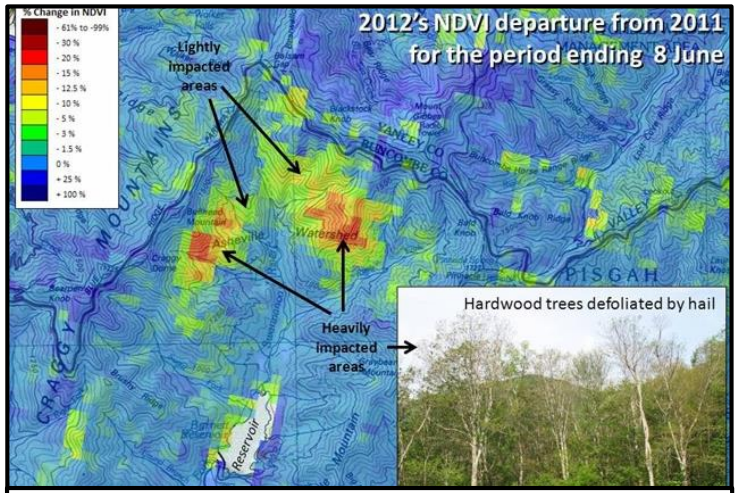
Presentation Title

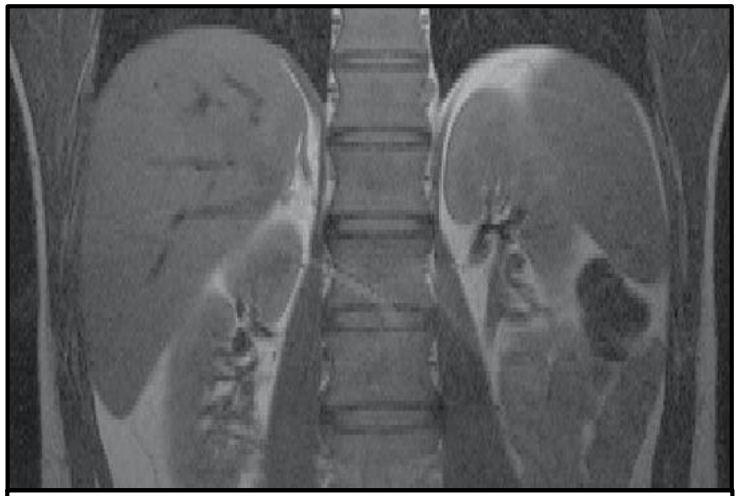
By Author

Date

IPCV Applications



NASA Develops Warning System for Detecting Forest Disturbances



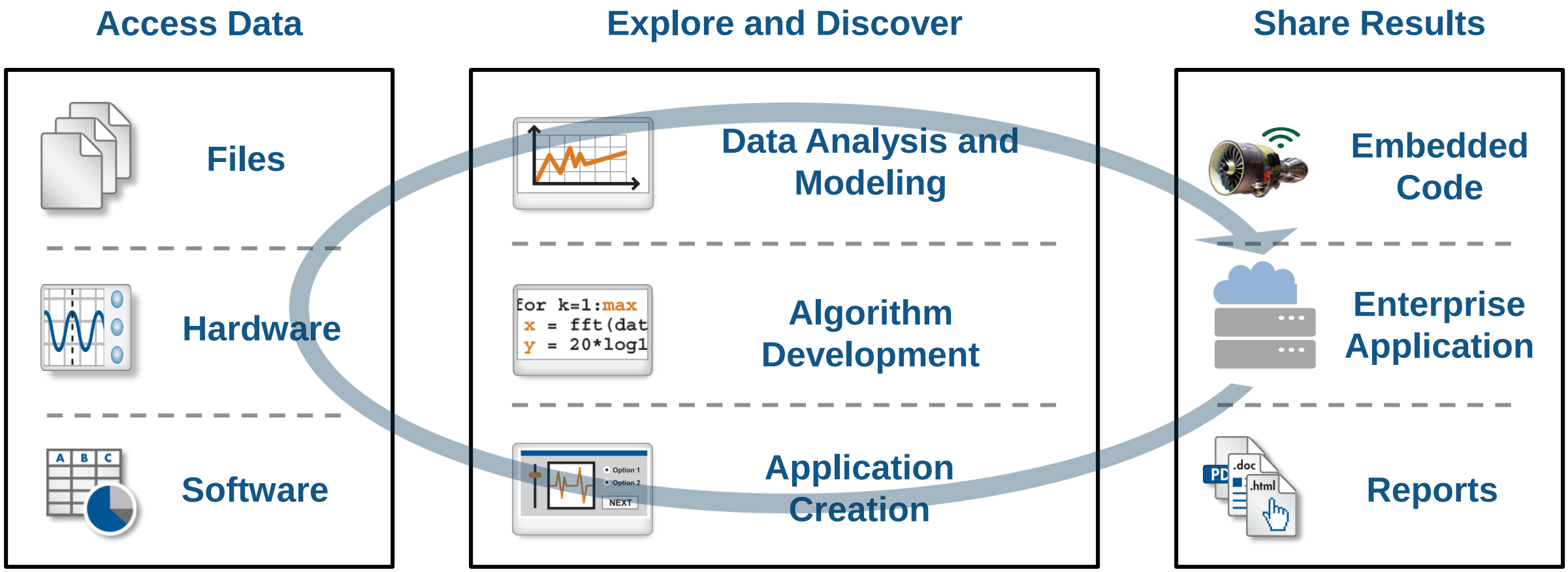
Beth Israel Medical Center Improves MRI Accuracy

CNH Develops Intelligent Filling System for Forage Harvesters

FLIR Accelerates Development of Thermal Imaging FPGA

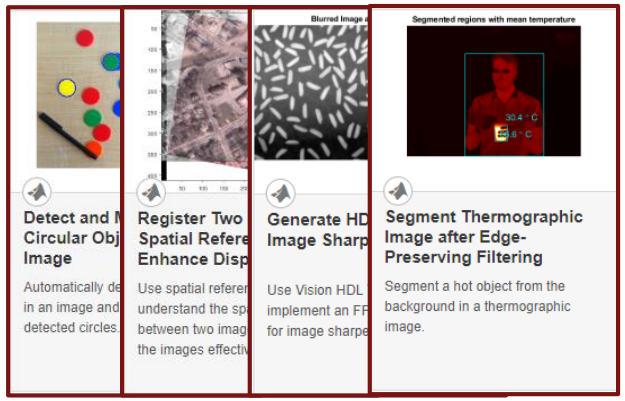
Veoneer (Autoliv) Builds Radar Sensor using LiDAR-Based Verification

Image Processing Workflow



Why Use MATLAB?

