Computer Vision System Design: Deep Learning, 3D Vision, and Embedded Vision

Dr. Amod Anandkumar
Tabrez Khan
Outline

- 3D Vision and Point Cloud Processing
- Object Classification Using Machine Learning and Deep Neural Networks
- Embedded Vision System Development using Automatic Code Generation
Creating Computer Vision and Machine Learning Algorithms that Can Analyze Works of Art

Challenge
Develop an algorithm that can classify a painting by style, genre, and artist as easily as a human being

Solution
Use MATLAB and machine learning techniques to extract features for classification and train the algorithm

Results
- Machine learning techniques quickly tested and applied
- Thousands of paintings classified with 60% accuracy
- Tools to enable art historians to navigate vast databases of images

“MATLAB is a tremendous advantage because it provides many ways to quickly and easily visualize results. These visualizations enable us to understand the results and use them to inform ongoing AI research.”

Ahmed Elgammal
Rutgers University

Link to article
3D Vision and Point Cloud Processing
What are Point Clouds?

- Point clouds represent a set of data points in a 3-D coordinate system.
- Typically used to measure physical world surfaces.
- Used for navigation and perception in robotics and Advanced Driver Assistance Systems (ADAS).
Common Sources of Point Cloud Data

- LIDAR
- Laser Scanner
- Depth Camera
- Stereo Camera

Point Cloud
Point Cloud Processing

- **I/O and visualization**
  - Live acquisition from Kinect (`pcfromkinect`)
  - Read and write PLY file format (`pcread`, `pcwrite`)
  - Static point cloud viewer (`pcshow`)
  - Streaming point cloud viewer (`pcplayer`)
  - Point cloud difference visualization (`pcshowpair`)

- **Point cloud handling**
  - Object to manage 3-D data (`pointCloud`)
  - Methods to find
    - closest points (`findNearestNeighbors`)
    - points in ROI (`findPointsInROI`)
    - points in radius (`findNeighborsInRadius`)
Point Cloud Registration and Stitching

- Rigid transformation \((pctransform)\)
- Rigid registration \((pcregrigid)\)
  - Iterative closest point (ICP) algorithm
- Downsampling \((pcdownsample)\)
- Denoising \((pcdenoise)\)
- Stitching \((pcmerge)\)

[Link to example]
Point Cloud Fitting to Geometric Shapes

- Estimate normal \( \text{pcnormals} \)
- Fit plane \( \text{pcfitplance} \)
- Fit sphere \( \text{pcfitsphere} \)
- Fit cylinder \( \text{pcfitcylinder} \)

Link to example
Structure From Motion From Multiple Views

Camera pose estimation and 3-D scene reconstruction from multiple views

- Manage view attributes (`viewSet`)
  - Keypoints and matches
  - Camera poses
- Perform 3-D reconstruction from multiple views (`triangulateMultiview`)
- Refine camera poses and 3-D world points (`bundleAdjustment`)

Link to example
Object Classification Using Machine Learning and Deep Neural Networks
Object Classification

Goal: Recognize objects in video stream
Overall Workflow for Object Classification

1. Traditional machine learning
2. Deep learning

Image / Video Frame → Preprocess • Resize / crop • Enhance / denoise → Object Detection • Foreground Extraction → Object Classification • Car or SUV?
## Common Challenges with Object Classification Workflow

<table>
<thead>
<tr>
<th>Steps</th>
<th>Challenge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Importing Data</td>
<td>Managing large sets of labeled images</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>Resizing, Enhancement, Data augmentation</td>
</tr>
<tr>
<td>Choosing a classifier or architecture</td>
<td>Background in machine learning and neural networks</td>
</tr>
<tr>
<td>Training and Classification</td>
<td>Computation-intensive task (program GPUs)</td>
</tr>
</tbody>
</table>
Machine Learning Workflow for Object Classification

