Complete Computer Vision Engineer Course - Detailed Study Plan

Phase 1: Mathematical Foundations - Detailed Study Plan

Duration: 3-4 weeks (60-80 hours total)

Module 1: Linear Algebra for Computer Vision (Week 1)

Primary Textbooks

1. **"Linear Algebra and Its Applications" by Gilbert Strang** (Chapters 1-7)
   - Focus: Chapters 1-3 (vectors, systems), 5-6 (eigenvalues, SVD)
2. **"Introduction to Linear Algebra" by Gilbert Strang** (More accessible alternative)

Online Courses

1. **MIT 18.06 Linear Algebra** (Free on MIT OpenCourseWare)
   - Lectures 1-16, 29-34 (focus on applications)
   - Professor Gilbert Strang's legendary course
   - Video lectures + problem sets with solutions
2. **3Blue1Brown - Essence of Linear Algebra** (YouTube)
   - Visual intuition for all key concepts
   - Watch before diving into rigorous proofs

Practical Resources

- **Python Implementation**: NumPy and SciPy documentation
- **Interactive Learning**: Khan Academy Linear Algebra
- **Practice Problems**: MIT 18.06 problem sets

Week 1 Schedule (20 hours)

Day 1-2: Vectors and Vector Spaces (6 hours)

- Theory: Vector operations, linear independence, span, basis
- Practice: Implement vector operations in NumPy
- Resources: MIT 18.06 Lectures 1-3, 3Blue1Brown videos 1-4

Day 3-4: Matrix Operations and Systems (6 hours)

- Theory: Matrix multiplication, inverses, Gaussian elimination
- Practice: Solve linear systems using NumPy.linalg

- Resources: MIT 18.06 Lectures 4-8, Strang Ch. 2-3

Day 5-6: Eigenvalues and Eigenvectors (8 hours)

- Theory: Characteristic equation, diagonalization
- Practice: Compute eigendecomposition, understand geometric meaning
- Resources: MIT 18.06 Lectures 21-25, implement PCA from scratch

Programming Assignments

```python
# Assignment 1: Implement basic linear algebra operations
# Assignment 2: Build PCA from scratch
# Assignment 3: Image compression using SVD
```

Module 2: Geometry and Projective Geometry (Week 2)

Primary Textbooks

1. **"Multiple View Geometry" by Hartley & Zisserman** (Chapters 2-4)
   - The bible of computer vision geometry
2. **"Computer Vision: A Modern Approach" by Forsyth & Ponce** (Chapters 1-3)

Online Courses

1. **Introduction to Computer Vision (Udacity CS373)**
   - Georgia Tech course, focus on geometry modules
2. **First Principles of Computer Vision (Columbia)** - YouTube
   - Excellent geometric intuition

Specialized Resources

- **OpenCV Tutorials**: Camera calibration and 3D reconstruction
- **Caltech Vision Course**: Lectures on projective geometry

Week 2 Schedule (20 hours)

Day 1-2: 2D/3D Transformations (6 hours)

- Theory: Rotation, translation, scaling matrices
- Practice: Implement 2D image transformations
- Code: Use OpenCV for affine transformations

Day 3-4: Projective Geometry (6 hours)

- Theory: Homogeneous coordinates, projective transformations
- Practice: Implement homography estimation
- Resources: Hartley & Zisserman Ch. 2

Day 5-6: Camera Models (8 hours)

- Theory: Pinhole camera, intrinsic/extrinsic parameters
- Practice: Camera calibration with OpenCV
- Project: Calibrate your phone camera

Programming Assignments

```python
# Assignment 1: 2D transformations and image warping
# Assignment 2: Homography estimation using RANSAC
# Assignment 3: Camera calibration pipeline
```

Module 3: Signal Processing Fundamentals (Week 3)

Primary Textbooks

1. **"Digital Signal Processing" by Oppenheim & Schafer** (Chapters 1-4, 7-8)
2. **"Digital Image Processing" by Gonzalez & Woods** (Chapters 3-5)

Online Courses

1. **Signals and Systems (MIT 6.003)** - MIT OpenCourseWare
   - Focus on Fourier analysis modules
2. **Digital Signal Processing (Coursera - École Polytechnique Fédérale de Lausanne)**

Practical Resources

- **SciPy Signal Processing**: Documentation and tutorials
- **PyImageSearch**: Practical image processing tutorials

Week 3 Schedule (20 hours)

Day 1-2: Fourier Transforms (6 hours)

- Theory: DFT, FFT, frequency domain analysis
- Practice: Implement FFT-based filtering
- Resources: MIT 6.003 Lectures on Fourier analysis