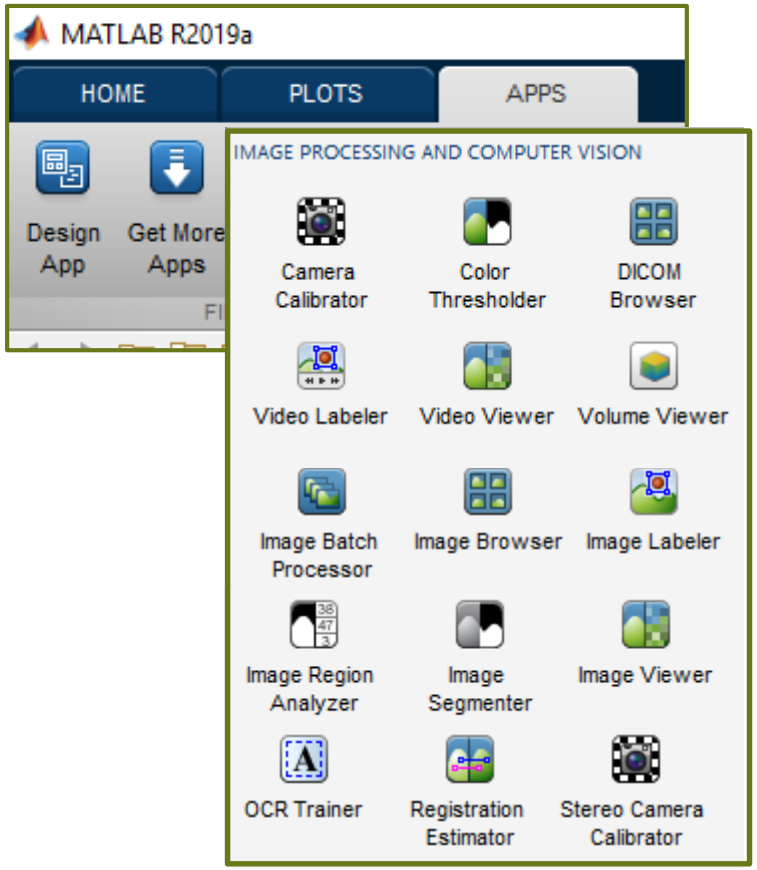
Ease of Use and Thorough Documentation



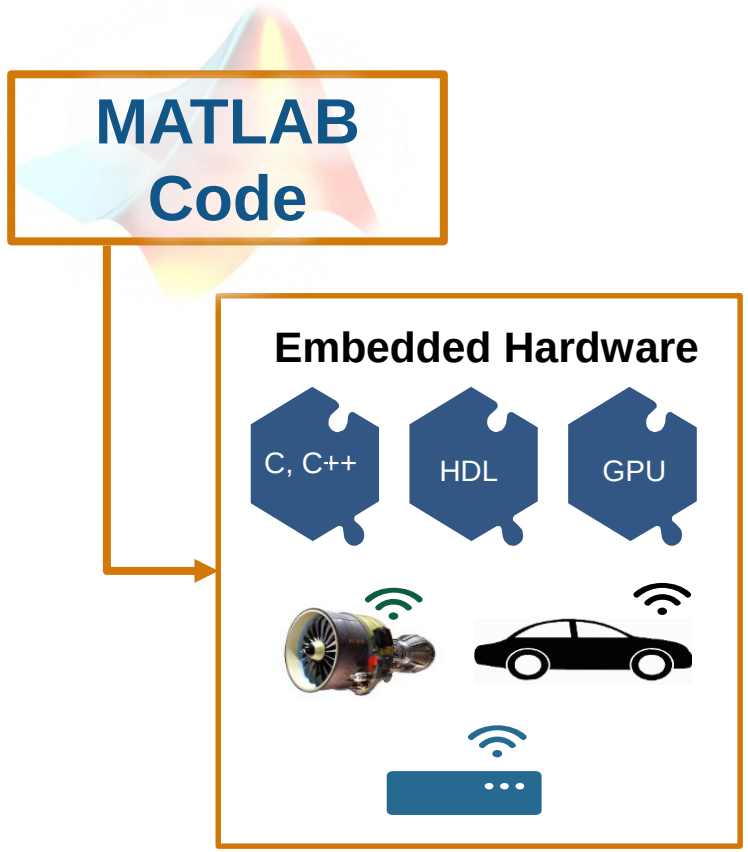
(…)

