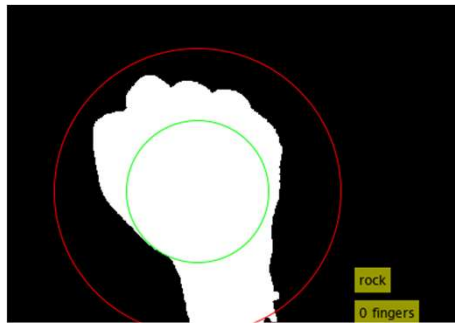
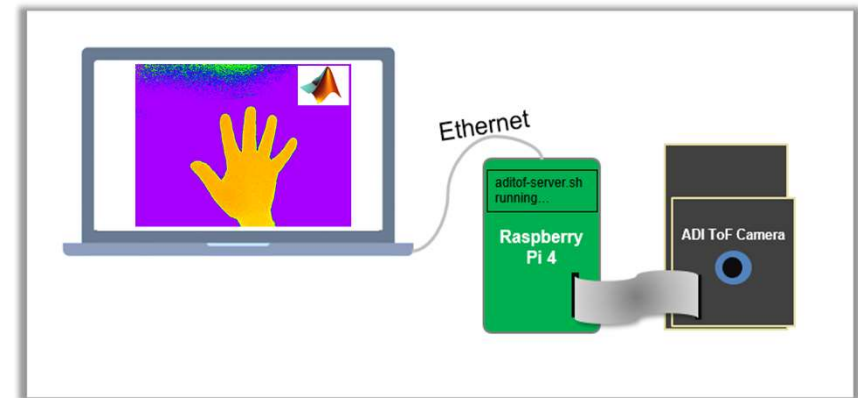
Installation

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.

MATLAB® with Analog Devices® 3D ToF Camera

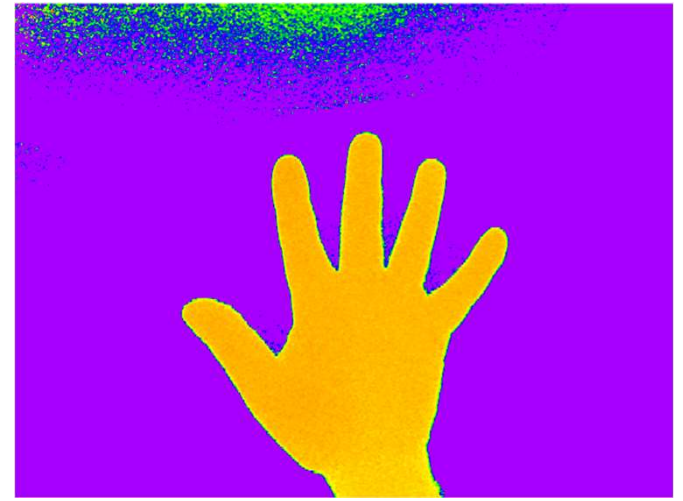
- ToF Camera makes it easy to find the hand
- MATLAB is used to figure out hand signal
- **Demo:** Rock, Paper, Scissors



NOTE: Additional toolboxes required are Image Processing, Computer Vision, and Image Acquisition

Step 1 – Detect the hand

- Grab image from camera
`depthMap = getsnapshot(depthVid);`
- Convert to HSV color space, threshold, and clean up edges
`hsv = rgb2hsv(depthMap);`
`BW = hsv(:, :, 1) <= 0.25;`
`BW = imopen(BW, strel('disk', 3));`
`BW = imfill(BW, 'holes');`



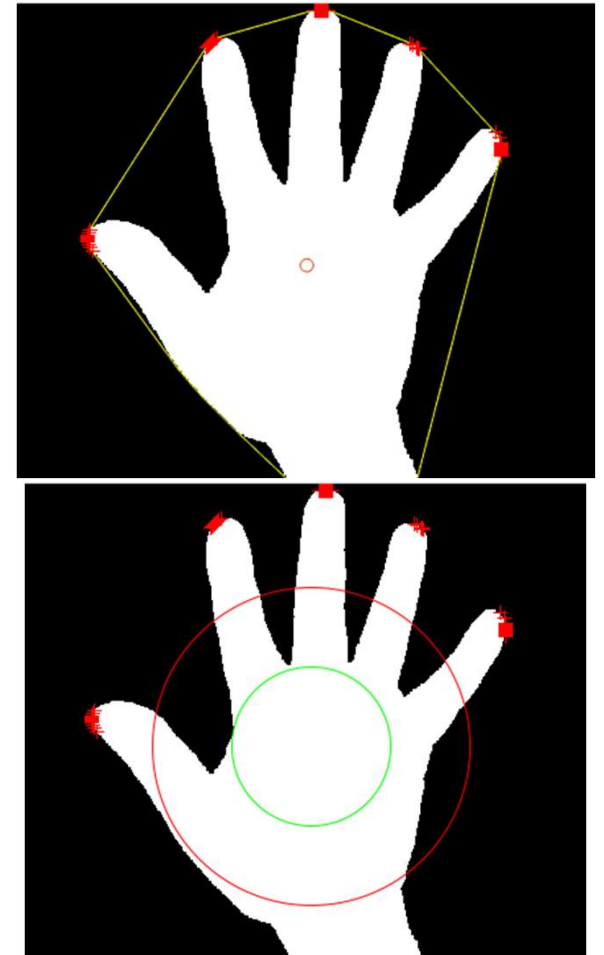
Step 2 – Measure hand, determine if fingers are extended