Day 3-4: Convolution and Filtering (6 hours)

- Theory: Convolution theorem, filter design
- Practice: Implement various image filters
- Code: Use scipy.signal for filter design

Day 5-6: Sampling and Reconstruction (8 hours)

- Theory: Nyquist theorem, aliasing, interpolation
- Practice: Image resizing and anti-aliasing
- Project: Build a simple image editor

Programming Assignments

```python
# Assignment 1: FFT-based image filtering
# Assignment 2: Custom convolution implementation
# Assignment 3: Multi-scale image processing
```

Concrete Study Materials and Resources

Essential Software Setup

```bash
# Python environment setup
conda create -n cv_math python=3.9
conda activate cv_math
pip install numpy scipy matplotlib opencv-python jupyter
pip install sympy plotly ipywidgets  # for interactive notebooks
```

Recommended Books (Priority Order)

1. **Gilbert Strang - "Linear Algebra and Its Applications"** ($50-80)
2. **Hartley & Zisserman - "Multiple View Geometry"** ($80-120)
3. **Gonzalez & Woods - "Digital Image Processing"** ($60-100)

Free Alternatives

- **Linear Algebra**: MIT 18.06 notes (free PDF)
- **Geometry**: Computer Vision Online textbook by Szeliski (free)
- **Signal Processing**: Think DSP by Allen Downey (free online)

Online Platforms

1. **MIT OpenCourseWare** (Free)

- 18.06 Linear Algebra
- 6.003 Signals and Systems

2. **Coursera** ($39-79/month)
   - Linear Algebra for Machine Learning (Imperial College)
   - Digital Signal Processing (EPFL)
3. **edX** ($50-100 per course)
   - MIT Introduction to Computational Thinking

Interactive Tools

- **Jupyter Notebooks**: For all programming assignments
- **Desmos Graphing Calculator**: For visualizing transformations
- **GeoGebra**: For geometric intuition
- **Wolfram Alpha**: For checking mathematical calculations

Daily Study Routine

Recommended Schedule (20 hours/week)

- **Morning (2 hours)**: Theory reading and note-taking
- **Afternoon (2 hours)**: Video lectures and online content
- **Evening (1 hour)**: Programming assignments and practice

Study Techniques

1. **Active Learning**: Implement concepts immediately in code
2. **Visual Learning**: Draw diagrams for geometric concepts
3. **Spaced Repetition**: Review previous day's material each morning
4. **Project-Based**: Build small projects to reinforce concepts

Assessment and Milestones

Week 1 Checkpoint: Linear Algebra Mastery

- [ ] Implement PCA from scratch
- [ ] Understand geometric meaning of eigenvalues
- [ ] Solve image compression using SVD
- [ ] Quiz: 20 multiple-choice questions on linear algebra

Week 2 Checkpoint: Geometry Proficiency

- [ ] Calibrate a camera using checkerboard pattern

- [ ] Implement homography estimation
- [ ] Understand projective transformations
- [ ] Project: Create a simple augmented reality app

Week 3 Checkpoint: Signal Processing Skills

- [ ] Design and implement custom image filters
- [ ] Understand frequency domain analysis
- [ ] Build a noise reduction algorithm
- [ ] Project: Create a simple photo enhancement tool

Final Assessment

- **Comprehensive Project**: Build a basic image stitching application that combines all three modules
- **Theory Exam**: 50 questions covering all mathematical foundations
- **Code Review**: Submit all programming assignments for peer review

Troubleshooting Common Issues

If You're Struggling with Linear Algebra:

- Start with Khan Academy's Linear Algebra course
- Use 3Blue1Brown for visual intuition
- Practice with smaller matrices first

If Geometry Seems Abstract:

- Use physical objects to understand transformations
- Work with simple 2D examples before 3D
- Implement transformations step-by-step

If Signal Processing is Overwhelming:

- Start with 1D signals before images
- Use audio examples for intuition
- Focus on practical applications first

Next Steps Preparation

- Set up development environment for Phase 2
- Download OpenCV datasets
- Familiarize yourself with computer vision terminology
- Join computer vision communities (Reddit, Discord, Stack Overflow)

Phase 2: Core Computer Vision - Detailed Study Plan

Duration: 6-8 weeks (120-160 hours total)

Module 4: Image Processing Fundamentals (Week 4-5)

Primary Textbooks

1. **"Digital Image Processing" by Gonzalez & Woods** (Chapters 2-5, 9-10)
2. **"Computer Vision: Algorithms and Applications" by Szeliski** (Chapter 3)
3. **"Learning OpenCV 4" by Kaehler & Bradski** (Chapters 5-10)