Need Technical Help?
- Technical Support
- Application Engineers

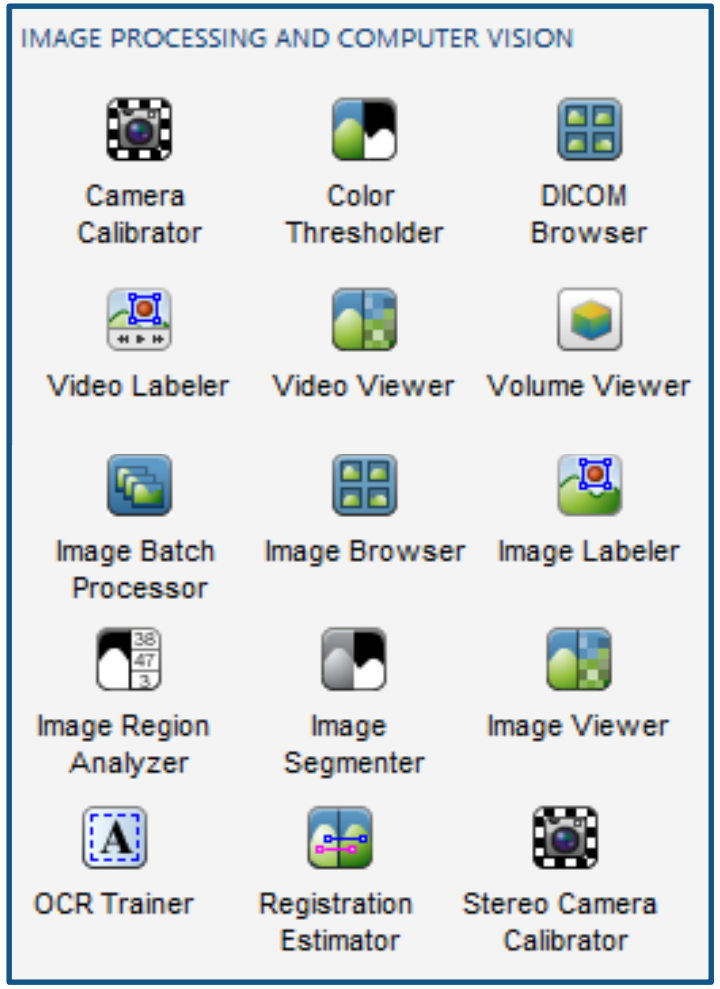
Rapid Prototyping and Algorithm Development

