

MATLAB EXPO

A Cloud-based MATLAB Visual Inspection System

Dr. Brett Shoelson, MathWorks



(He/Him)

Arvind Hosagrahara, MathWorks



(He/Him)



What is Automated Visual Inspection?

Automated visual inspection is the evaluation of images or video, typically to detect failures and quality defects—often in manufacturing processes.

Automated Defect Detection

Machine Vision

Optical Inspection

Automated Inspection

MATLAB AI in a Cloud-based Visual Inspection System

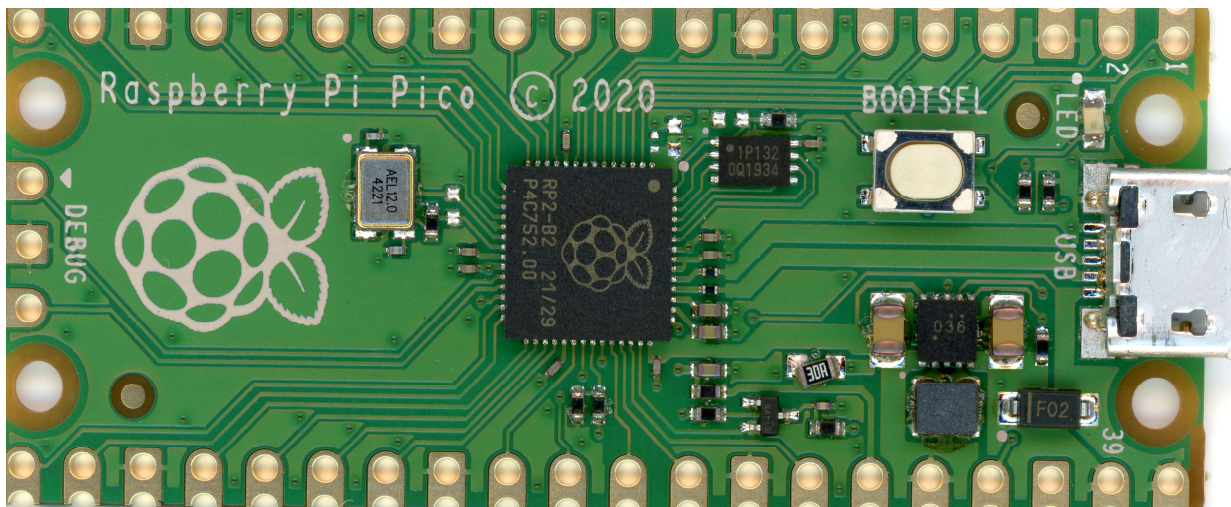
Requirements: A visual inspection system should:

- Be secure
- Run at-scale
- Be re-purposable for different applications

MATLAB's AI solution was operationalized on the cloud using:

- Microservices built to modern standards and best practices for scalability / security
- DevOps processes for agility in development and deployment of AI and vision algorithms