- Determine hand area, centroid, and convex hull
 - Convex hull allows us to find the fingers

```
blobs = regionprops(BW,'Area','Centroid','ConvexHull');
```
 - Determine if fingers are extended
 - Distance transform helps us estimate radius (green line)

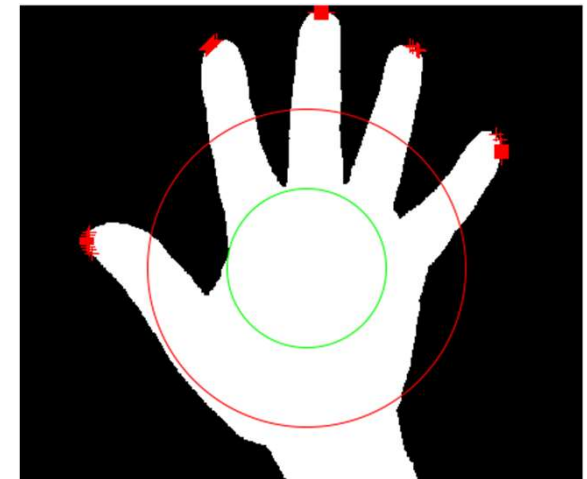
```
bwd = bwdist(~BW);  
radius = bwd(round(c(2)), round(c(1)));
```

 - Sub-select for the points outside 2*radius estimate (red line)
- ```
d = sqrt((x-c(1)).^2+(y-c(2)).^2);
x((d - 2*radius)<0) = 0;
y((d - 2*radius)<0) = 0;
```



## Step 3 – Count the number of fingers

- Eliminate points within 10 pixels of each other
- If no fingers, then it's "rock"
- If 1 finger, then it's "unclassified"
- If 2 fingers, then it's "scissors"
- If 3-5 fingers, then it's "paper"

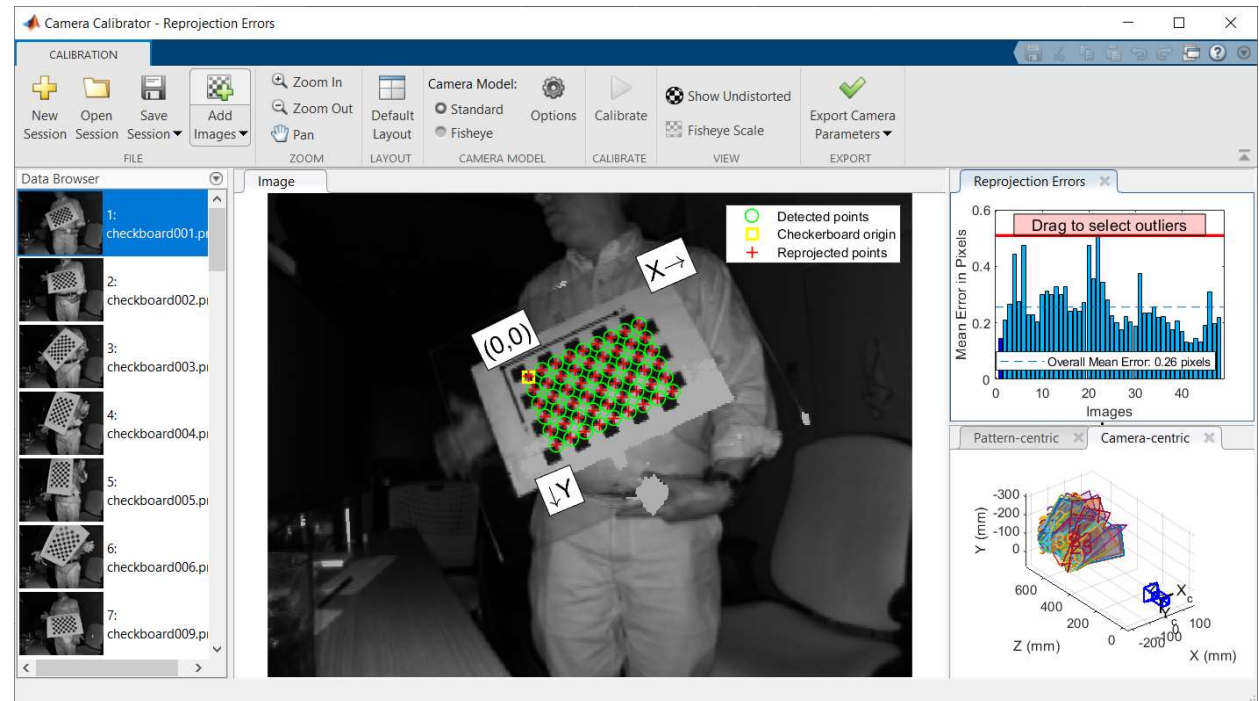




# Accelerate Calibration Workflow with cameraCalibrator

## ■ Steps

- Use checkboard printout with squares of a known size (29mm in this case)
- Load into Camera Calibrator app
- Press the calibration button
- Inspect results



## Next Steps

- Use executable provided to try it yourself
- Download the code
- Evaluate MATLAB<sup>®</sup>, Image Processing Toolbox<sup>™</sup>, Computer Vision Toolbox<sup>™</sup>, and Image Acquisition Toolbox<sup>™</sup>