Online Courses

1. **Computer Vision Basics (Coursera - University at Buffalo)**
   - Focus on image processing modules
2. **PyImageSearch University** (Paid but comprehensive)
   - Practical computer vision with OpenCV

Week 4-5 Schedule (40 hours)

Week 4: Basic Image Operations

- **Day 1-2: Image Representation (6 hours)**
  - Theory: Pixels, color spaces (RGB, HSV, LAB), bit depth
  - Practice: Convert between color spaces, analyze histograms
  - Code: Implement color space conversions from scratch
  - Resources: Gonzalez Ch. 2, OpenCV color space tutorial
- **Day 3-4: Histogram Processing (6 hours)**
  - Theory: Histogram equalization, specification, local enhancement
  - Practice: Build automatic contrast enhancement
  - Project: Create HDR-like effect using histogram manipulation
  - Resources: Gonzalez Ch. 3, implement CLAHE algorithm
- **Day 5-7: Spatial Filtering (8 hours)**
  - Theory: Linear/non-linear filtering, convolution masks
  - Practice: Implement smoothing, sharpening, edge-preserving filters
  - Code: Build bilateral filter from scratch
  - Resources: Szeliski Ch. 3.3, OpenCV filtering tutorial

Week 5: Advanced Processing

- **Day 1-2: Morphological Operations (6 hours)**
  - Theory: Erosion, dilation, opening, closing, morphological reconstruction
  - Practice: Noise removal, shape analysis, text processing
  - Project: License plate character segmentation
  - Resources: Gonzalez Ch. 9, implement morphological operations
- **Day 3-4: Edge Detection (8 hours)**
  - Theory: Gradient operators, Laplacian, Canny edge detector
  - Practice: Implement Sobel, Prewitt, Canny from scratch
  - Project: Automatic document scanner (edge detection + perspective correction)
  - Resources: Detailed Canny implementation tutorial
- **Day 5-7: Corner Detection (6 hours)**
  - Theory: Harris corner detector, FAST, Shi-Tomasi
  - Practice: Feature point detection and matching
  - Code: Implement Harris corner detector
  - Resources: Multiple View Geometry Ch. 4

Programming Assignments

```python
# Week 4 Assignments:
# 1. Custom histogram equalization implementation
# 2. Multi-scale edge-preserving smoothing
# 3. Real-time color space converter

# Week 5 Assignments:
# 1. Complete Canny edge detector from scratch
# 2. Document scanner with perspective correction
# 3. Corner detection comparison study
```

Module 5: Feature Extraction and Description (Week 6)

Primary Resources

1. **"Computer Vision: Algorithms and Applications" by Szeliski** (Chapter 4)
2. **Original SIFT Paper** by David Lowe (2004)
3. **OpenCV Feature Detection Tutorials**

Week 6 Schedule (20 hours)

Day 1-2: Scale-Invariant Features (6 hours)

- Theory: Scale-space theory, Difference of Gaussians, SIFT algorithm
- Practice: Implement simplified SIFT descriptor
- Resources: Lowe's original paper, detailed SIFT tutorial

Day 3-4: Speed-Optimized Features (6 hours)

- Theory: SURF, ORB, BRIEF descriptors
- Practice: Compare feature detectors on various images
- Project: Real-time feature matching application

Day 5-7: Local Binary Patterns and HOG (8 hours)

- Theory: LBP for texture analysis, HOG for object detection
- Practice: Implement pedestrian detection using HOG+SVM
- Code: Build texture classification system

Programming Assignments

```python
# 1. SIFT keypoint detector implementation
# 2. Feature matching with RANSAC
# 3. Real-time ORB feature tracking
```

Module 6: Image Matching and Registration (Week 7)

Primary Resources

1. **"Multiple View Geometry" by Hartley & Zisserman** (Chapters 4-5)
2. **"Computer Vision: Algorithms and Applications" by Szeliski** (Chapter 6)

Week 7 Schedule (20 hours)

Day 1-2: Feature Matching (6 hours)

- Theory: Nearest neighbor matching, ratio test, cross-checking
- Practice: Robust feature matching pipeline
- Code: Implement FLANN-based matcher

Day 3-4: Geometric Verification (6 hours)

- Theory: RANSAC algorithm, fundamental matrix estimation
- Practice: Outlier rejection in feature matching
- Project: Automatic image stitching

Day 5-7: Optical Flow (8 hours)