Sample Problem: Detecting and characterizing defects on a Raspberry Pi

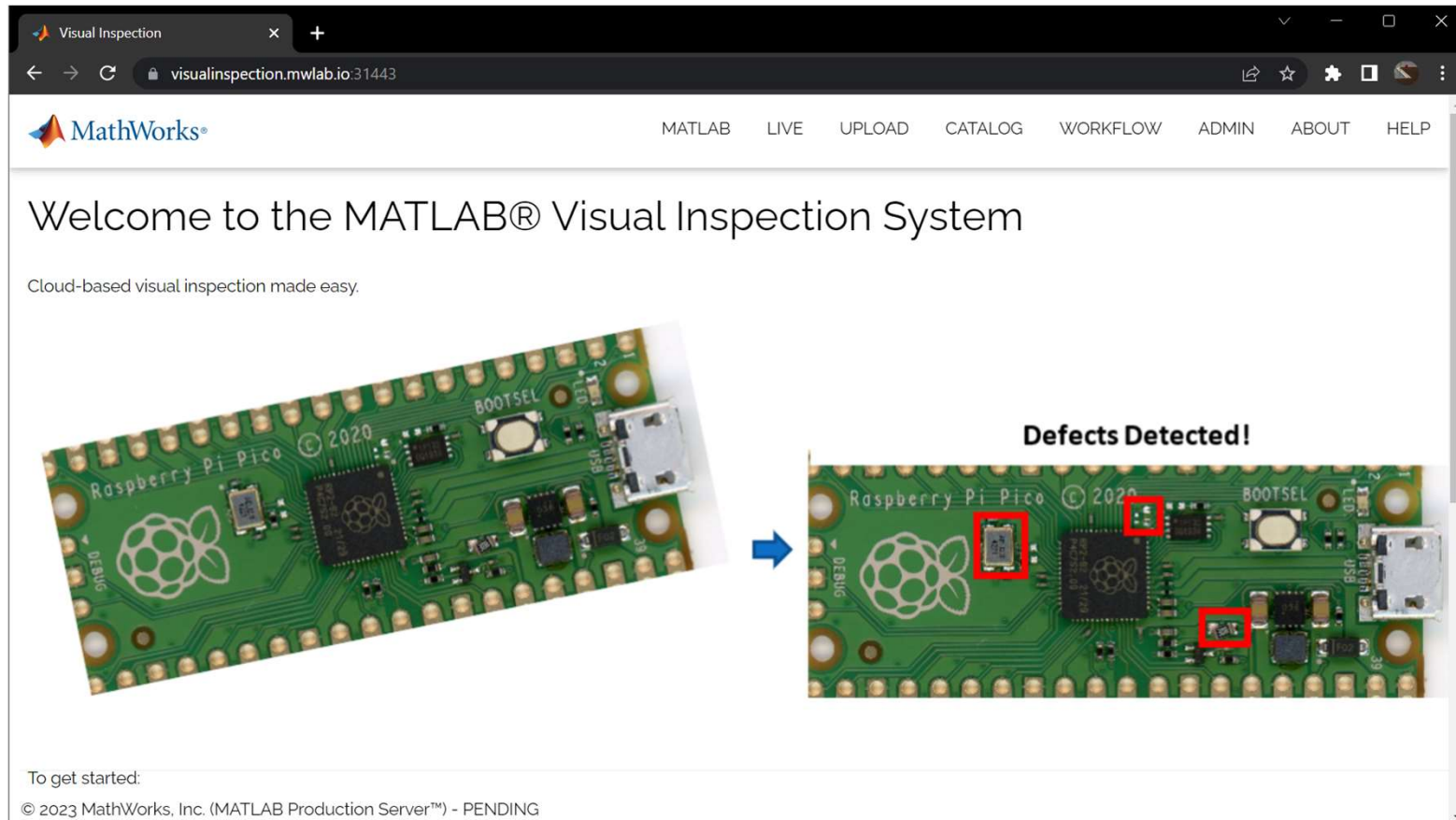


Potential defects include:

- Misaligned components
- Bad assembly
- Damage
- Missing Solder
- Labeling mistakes
- Other?

The defects in these boards were artificially introduced for demonstration purposes

Demonstration



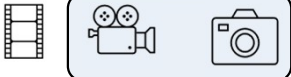
The screenshot shows a web browser window with the URL `visualinspection.mwlab.io:31443`. The page header includes the MathWorks logo and navigation links: `MATLAB`, `LIVE`, `UPLOAD`, `CATALOG`, `WORKFLOW`, `ADMIN`, `ABOUT`, and `HELP`. The main content area displays the text "Welcome to the MATLAB® Visual Inspection System" and "Cloud-based visual inspection made easy." Below this, there are two images of a Raspberry Pi Pico board. The left image shows the board in its original state. A blue arrow points to the right image, which is titled "Defects Detected!". In this second image, three components on the board are highlighted with red boxes, indicating detected defects. The bottom of the page contains the text "To get started:" and "© 2023 MathWorks, Inc. (MATLAB Production Server™) - PENDING".