Input Image → Feature Extraction → Classification

Training Data → Feature Extraction → Learning or Modelling → Training

Classifier / Model → ‘hat’
Object Classification with Bag of Words

Create Bag of Features

Encode into Histogram of Visual Words

Prepare Training Data

Evaluate the Classifier
Many Options for Feature Extraction and Machine Learning

<table>
<thead>
<tr>
<th>Feature Extraction</th>
<th>Machine Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRISK, FREAK, SURF</td>
<td>SVM</td>
</tr>
<tr>
<td>Histogram of Oriented Gradients (HoG)</td>
<td>Decision trees</td>
</tr>
<tr>
<td>Using box filters (integral images)</td>
<td>AdaBoost</td>
</tr>
<tr>
<td>Bag of visual words</td>
<td>Bagged trees</td>
</tr>
<tr>
<td>Color-based features</td>
<td>k-NN</td>
</tr>
<tr>
<td>Frequency-domain features</td>
<td>Discriminant analysis</td>
</tr>
<tr>
<td></td>
<td>Bayes classifiers</td>
</tr>
</tbody>
</table>

Many permutations and combinations to fit your needs

*Computer Vision System Toolbox*

*Statistics & Machine Learning Toolbox*
Deep learning performs **end-end learning** by learning **features, representations and tasks** directly from **images, text and sound**.
Why is Deep Learning so Popular?

- **Results**: Achieved substantially better results on ImageNet large-scale recognition challenge
  - 95% + accuracy on ImageNet 1000 class challenge

- **Computing Power**: GPU’s and advances in processor technologies have enabled us to train networks on massive sets of data.

- **Data**: Availability of storage and access to large sets of labeled data
  - E.g. ImageNet, PASCAL VoC, Kaggle

<table>
<thead>
<tr>
<th>Year</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-2012 (traditional computer vision and machine learning techniques)</td>
<td>&gt; 25%</td>
</tr>
<tr>
<td>2012 (Deep Learning)</td>
<td>~ 15%</td>
</tr>
<tr>
<td>2015 (Deep Learning)</td>
<td>&lt;5%</td>
</tr>
</tbody>
</table>
Convolutional Neural Networks

- Train “deep” neural networks on structured data (e.g. images, signals, text)
- Implements Feature Learning: Eliminates need for “hand crafted” features
- Trained using GPUs for performance
Two Approaches for Deep Learning

1. Train a deep neural network from scratch

Thoughts:
- **Convolutional Neural Network (CNN)**
- **Learned features**
- **New Task**

- **Car ✓**
- **Truck ✗**
- **Bicycle ✗**

- **Lots of data**

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

Thoughts:
- **Pre-trained CNN**
- **Fine-tune network weights**

- **Car ✓**
- **Truck ✗**

- **Medium amounts of data**
Two Deep Learning Approaches

Approach 1: Train a Deep Neural Network from Scratch

Convolutional Neural Network (CNN)

Learned features

- Car ✓
- Truck x
- Bicycle x

Recommended only when:

<table>
<thead>
<tr>
<th>Training data</th>
<th>1000s to millions of labeled images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation</td>
<td>Compute intensive (requires GPU)</td>
</tr>
<tr>
<td>Training Time</td>
<td>Days to Weeks for real problems</td>
</tr>
<tr>
<td>Model accuracy</td>
<td>High (can over fit to small datasets)</td>
</tr>
</tbody>
</table>
Two Deep Learning Approaches

Approach 2: Fine-Tune a Pre-Trained Model (Transfer Learning)

CNN trained on massive sets of data
- Learned robust representations of images from larger data set
- Can be fine-tuned for use with *new data or task* with small – medium size datasets

**Recommended when:**
- **Training data**: 100s to 1000s of labeled images (small)
- **Computation**: Moderate computation (GPU optional)
- **Training Time**: Seconds to minutes
- **Model accuracy**: Good, depends on the pre-trained CNN model
Object Classification Using Convolutional Neural Networks

Transfer Learning: Fine-Tune a Pre-Trained Model

Pre-trained CNN (AlexNet – 1000 Classes)