- Theory: Lucas-Kanade method, Horn-Schunck method
- Practice: Object tracking using optical flow
- Code: Implement sparse optical flow tracker

Programming Assignments

```python
# 1. Robust homography estimation
# 2. Lucas-Kanade optical flow tracker
# 3. Multi-image panorama stitching
```

Module 7: Camera Geometry and 3D Vision (Week 8-9)

Primary Resources

1. **"Multiple View Geometry" by Hartley & Zisserman** (Chapters 6-12)
2. **"An Invitation to 3-D Vision" by Ma, Soatto, Kosecka, Sastry**
3. **OpenCV 3D Reconstruction Tutorials**

Week 8-9 Schedule (40 hours)

Week 8: Stereo Vision

- **Day 1-3: Camera Calibration (9 hours)**
  - Theory: Intrinsic/extrinsic parameters, distortion models
  - Practice: Multi-camera calibration system
  - Project: Build 3D scanner using stereo cameras
  - Code: Implement Zhang's calibration method
- **Day 4-7: Stereo Matching (11 hours)**
  - Theory: Epipolar geometry, disparity estimation, stereo algorithms
  - Practice: Dense stereo reconstruction
  - Code: Implement block matching and semi-global matching

Week 9: Structure from Motion

- **Day 1-3: Two-View Geometry (9 hours)**
  - Theory: Essential matrix, fundamental matrix, triangulation
  - Practice: 3D point reconstruction from two views
  - Code: Implement eight-point algorithm

- **Day 4-7: Multi-View Reconstruction (11 hours)**
  - Theory: Bundle adjustment, sequential SfM, global SfM
  - Practice: Build complete SfM pipeline
  - Project: 3D model reconstruction from photos

Programming Assignments

```python
# Week 8:
# 1. Complete camera calibration toolbox
# 2. Real-time stereo depth estimation
# 3. 3D point cloud visualization

# Week 9:
# 1. Two-view structure from motion
# 2. Multi-view triangulation
# 3. Bundle adjustment implementation
```

Phase 3: Machine Learning for Computer Vision - Detailed Study Plan

Duration: 4-5 weeks (80-100 hours total)

Module 8: Classical Machine Learning (Week 10-11)

Primary Resources

1. **"Pattern Recognition and Machine Learning" by Bishop** (Chapters 3, 4, 7, 9)
2. **"The Elements of Statistical Learning" by Hastie, Tibshirani, Friedman**
3. **scikit-learn Documentation and Tutorials**

Week 10-11 Schedule (40 hours)

Week 10: Classification and Clustering

- **Day 1-2: Support Vector Machines (6 hours)**
  - Theory: SVM optimization, kernel trick, multi-class SVM
  - Practice: Image classification with HOG+SVM
  - Code: Implement simple SVM from scratch
  - Project: Handwritten digit recognition
- **Day 3-4: Decision Trees and Ensemble Methods (6 hours)**
  - Theory: Decision trees, random forests, boosting

    - Practice: Feature selection for image classification
    - Code: Build random forest for texture classification
- **Day 5-7: Clustering Algorithms (8 hours)**
    - Theory: K-means, hierarchical clustering, DBSCAN
    - Practice: Image segmentation using clustering
    - Project: Automatic color palette extraction

Week 11: Advanced Topics

- **Day 1-3: Dimensionality Reduction (9 hours)**
    - Theory: PCA, LDA, t-SNE, manifold learning
    - Practice: Eigenfaces for face recognition
    - Code: Implement kernel PCA
- **Day 4-7: Expectation-Maximization (11 hours)**
    - Theory: EM algorithm, Gaussian mixture models
    - Practice: Background subtraction using GMM
    - Project: Automatic object segmentation

Programming Assignments

```python
# Week 10:
# 1. HOG+SVM pedestrian detector
# 2. Random forest for scene classification
# 3. K-means image segmentation tool

# Week 11:
# 1. Eigenfaces implementation
# 2. GMM background subtraction
# 3. t-SNE visualization of image features
```

Module 9: Deep Learning Foundations (Week 12-13)

Primary Resources

1. **"Deep Learning" by Goodfellow, Bengio, Courville** (Chapters 6-9)
2. **"Hands-On Machine Learning" by Aurélien Géron** (Chapters 10-15)
3. **CS231n Stanford Course** (Online lectures and assignments)

Week 12-13 Schedule (40 hours)