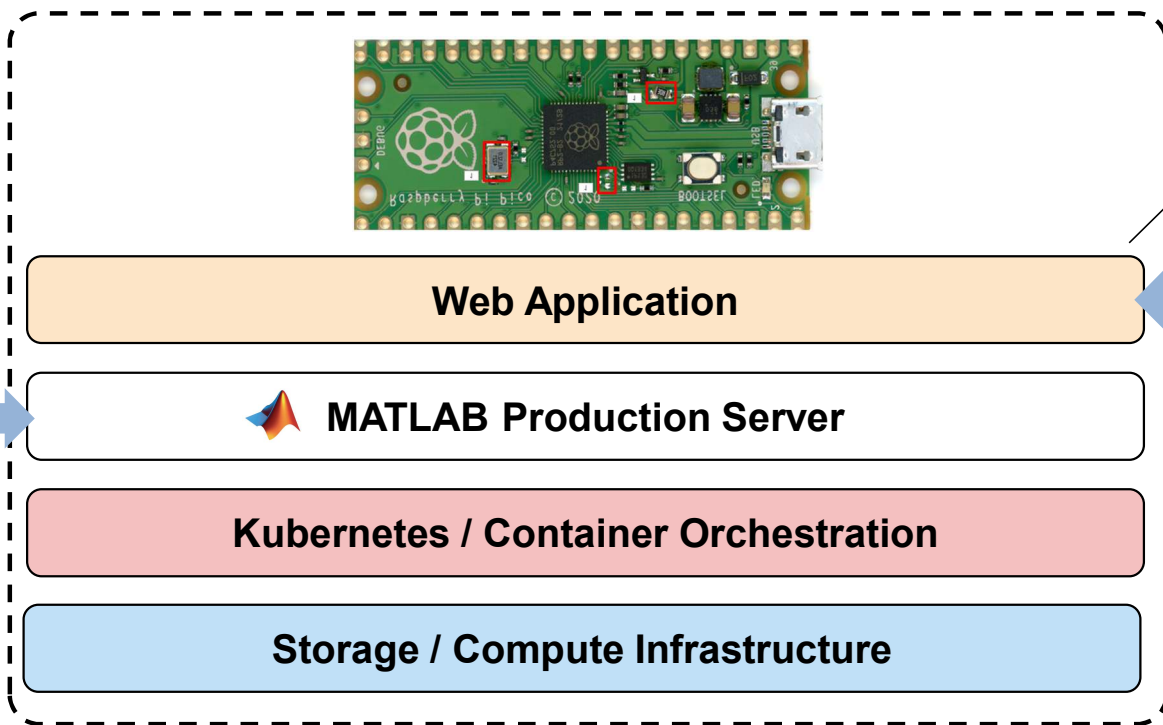
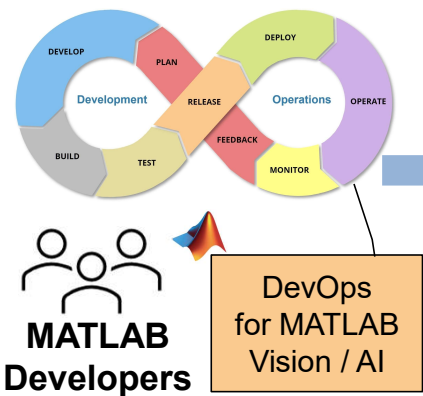
The defects in these boards were artificially introduced for demonstration purposes

System Architecture and Overview

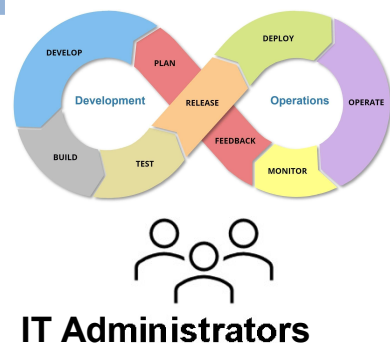
End Users 

Access from any Mobile Device with any standards-compliant browser

Video / Stream  Images



DevOps for Cloud Infrastructure




MATLAB Algorithm Development

Image Processing Toolbox
Computer Vision Toolbox
Deep Learning Toolbox
Statistics and Machine Learning Toolbox

...

MATLAB Algorithm Development

Contribute | Manage Add-Ons



Computer Vision Toolbox Automated Visual Inspection Library
by MathWorks Computer Vision Toolbox Team STAFF

Identify anomalies or defects in images to assist and improve quality assurance processes.

MathWorks Optional Feature

★★★★★ (0)

67 Downloads ⓘ

Updated 15 Mar 2023

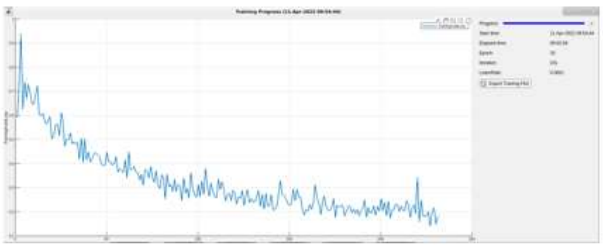
Manage

Overview
Reviews (0)
Discussions (0)

The Computer Vision Toolbox™ Automated Visual Inspection Library offers functions that enable you to train, calibrate, and evaluate anomaly detection networks.

The library enables:

Training and evaluating state of the art anomaly detectors including [PatchCore](#), [FCDD](#), and [FastFlow](#). All detectors support standalone deployment with MATLAB Coder, GPU Coder, and MATLAB Compiler.