New Data

Car

SUV

New Task – 2 Class Classification
## Challenge

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Solution</th>
<th>Validated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing large sets of labeled images</td>
<td><code>imageSet</code> or <code>imageDataStore</code> to handle large sets of images</td>
<td>✓</td>
</tr>
<tr>
<td>Resizing, Enhancement, Data augmentation</td>
<td><code>imresize</code>, <code>imcrop</code>, <code>imadjust</code>, <code>imageInputLayer</code>, etc.</td>
<td>✓</td>
</tr>
<tr>
<td>Background in machine learning and neural networks</td>
<td>Classification Learner App, intuitive well-documented interfaces and examples</td>
<td>✓</td>
</tr>
<tr>
<td>Computation intensive task (program GPUs)</td>
<td>Training supported on GPUs No GPU programming expertise required</td>
<td>✓</td>
</tr>
</tbody>
</table>
Embedded Vision System Development using Automatic Code Generation
Typical Workflow for Embedded Vision System Development

Algorithm Development

• Is my idea new? What is required?
• Is it robust to all kinds of conditions? (lighting noise, etc.)

Implementation

• Consideration of HW platform
  • FPGA? CPU? DSP? GPU?
• Speed and resource requirement
  • Resolution, Frame-rate constraint
  • Memory constraint

Development of the algorithm and implementation are often done by different groups
MATLAB Coder app with Integrated Editor and Simplified Workflow

New user interface simplifies code generation workflow
Embedded Coder for Optimized Code

Embedded Coder extends MATLAB Coder with

- Processor-specific code generation
  - Built-in support for select processors
  - Open APIs for use with any processor

- Speed, memory, and code appearance advanced features
Supported MATLAB Language Features and Functions

Broad set of language features and functions/system objects supported for code generation

<table>
<thead>
<tr>
<th>Matrices and Arrays</th>
<th>Data Types</th>
<th>Programming Constructs</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Matrix operations</td>
<td>• Complex numbers</td>
<td>• Arithmetic, relational, and logical operators</td>
<td>• MATLAB functions and subfunctions</td>
</tr>
<tr>
<td>• N-dimensional arrays</td>
<td>• Integer math</td>
<td>• Program control (if, for, while, switch)</td>
<td>• Variable-length argument lists</td>
</tr>
<tr>
<td>• Subscripting</td>
<td>• Double/single-precision</td>
<td></td>
<td>• Function handles</td>
</tr>
<tr>
<td>• Frames</td>
<td>• Fixed-point arithmetic</td>
<td></td>
<td>Supported algorithms</td>
</tr>
<tr>
<td>• Persistent variables</td>
<td>• Characters</td>
<td></td>
<td>• More than 1300 MATLAB operators, functions, and System objects for:</td>
</tr>
<tr>
<td>• Global variables</td>
<td>• Structures</td>
<td></td>
<td>• Communications</td>
</tr>
<tr>
<td></td>
<td>• Cell arrays</td>
<td></td>
<td>• Computer vision</td>
</tr>
<tr>
<td></td>
<td>• Numeric class</td>
<td></td>
<td>• Image processing</td>
</tr>
<tr>
<td></td>
<td>• Variable-sized data</td>
<td></td>
<td>• Neural networks</td>
</tr>
<tr>
<td></td>
<td>• MATLAB Class</td>
<td></td>
<td>• Phased Array signal processing</td>
</tr>
<tr>
<td></td>
<td>• System objects</td>
<td></td>
<td>• Robotics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Signal processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Statistics and machine learning</td>
</tr>
</tbody>
</table>
Supported Functions

- Aerospace Toolbox
- Communications System Toolbox
- Computer Vision System Toolbox
- DSP System Toolbox
- Image Acquisition Toolbox
- Image Processing Toolbox
- Neural Networks Toolbox
- Phased Array System Toolbox
- Robotics System Toolbox
- Signal Processing Toolbox
- Statistics and Machine Learning Toolbox
- System Identification Toolbox
With MATLAB Coder, design engineers can:

- Maintain one design in MATLAB
- Design faster and get to C quickly
- Test more systematically and frequently
- Spend more time improving algorithms in MATLAB
Vision HDL Toolbox

*Design and prototype video image processing systems*

- Modeling hardware behavior of the algorithms
  - Pixel-based functions and blocks
  - Conversion between frames and pixels
  - Standard and custom frame sizes

- Prototyping algorithms on hardware
  - *(With HDL Coder)* Efficient and readable HDL code
  - *(With HDL Verifier)* FPGA-in-the-loop testing and acceleration
Pixel Based Video Image Algorithms