Week 12: Neural Network Fundamentals

- **Day 1-2: Perceptron to Multilayer Networks (6 hours)**
  - Theory: Perceptron, MLP, universal approximation theorem
  - Practice: Implement neural network from scratch (NumPy only)
  - Code: Build XOR classifier with hidden layers
- **Day 3-4: Backpropagation Algorithm (6 hours)**
  - Theory: Chain rule, gradient computation, computational graphs
  - Practice: Implement backpropagation from scratch
  - Debug: Common backpropagation mistakes and fixes
- **Day 5-7: Optimization and Regularization (8 hours)**
  - Theory: SGD, Adam, RMSprop, dropout, batch normalization
  - Practice: Compare optimizers on image classification
  - Code: Implement different optimization algorithms

Week 13: Deep Learning for Images

- **Day 1-3: Introduction to CNNs (9 hours)**
  - Theory: Convolution operation, pooling, CNN architecture design
  - Practice: Build simple CNN for CIFAR-10
  - Code: Implement convolution and pooling layers
- **Day 4-7: Transfer Learning and Data Augmentation (11 hours)**
  - Theory: Feature extraction vs fine-tuning, data augmentation strategies
  - Practice: Transfer learning with pre-trained networks
  - Project: Custom image classifier with limited data

Programming Assignments

```python
# Week 12:
# 1. Neural network from scratch (NumPy)
# 2. Gradient checking implementation
# 3. Optimizer comparison study

# Week 13:
# 1. CNN implementation from scratch
# 2. Transfer learning pipeline
# 3. Data augmentation library
```

Phase 4: Deep Computer Vision - Detailed Study Plan

Duration: 8-10 weeks (160-200 hours total)

Module 10: Convolutional Neural Networks (Week 14-15)

Primary Resources

1. **CS231n Stanford Course** (Full course + assignments)
2. **"Deep Learning for Computer Vision" by Raschka & Mirjalili**
3. **PyTorch/TensorFlow Official Tutorials**

Week 14-15 Schedule (40 hours)

Week 14: CNN Architectures

- **Day 1-2: Classic Architectures (6 hours)**
  - Theory: LeNet, AlexNet, VGG architecture principles
  - Practice: Implement and train classic architectures
  - Compare: Performance analysis on CIFAR-10
- **Day 3-4: Residual Networks (6 hours)**
  - Theory: Residual connections, identity mapping, ResNet variants
  - Practice: Build ResNet from scratch in PyTorch
  - Project: Compare ResNet depths on image classification
- **Day 5-7: Modern Architectures (8 hours)**
  - Theory: DenseNet, EfficientNet, RegNet design principles
  - Practice: Architecture search and scaling laws
  - Code: Implement EfficientNet building blocks

Week 15: CNN Analysis and Visualization

- **Day 1-3: CNN Visualization (9 hours)**
  - Theory: Activation maximization, gradient-based methods, CAM/Grad-CAM
  - Practice: Visualize what CNNs learn at different layers
  - Tools: Use Captum for model interpretability
- **Day 4-7: Advanced Training Techniques (11 hours)**
  - Theory: Learning rate scheduling, progressive resizing, mixup
  - Practice: State-of-the-art training recipes
  - Project: Achieve high accuracy on ImageNet subset

Programming Assignments

```python
# Week 14:
# 1. ResNet implementation and ablation study
# 2. Architecture comparison framework
# 3. Custom CNN architecture design

# Week 15:
# 1. CNN visualization toolkit
# 2. Advanced training pipeline
# 3. Model interpretability analysis
```

Module 11: Object Detection (Week 16-17)

Primary Resources

1. **"Deep Learning for Computer Vision" by Raschka & Mirjalili** (Object Detection chapters)
2. **Original Papers**: R-CNN, Fast R-CNN, Faster R-CNN, YOLO, SSD
3. **Detectron2 Documentation and Tutorials**

Week 16-17 Schedule (40 hours)

Week 16: Two-Stage Detectors

- **Day 1-2: R-CNN Family (6 hours)**
  - Theory: Region proposals, R-CNN, Fast R-CNN evolution
  - Practice: Implement Fast R-CNN training pipeline
  - Code: Build RoI pooling layer
- **Day 3-4: Faster R-CNN (6 hours)**
  - Theory: Region Proposal Network, anchor generation
  - Practice: Train Faster R-CNN on custom dataset
  - Debug: Common training issues and solutions
- **Day 5-7: Advanced Two-Stage Methods (8 hours)**
  - Theory: FPN, Mask R-CNN, Cascade R-CNN
  - Practice: Multi-scale object detection
  - Project: Custom object detection system

Week 17: Single-Stage Detectors

- **Day 1-3: YOLO Family (9 hours)**
  - Theory: YOLO v1-v5 evolution, anchor-free detection