Requires

- ✔ Computer Vision Toolbox
- ✔ Image Processing Toolbox

MATLAB Release Compatibility

Created with R2022b
Compatible with R2022b to R2023a

Platform Compatibility

Windows macOS Linux

Tags Add Tags

computer vision
deep learning
image processing

MATLAB Algorithm Development

Documentation Examples Functions Blocks Apps

Automated Visual Inspection

R2023a

Automate quality assurance tasks using anomaly detection and classification techniques

Automated visual inspection (AVI) is a set of techniques used to determine whether an image represents a normal ("good") state or an anomalous ("defective") state. AVI assists and improves quality assurance processes commonly found in manufacturing settings. Modern visual inspection uses machine learning and deep learning techniques to produce useful results.

The specific technique you select to automate a visual inspection task depends on several factors. These factors include the amount of training data available for normal and anomalous samples, the number of anomaly classes to recognize, and the type of localization information required for understanding and monitoring predictions.

To perform automated visual inspection, download the Computer Vision Toolbox™ Automated [Visual Inspection Library](#) from the Add-On Explorer. For more information on downloading add-ons, see [Get and Manage Add-Ons](#). Some functionality also requires Deep Learning Toolbox™.

Functions

expand all

> Load Training Data

> Train Anomaly Detector

> Detect Anomalies Using Deep Learning

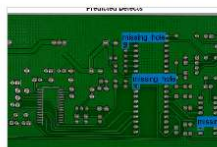
> Visualize and Evaluate Results

Topics

[Getting Started with Anomaly Detection Using Deep Learning](#)

Anomaly detection using deep learning is an increasingly popular approach to automating visual inspection tasks.

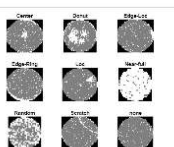
Featured Examples



Detect Defects on Printed Circuit Boards Using YOLO v4 Network

Detect, localize, and classify defects in printed circuit boards (PCBs) using a you only look once version 4 (YOLO v4) deep learning network.

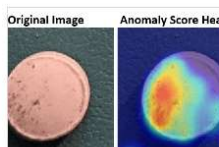
[Open Live Script](#)



Classify Defects on Wafer Maps Using Deep Learning

Classify manufacturing defects on wafer maps using a simple convolutional neural network (CNN).

[Open Live Script](#)



Detect Image Anomalies Using Explainable FCDD Network

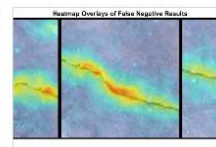
Use an anomaly detector to distinguish between normal pills and pills with anomalous chips or contamination.

[Open Live Script](#)



Detect Anomalies in Pills During Live Image Acquisition

Detect anomalies in pills during live image acquisition.



Detect Image Anomalies Using Pretrained ResNet-18 Feature Embeddings

Train a similarity-based anomaly detector using one-class learning of feature embeddings extracted from a pretrained ResNet-18

[Open Live Script](#)

MATLAB Algorithm Development

Documentation Examples **Functions** Blocks Apps

Automated Visual Inspection — Functions

Load Training Data

| | |
|-------------------------------------|---|
| <code>groundTruth</code> | Ground truth label data |
| <code>sceneLabelTrainingData</code> | Create training data for scene classification from ground truth |
| <code>splitAnomalyData</code> | Split data into training, validation and testing sets for anomaly detection |

Train Anomaly Detector

| | |
|--|--|
| <code>trainFCDDAnomalyDetector</code> | Train fully convolutional data description (FCDD) anomaly detection network |
| <code>trainFastFlowAnomalyDetector</code> | Train FastFlow anomaly detection network |
| <code>trainPatchCoreAnomalyDetector</code> | Train PatchCore anomaly detection network |
| <code>anomalyThreshold</code> | Optimal anomaly threshold for set of anomaly scores and corresponding labels |