- **Analysis & Enhancement**
  - Edge Detection, Median Filter

- **Conversions**
  - Chroma Resampling, Color-Space Converter
  - Demosaic Interpolator, Gamma Corrector, Look-up Table

- **Filters**
  - Image Filter, Median Filter

- **Morphological Operations**
  - Dilation, Erosion,
  - Opening, Closing

- **Statistics**
  - Histogram
  - Image Statistics

- **I/O Interfaces**
  - Frame to Pixels, Pixels to Frame, FIL versions

- **Utilities**
  - Pixel Control Bus Creator
  - Pixel Control Bus Selector
Frame To Pixels and Pixels To Frame
Shipping Examples

**Simulink Examples**

- **Gamma Correction**
  Uses: Simulink, HDL Coder, Computer Vision System Toolbox

- **Histogram Equalization**
  Uses: Simulink, HDL Coder, Computer Vision System Toolbox

- **Edge Detection and Image Overlay**
  Uses: Simulink, HDL Coder, Computer Vision System Toolbox

- **Edge Detection and Image Overlay with Impaired Frame**
  Uses: Simulink, HDL Coder, Computer Vision System Toolbox

**Matlab Examples**

- **Pixel-Streaming Design in MATLAB**
  Uses: Matlab, Computer Vision System Toolbox

- **Accelerate a Pixel-Streaming Design Using MATLAB Coder**
  Uses: Matlab, Computer Vision System Toolbox, Matlab Coder

- **Enhanced Edge Detection from Noisy Color Video**
  Uses: Matlab, Computer Vision System Toolbox, Matlab Coder

- **Image Filtering using Vision HDL Blocks**
  Uses: Simulink, HDL Coder, Computer Vision System Toolbox
A Complete Solution for Embedded Vision

Frame based

Pixel based

Computer Vision System Toolbox
Image Processing Toolbox
Vision HDL Toolbox
HDL Coder
MATLAB Coder
Fixed Point Designer
HDL Verifier
MATLAB / Simulink
Challenge
Accelerate the implementation of advanced thermal imaging filters and algorithms on FPGA hardware

Solution
Use MATLAB to develop, simulate, and evaluate algorithms, and use HDL Coder to implement the best algorithms on FPGAs

Results
- Time from concept to field-testable prototype reduced by 60%
- Enhancements completed in hours, not weeks
- Code reuse increased from zero to 30%

"With MATLAB and HDL Coder we are much more responsive to marketplace needs. We now embrace change, because we can take a new idea to a real-time-capable hardware prototype in just a few weeks. There is more joy in engineering, so we've increased job satisfaction as well as customer satisfaction."

Nicholas Hogasten
FLIR Systems
Use MATLAB for Image Processing and Computer Vision

- MATLAB is a great environment for algorithm development
  - Simple and intuitive interfaces
  - Apps to simplify time-consuming workflows
- Extensive algorithm coverage
  - Image analysis and enhancement,
  - Object detection, recognition and tracking, 3D vision, point cloud processing
  - Machine learning and deep neural networks
- Exhaustive documentation and application examples
- Integrated workflow from algorithm development to implementation on embedded systems
Call to Action
Get Hardware Support Packages

From the MATLAB Toolstrip:
Add-Ons → Get Hardware Support Packages

From the MATLAB Command Line:
>> supportPackageInstaller
Call To Action

- Explore more examples at
  

- Learn more with recorded webinars
  
  - Image Processing Made Easy
  - Computer Vision with MATLAB for Object Detection and Tracking
  - Machine Learning Made Easy
  - Simplifying the Development of Computer Vision Systems
Training Services

Exploit the full potential of MathWorks products

Flexible delivery options:

- Public training available in several cities
- Onsite training with standard or customized courses
- Web-based training with live, interactive instructor-led courses

More than 30 course offerings including:

- Image Processing with MATLAB
- Computer Vision with MATLAB
- Machine Learning with MATLAB
- MATLAB to C with MATLAB Coder

www.mathworks.in/training
Training Services
Exploit the full potential of MathWorks products

Computer Vision with MATLAB

This one-day course provides hands-on experience with performing computer vision tasks. Examples and exercises demonstrate the use of appropriate MATLAB® and Computer Vision System Toolbox™ functionality.
Topics include:

- Importing, displaying and annotating images and videos
- Detecting, extracting and matching object features
- Automatically aligning images using geometric transformations
- Detecting objects in images and videos
- Tracking objects and estimating their motion in a video
- Removing lens distortion from images
- Measuring planar objects

MATLAB to C with MATLAB Coder

This two-day course focuses on generating C code from MATLAB® code using MATLAB Coder™. The focus is on developing MATLAB code that is ready for code generation, generating C code that meets optimization requirements, and integrating generated code into parent projects and external modules. This course is intended for intermediate to advanced MATLAB users.
# MathWorks Training

**Upcoming Public Trainings** *(see our website for full year schedule)*

<table>
<thead>
<tr>
<th>Course</th>
<th>Dates</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB Fundamentals</td>
<td>Apr 11 – 13</td>
<td>Bangalore</td>
</tr>
<tr>
<td>Simulink for System and Algorithm Modeling</td>
<td>Apr 14 – 15</td>
<td>Bangalore</td>
</tr>
<tr>
<td>Signal Processing with MATLAB</td>
<td>May 2 -3</td>
<td>Bangalore</td>
</tr>
<tr>
<td><strong>Image Processing with MATLAB</strong></td>
<td>May 4 - 5</td>
<td>Bangalore</td>
</tr>
<tr>
<td><strong>Computer Vision with MATLAB</strong></td>
<td>May 6</td>
<td>Bangalore</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>May 16-18</td>
<td>Pune</td>
</tr>
<tr>
<td>Simulink for System and Algorithm Modeling</td>
<td>May 19-20</td>
<td>Pune</td>
</tr>
<tr>
<td><strong>Programming Xilinx Zynq SoCs with MATLAB and Simulink</strong></td>
<td>July 13 - 14</td>
<td>Bangalore</td>
</tr>
<tr>
<td><strong>MATLAB to C with MATLAB Coder</strong></td>
<td>Sep 19 - 20</td>
<td>Bangalore</td>
</tr>
</tbody>
</table>

**Self-Paced Courses**

<table>
<thead>
<tr>
<th>Course</th>
<th>Dates</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB Fundamentals</td>
<td>On Demand</td>
<td>Online</td>
</tr>
<tr>
<td>MATLAB Programming Techniques</td>
<td>On Demand</td>
<td>Online</td>
</tr>
<tr>
<td>MATLAB for Data Processing and Visualization</td>
<td>On Demand</td>
<td>Online</td>
</tr>
<tr>
<td>MATLAB for Financial Applications</td>
<td>On Demand</td>
<td>Online</td>
</tr>
</tbody>
</table>

**MathWorks Certification Exams**

<table>
<thead>
<tr>
<th>Certification</th>
<th>Dates</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>MathWorks Certified MATLAB Associate</td>
<td>June 27</td>
<td>Pune</td>
</tr>
<tr>
<td>MathWorks Certified MATLAB Associate</td>
<td>Aug 29</td>
<td>Bangalore</td>
</tr>
</tbody>
</table>

Guaranteed to run

Email: [training@mathworks.in](mailto:training@mathworks.in)  
URL: [http://www.mathworks.in/services/training](http://www.mathworks.in/services/training)  
Phone: 080-6632-6000
### MathWorks India Contact Details

<table>
<thead>
<tr>
<th>URL:</th>
<th><a href="http://www.mathworks.in">http://www.mathworks.in</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>E-mail:</td>
<td><a href="mailto:info@mathworks.in">info@mathworks.in</a></td>
</tr>
<tr>
<td>Technical Support:</td>
<td><a href="http://www.mathworks.in/myservicerequests">www.mathworks.in/myservicerequests</a></td>
</tr>
<tr>
<td>Tel:</td>
<td>+91-80-6632 6000</td>
</tr>
<tr>
<td>Fax:</td>
<td>+91-80-6632 6010</td>
</tr>
</tbody>
</table>

- **MathWorks India Private Limited**
  9th Floor, ‘B’ Wing, Etamin Block
  Prestige Technology Park II
  Marathahalli – Sarjapur Ring Road
  Bangalore – 560103, Karnataka India

---

**Thank You for Attending**

**Talk to Us – We are Happy to Support You**