    - Practice: Implement YOLO v3 from scratch
    - Deploy: Real-time detection application
- **Day 4-7: Modern Single-Stage Detectors (11 hours)**
    - Theory: SSD, RetinaNet, FCOS, DETR
    - Practice: Focal loss implementation and training
    - Project: Comparison of detection frameworks

Programming Assignments

```python
# Week 16:
# 1. Fast R-CNN implementation
# 2. Custom dataset annotation and training
# 3. Multi-scale detection evaluation

# Week 17:
# 1. YOLO v3 from scratch
# 2. Real-time detection application
# 3. Detection framework comparison
```

Module 12: Semantic and Instance Segmentation (Week 18)

Primary Resources

1. **Original Papers**: FCN, U-Net, DeepLab series, Mask R-CNN
2. **Segmentation Models PyTorch Library**
3. **MMSegmentation Framework**

Week 18 Schedule (20 hours)

Day 1-2: Semantic Segmentation (6 hours)

- Theory: FCN, U-Net, encoder-decoder architectures
- Practice: Medical image segmentation with U-Net
- Code: Implement FCN with skip connections

Day 3-4: Advanced Semantic Segmentation (6 hours)

- Theory: DeepLab series, dilated convolutions, ASPP
- Practice: Cityscapes dataset segmentation
- Code: Implement dilated convolutions

Day 5-7: Instance Segmentation (8 hours)

- Theory: Mask R-CNN, panoptic segmentation
- Practice: Instance segmentation on COCO dataset
- Project: Real-time person segmentation for video calls

Programming Assignments

```python
# 1. U-Net for medical image segmentation
# 2. DeepLab v3+ implementation
# 3. Real-time segmentation application
```

Module 13: Advanced CNN Applications (Week 19)

Week 19 Schedule (20 hours)

Day 1-2: Face Recognition (6 hours)

- Theory: FaceNet, ArcFace, face verification vs identification
- Practice: Build face recognition system
- Code: Implement triplet loss

Day 3-4: Human Pose Estimation (6 hours)

- Theory: OpenPose, PoseNet, bottom-up vs top-down approaches
- Practice: Real-time pose estimation
- Project: Fitness app with pose correction

Day 5-7: Image Generation (8 hours)

- Theory: VAE, GAN basics, style transfer
- Practice: Generate new images with VAE
- Code: Implement neural style transfer

Phase 5: Modern Computer Vision - Detailed Study Plan

Duration: 6-8 weeks (120-160 hours total)

Module 14: Vision Transformers and Attention (Week 20-21)

Primary Resources

1. **"Attention Is All You Need"** - Original Transformer paper

2. **"An Image is Worth 16x16 Words"** - Vision Transformer paper
3. **Hugging Face Transformers Documentation**

Week 20-21 Schedule (40 hours)

Week 20: Attention Mechanisms

- **Day 1-3: Self-Attention and Transformers (9 hours)**
  - Theory: Attention mechanism, multi-head attention, positional encoding
  - Practice: Implement transformer block from scratch
  - Code: Build text classification transformer
- **Day 4-7: Vision Transformers (11 hours)**
  - Theory: ViT architecture, patch embedding, classification token
  - Practice: Train ViT on image classification
  - Compare: ViT vs CNN performance analysis

Week 21: Advanced Transformer Architectures

- **Day 1-3: Hierarchical Vision Transformers (9 hours)**
  - Theory: Swin Transformer, shifted window attention
  - Practice: Implement Swin Transformer blocks
  - Project: Object detection with Swin Transformer
- **Day 4-7: Detection Transformers (11 hours)**
  - Theory: DETR, object queries, bipartite matching
  - Practice: Train DETR on custom dataset
  - Code: Implement Hungarian algorithm for matching

Programming Assignments

```python
# Week 20:
# 1. Transformer from scratch (PyTorch)
# 2. Vision Transformer implementation
# 3. ViT vs CNN comparison study

# Week 21:
# 1. Swin Transformer implementation
# 2. DETR object detection
# 3. Attention visualization tools
```

Module 15: 3D Computer Vision (Week 22-23)

Primary Resources

1. **"3D Computer Vision: Principles and Applications"** by various authors
2. **Open3D Documentation and Tutorials**
3. **NeRF and 3D Gaussian Splatting Papers**

Week 22-23 Schedule (40 hours)