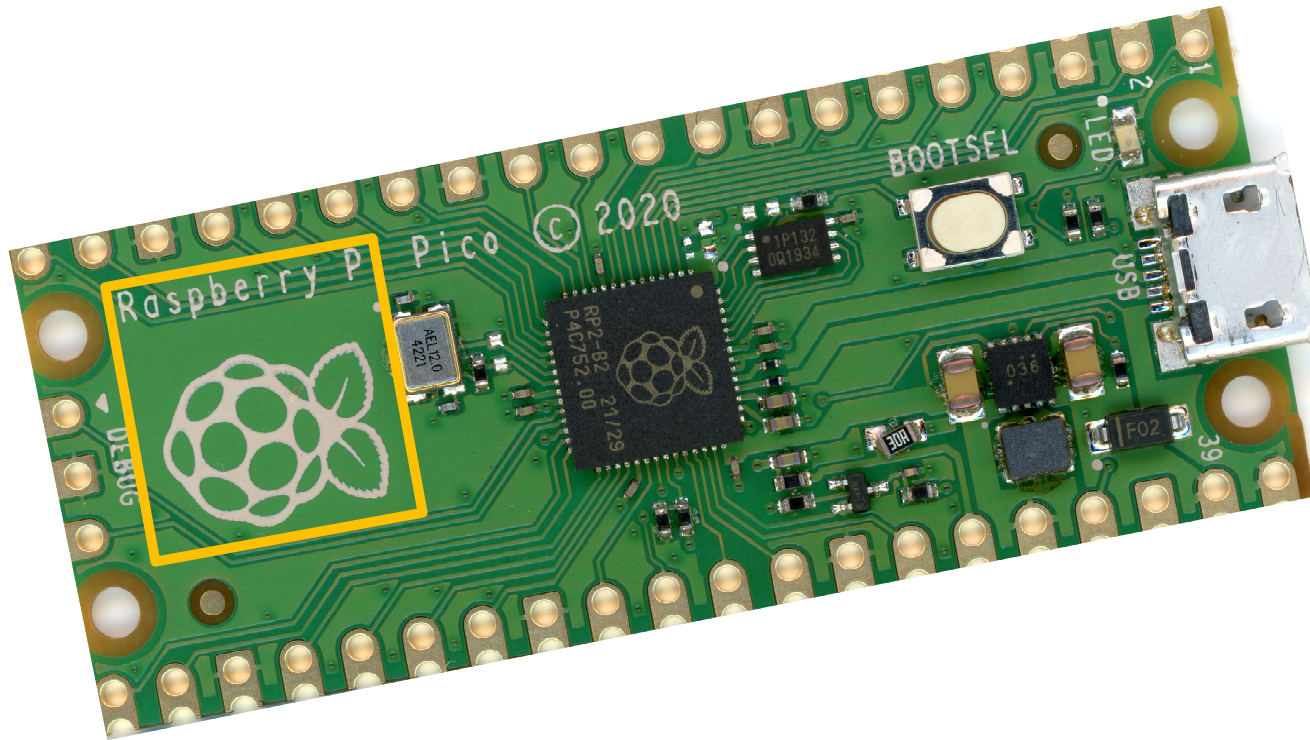
Detect Anomalies Using Deep Learning

| | |
|---------------------------------------|--|
| <code>fcddAnomalyDetector</code> | Detect anomalies using fully convolutional data description (FCDD) network for anomaly detection |
| <code>fastFlowAnomalyDetector</code> | Detect anomalies using FastFlow network |
| <code>patchCoreAnomalyDetector</code> | Detect anomalies using PatchCore network |
| <code>classify</code> | Classify image as normal or anomalous |
| <code>predict</code> | Predict unnormalized anomaly scores |

Visualize and Evaluate Results

| | |
|--|---|
| <code>anomalyMap</code> | Predict per-pixel anomaly score map |
| <code>anomalyMapOverLay</code> | Overlay heatmap on image using per-pixel anomaly scores |
| <code>viewAnomalyDetectionResults</code> | View anomaly detection results |
| <code>evaluateAnomalyDetection</code> | Evaluate anomaly detection results against ground truth |
| <code>anomalyDetectionMetrics</code> | Anomaly detection metrics |