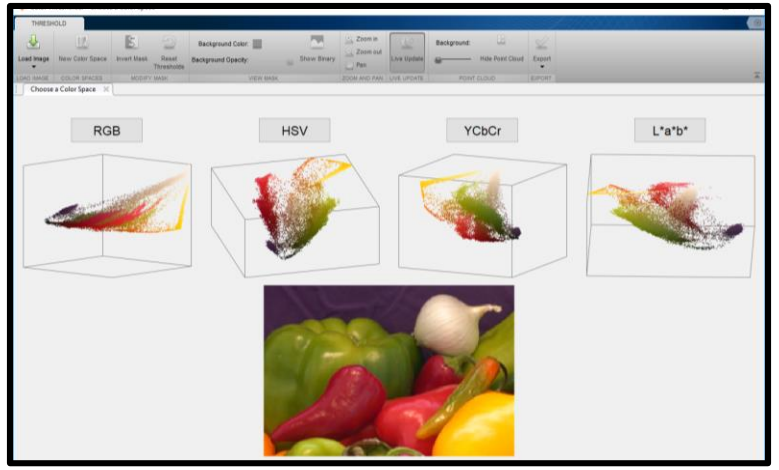


Code Generation for Embedded Deployment



Apps Accelerate Workflows





Color Thresholder App

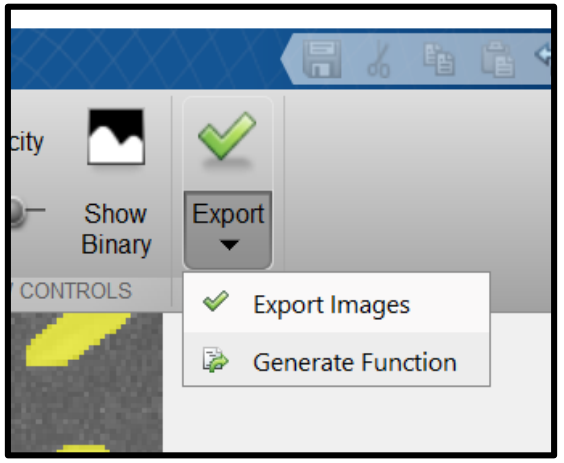


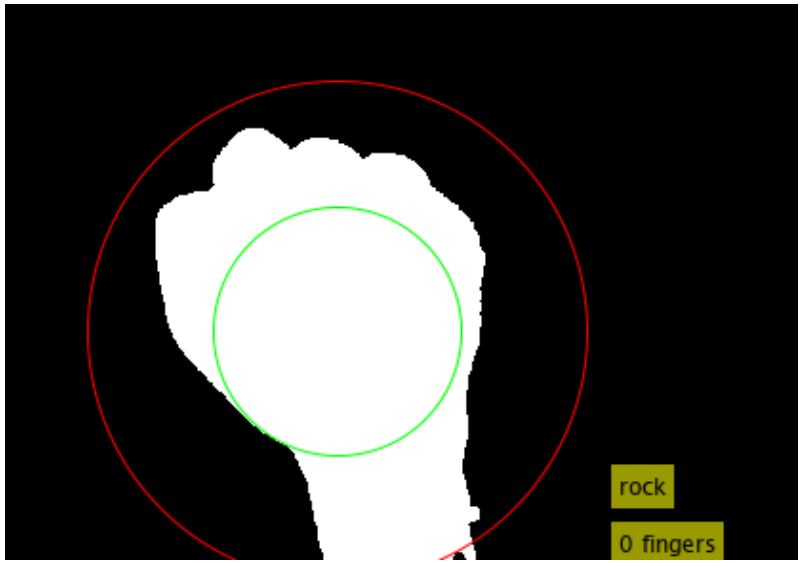
Image Processing Toolbox & Computer Vision Toolbox

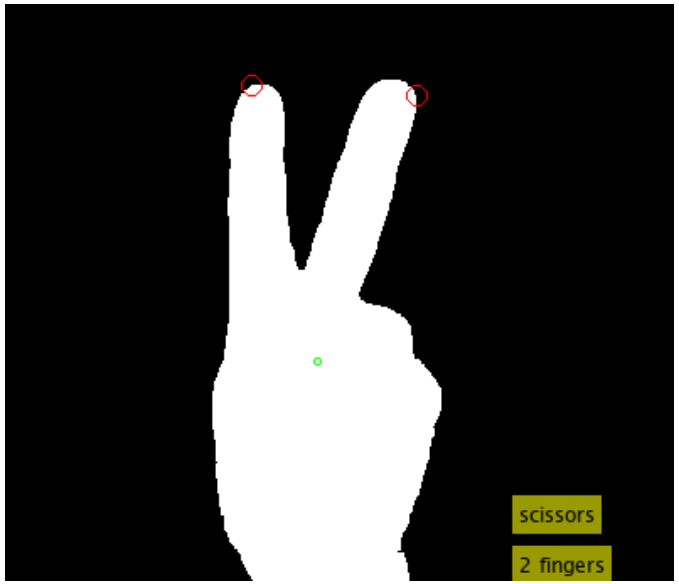
Import, Display, and Exploration	Camera Calibration and 3D-Vision
Geometric Transform and Image Registration	Tracking and Motion Estimation
Image Filtering and Enhancement	Feature Detection and Extraction
Image Segmentation and Analysis	LiDAR and Point Cloud Processing
3D Volumetric Processing	Deep Learning