Week 22: Point Cloud Processing

- **Day 1-3: Point Cloud Basics (9 hours)**
  - Theory: Point cloud representation, PCL operations
  - Practice: Point cloud filtering and segmentation
  - Tools: Open3D for point cloud processing
- **Day 4-7: 3D Object Detection (11 hours)**
  - Theory: PointNet, PointNet++, voxel-based methods
  - Practice: 3D object detection in LIDAR data
  - Project: Autonomous driving perception system

Week 23: Neural Radiance Fields

- **Day 1-4: NeRF Implementation (12 hours)**
  - Theory: Volume rendering, positional encoding, neural fields
  - Practice: Implement basic NeRF from scratch
  - Code: Train NeRF on synthetic scenes
- **Day 5-7: Advanced 3D Methods (8 hours)**
  - Theory: 3D Gaussian Splatting, Instant NGP
  - Practice: Real-time neural rendering
  - Project: 3D scene reconstruction from phone videos

Programming Assignments

```python
# Week 22:
# 1. Point cloud processing pipeline
# 2. PointNet implementation
# 3. 3D object detection system

# Week 23:
# 1. NeRF from scratch
# 2. 3D Gaussian Splatting
# 3. Real-time 3D reconstruction
```

Module 16: Video Analysis (Week 24-25)

Primary Resources

1. **Video Understanding Papers**: I3D, SlowFast, Video Swin Transformer
2. **MMAction2 Framework**
3. **PyTorchVideo Library**

Week 24-25 Schedule (40 hours)

Week 24: Action Recognition

- **Day 1-3: 3D CNNs (9 hours)**
  - Theory: 3D convolutions, C3D, I3D architectures
  - Practice: Action recognition on UCF-101
  - Code: Implement 3D ResNet
- **Day 4-7: Two-Stream Networks (11 hours)**
  - Theory: RGB and optical flow streams, temporal modeling
  - Practice: Build two-stream action recognition
  - Project: Real-time activity recognition

Week 25: Video Object Detection and Tracking

- **Day 1-4: Video Object Detection (12 hours)**
  - Theory: Temporal consistency, video object detection methods
  - Practice: Extend image detectors to video
  - Code: Implement temporal feature aggregation
- **Day 5-7: Multi-Object Tracking (8 hours)**
  - Theory: DeepSORT, FairMOT, tracking by detection
  - Practice: Multi-person tracking system

- Project: Sports analytics application

Phase 6: Specialized Applications - Detailed Study Plan

Duration: 4-5 weeks (80-100 hours total)

Module 17: Real-time Computer Vision (Week 26-27)

Primary Resources

1. **TensorRT Developer Guide**
2. **ONNX Documentation**
3. **OpenVINO Toolkit Tutorials**
4. **Mobile AI Frameworks Documentation**

Week 26-27 Schedule (40 hours)

Week 26: Model Optimization

- **Day 1-2: Quantization Techniques (6 hours)**
  - Theory: Post-training quantization, quantization-aware training
  - Practice: Quantize models for INT8 inference
  - Tools: TensorRT, PyTorch quantization
- **Day 3-4: Model Pruning and Distillation (6 hours)**
  - Theory: Structured/unstructured pruning, knowledge distillation
  - Practice: Compress large models for edge deployment
  - Code: Implement magnitude-based pruning
- **Day 5-7: Hardware Acceleration (8 hours)**
  - Theory: GPU optimization, TensorRT, CUDA kernels
  - Practice: Optimize inference pipeline
  - Benchmark: Compare different optimization techniques

Week 27: Mobile and Edge Deployment

- **Day 1-3: Mobile Frameworks (9 hours)**
  - Theory: TensorFlow Lite, CoreML, ONNX Runtime
  - Practice: Deploy model to mobile device
  - Project: Real-time mobile object detection app
- **Day 4-7: Edge Computing (11 hours)**
  - Theory: Edge TPU, Intel Movidius, Jetson Nano deployment

  - Practice: Set up edge inference pipeline
  - Project: Smart camera system with local processing

Programming Assignments

```python
# Week 26:
# 1. Model quantization pipeline
# 2. Knowledge distillation framework
# 3. TensorRT optimization benchmark

# Week 27:
# 1. Mobile app with ML inference
# 2. Edge deployment pipeline
# 3. Real-time performance optimization
```

Module 18: Domain-Specific Applications (Week 28-29)

Week 28-29 Schedule (40 hours)

Week 28: Autonomous Driving

- **Day 1-2: Lane Detection (6 hours)**
  - Theory: Traditional methods, deep learning approaches
  - Practice: Build lane detection system
  - Dataset: Work with CULane dataset
- **Day 3-4: Traffic Sign Recognition (6 hours)**
  - Theory: Classification and detection approaches
  - Practice: German Traffic Sign Recognition Benchmark
  - Project: End-to-end traffic sign system
- **Day 5-7: 3D Object Detection for Autonomous Driving (8 hours)**
  - Theory: LIDAR-camera fusion, BEV representations
  - Practice: 3D detection on KITTI dataset
  - Code: Implement PointPillars