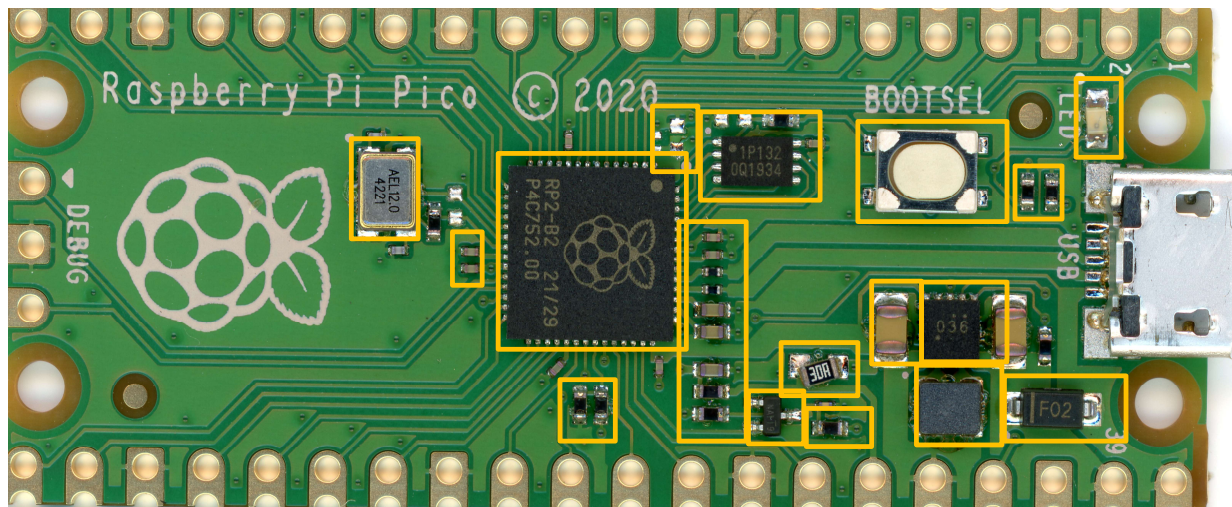
Sample Problem: Detecting Defects on a Raspberry Pi



Template-based orientation and preprocessing...

The defects in these boards were artificially introduced for demonstration purposes

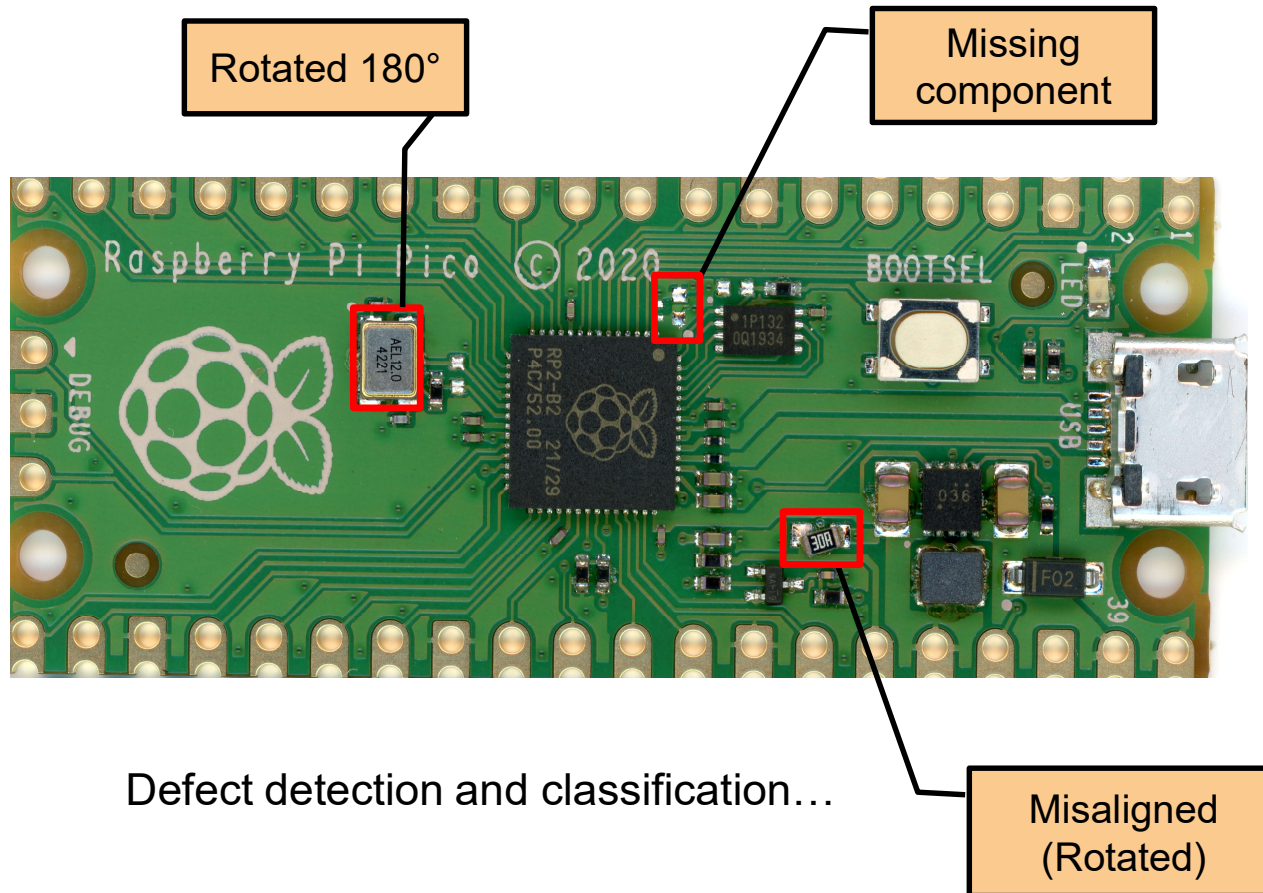
Sample Problem: Detecting Defects on a Raspberry Pi