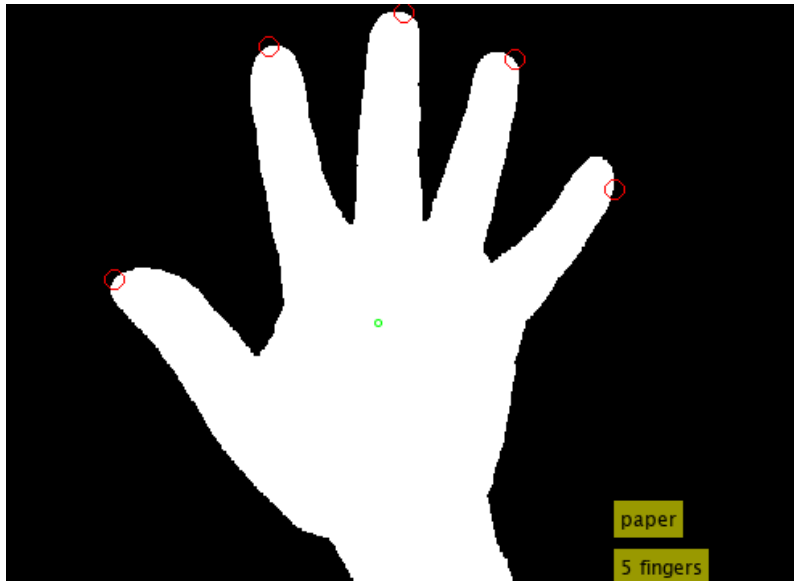
MATLAB with ADI ToF Camera

- Example: Rock, Paper, Scissors

- ToF Camera makes it easy to find the hand
- MATLAB is used to figure out hand signal







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

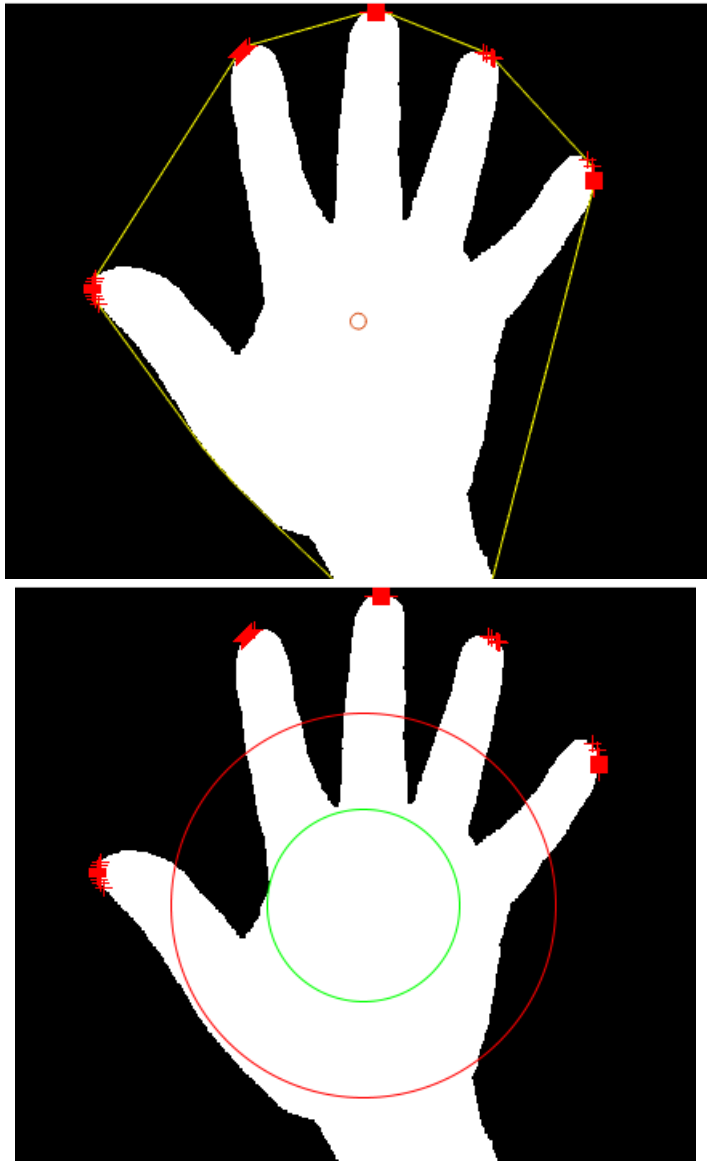
```
blobs = regionprops(BW,'Area','Centroid','ConvexHull');
```