Component detection...

The defects in these boards were artificially introduced for demonstration purposes

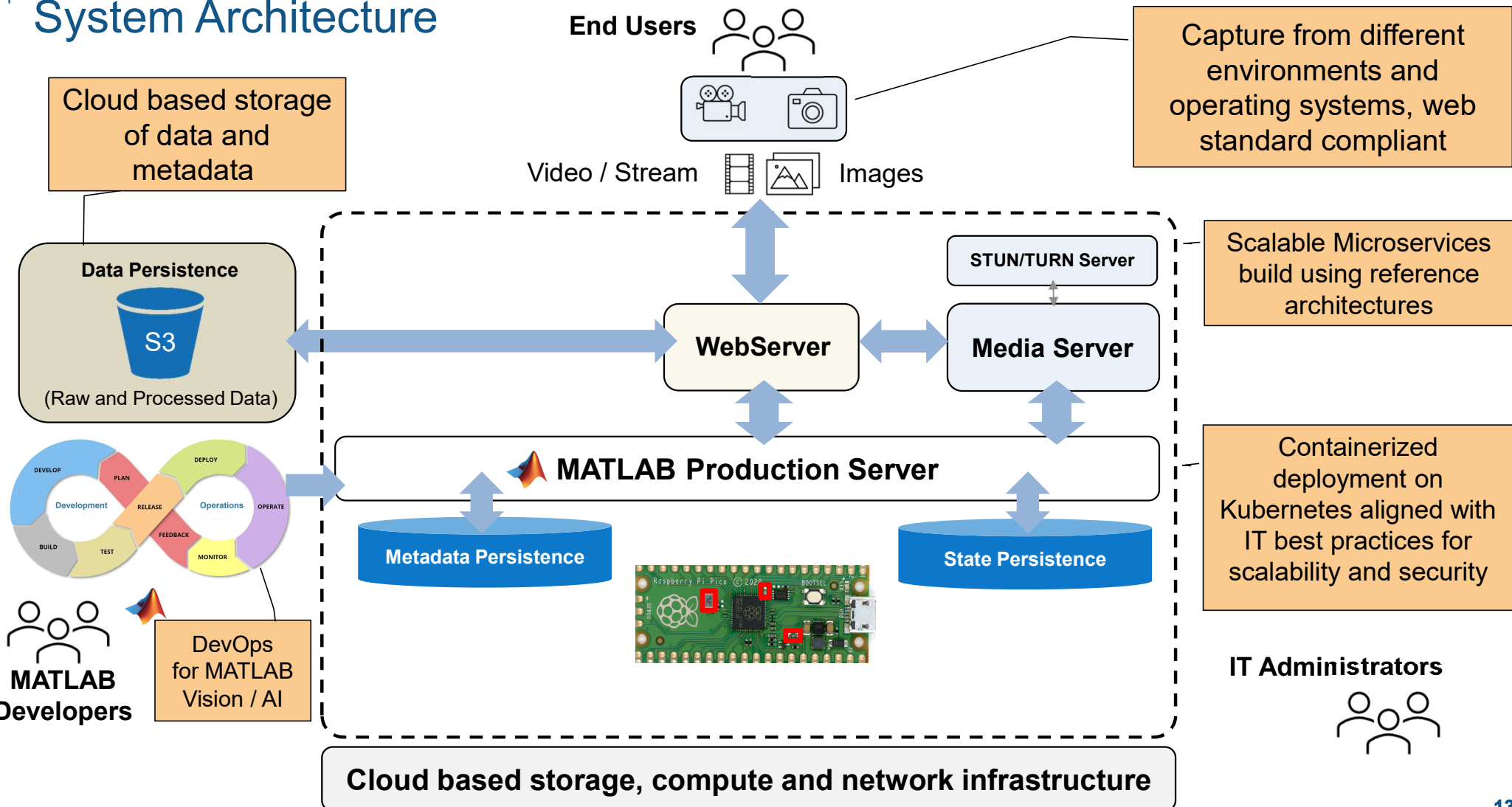
Sample Problem: Detecting Defects on a Raspberry Pi