- Determine if fingers are extended
  - Distance transform helps us estimate radius (green line)

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

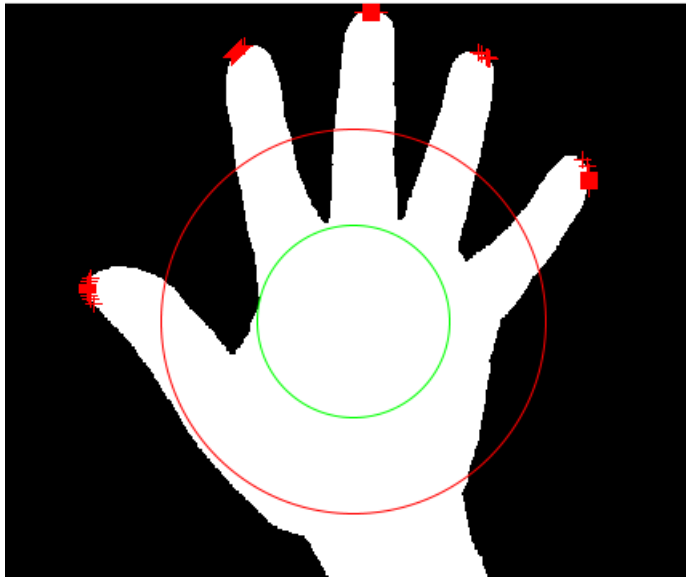
  - Sub-select for the points outside 2*radius estimate (red line)

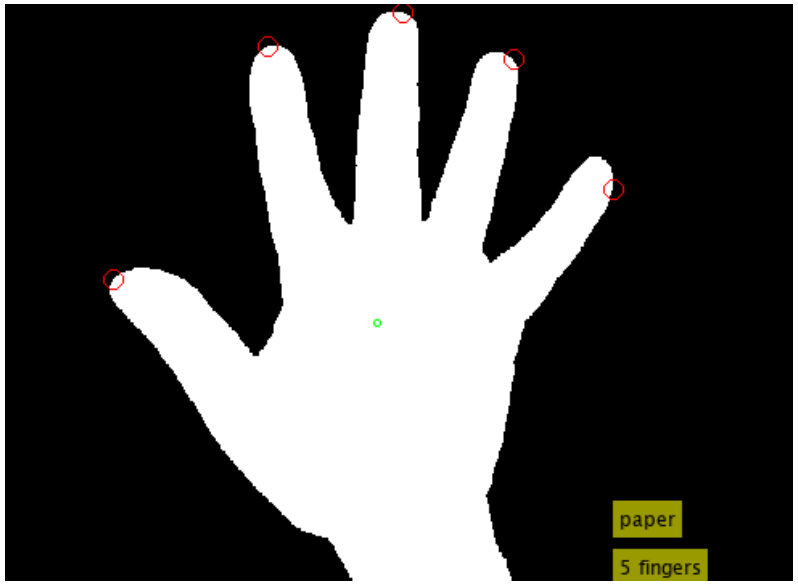
```
d = sqrt((x-c(1)).^2+(y-c(2)).^2);
x((d - 2*radius)<0) = 0;
y((d - 2*radius)<0) = 0;
```



Step 3 – Count the number of fingers

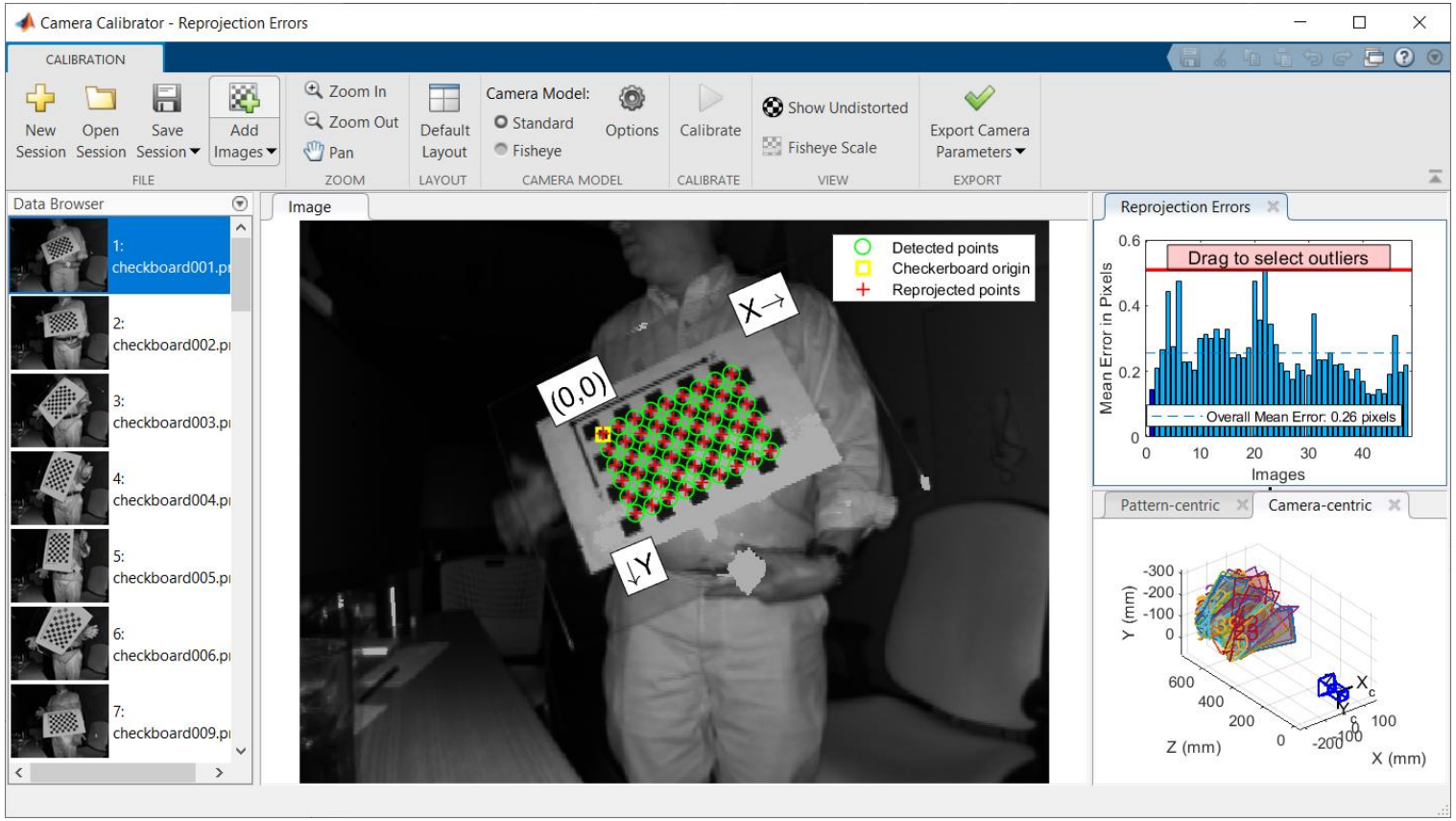
- Eliminate points within 10 pixels of each other
- If no fingers, then it's "rock"
- If 1 finger, then it's "unclassified"
- If 2 fingers, then it's "scissors"
- If 3-5 fingers, then it's "paper"





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, Image Processing Toolbox, and Computer Vision Toolbox