- QR-Code Triggering
- 
- Live, Constrained Capture (iPhone, iPad)
 - Automatic updating of ground truth and model
 - Scalable, Cloud-Based Analysis and Reporting

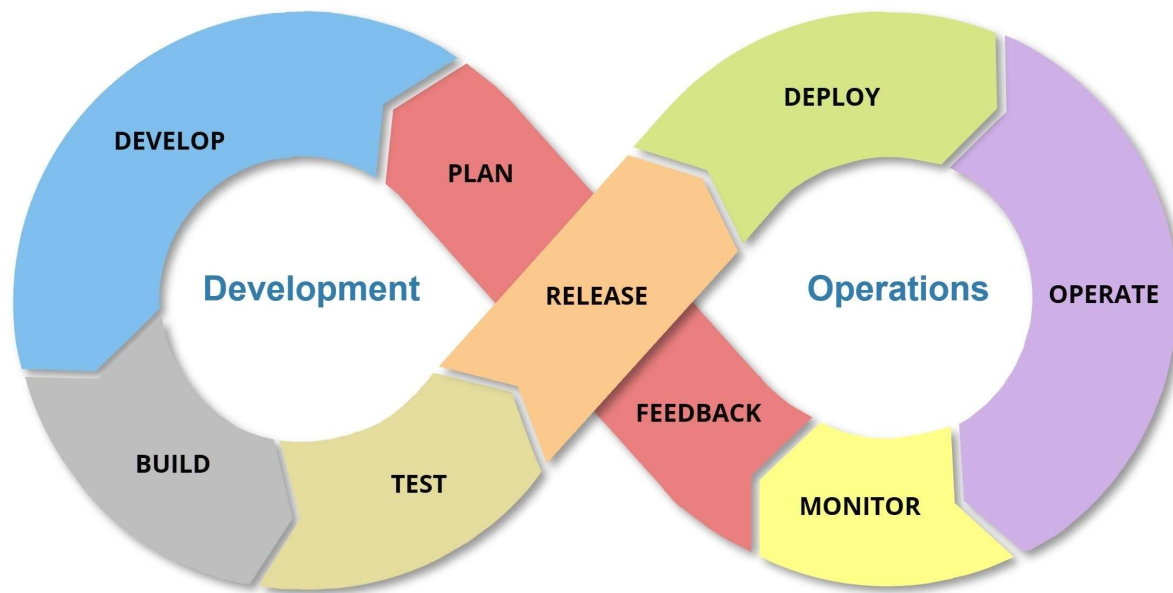
The defects in these boards were artificially introduced for demonstration purposes

System Architecture



MATLAB from Prototype to Production

- Modern DevOps based automated continuous deployment of MATLAB applications



MATLAB Deployment and Scaling

The screenshot displays the OpenShift Cluster Dashboard for a cluster named 'visual-inspection'. The interface includes a sidebar with navigation options like Cluster, Projects/Namespaces, Nodes, and Workload. The main content area shows cluster details such as 'Provider: RKE1', 'Kubernetes Version: v1.24.10', and 'Created: 3.2 days ago'. Key resource metrics are presented in a grid: 202 Total Resources, 3 Nodes, and 25 Deployments. A 'Capacity' section features three bar charts for Pods (Used 61/220, 27.73%), Cores (Reserved 5.76/16, 36.00%; Used 0.44/16, 2.75%), and Memory (Reserved 2.43/31 GiB, Used 5.92/31 GiB). At the bottom, the status of core components (Etcd, Scheduler, Controller Manager) is shown as 'v2.7.1' with green checkmarks. On the right side, a vertical list of deployment cards is visible, each showing its age (e.g., 3.1 days) and a progress bar. One deployment card is expanded to show 'Running' status with 1 instance and a 'Scale' control set to 1.

Take-aways and Conclusion

- MathWorks products along with published reference architectures can be leveraged to build production-grade visual inspection systems for the cloud
- Secure, scalable and agile solutions for AI/Visual Inspection can be built to IT DevOps best practices
- Domain specific toolboxes and support packages are available for MATLAB users to go from prototype to production quickly

Q & A

MATLAB EXPO

Thank you



© 2023 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [mathworks.com/trademarks](https://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.