Week 29: Medical Imaging and Industrial Applications

- **Day 1-3: Medical Image Analysis (9 hours)**
  - Theory: X-ray, CT, MRI analysis techniques
  - Practice: Chest X-ray abnormality detection
  - Ethics: Medical AI bias and fairness considerations

- **Day 4-7: Industrial Inspection (11 hours)**
  - Theory: Defect detection, quality control systems
  - Practice: Surface defect detection
  - Project: Automated inspection system

Module 19: Ethics and Bias in Computer Vision (Week 30)

Week 30 Schedule (20 hours)

Day 1-2: Dataset Bias and Fairness (6 hours)

- Theory: Bias sources, demographic parity, equalized odds
- Practice: Analyze bias in face recognition systems
- Tools: Fairness evaluation frameworks

Day 3-4: Privacy and Security (6 hours)

- Theory: Differential privacy, adversarial attacks, federated learning
- Practice: Implement privacy-preserving techniques
- Code: Adversarial example generation

Day 5-7: Responsible AI Development (8 hours)

- Theory: AI ethics frameworks, regulatory landscape
- Practice: Develop ethical guidelines for CV systems
- Project: Bias mitigation in real-world application

Phase 7: Capstone Projects - Detailed Study Plan

Duration: 3-4 weeks (60-80 hours total)

Project Development Framework (Week 31-34)

Project Options and Resources

Option 1: End-to-End Object Detection System

Week 31-32: Development (40 hours)

- Data collection and annotation (Roboflow, LabelBox)
- Model training and optimization
- Deployment pipeline setup
- Performance monitoring system

Week 33-34: Polish and Documentation (20 hours)

- Code review and refactoring
- Documentation and user guides
- Performance benchmarking
- Demo preparation

Option 2: 3D Reconstruction Application

Week 31-32: Core Implementation (40 hours)

- Multi-view stereo or SfM pipeline
- Point cloud processing and mesh generation
- Web interface for 3D visualization
- Real-time reconstruction optimization

Option 3: Real-time Video Analytics

Week 31-32: System Development (40 hours)

- Multi-stream video processing
- Real-time inference optimization
- Alert and notification system
- Dashboard and analytics interface

Option 4: Medical Image Analysis Tool

Week 31-32: Clinical Application (40 hours)

- Medical dataset processing
- Diagnostic model development
- Uncertainty quantification
- Clinical validation framework

Option 5: Custom Research Project

Week 31-32: Research Implementation (40 hours)

- Literature review and baseline implementation
- Novel method development
- Experimental validation
- Research paper writing

Assessment Criteria

- **Technical Implementation (40%)**
- **Innovation and Creativity (20%)**
- **Documentation and Presentation (20%)**
- **Real-world Impact (20%)**

Final Deliverables

- Complete source code with documentation
- Technical report or research paper
- Live demonstration
- Deployment guide
- Future work recommendations

Resource Summary and Cost Breakdown

Essential Software (Free)

- Python, PyTorch/TensorFlow, OpenCV, scikit-learn
- Jupyter notebooks, Git, Docker
- Cloud platforms (free tiers): Google Colab, Kaggle

Recommended Books ($300-500 total)

1. Hartley & Zisserman - Multiple View Geometry ($120)
2. Szeliski - Computer Vision: Algorithms and Applications (Free online)
3. Gonzalez & Woods - Digital Image Processing ($100)
4. Goodfellow et al. - Deep Learning (Free online)
5. Bishop - Pattern Recognition and ML ($80)

Online Courses ($200-400 total)

- CS231n Stanford (Free)
- Coursera Computer Vision courses ($39/month × 3 months)
- PyImageSearch University (Optional, $200)

Hardware Requirements

- GPU: RTX 3060 or better ($300-800)
- RAM: 16GB minimum, 32GB recommended

- Storage: 1TB SSD for datasets

Total Estimated Cost

- **Minimum**: $500-800 (used GPU, free resources)
- **Recommended**: $1000-1500 (new GPU, paid courses)
- **Premium**: $2000+ (workstation-grade hardware)

Time Investment

- **Total Duration**: 30-34 weeks (7-8.5 months)
- **Weekly Commitment**: 20-25 hours
- **Total Hours**: 600-850 hours
- **Flexible Timeline**: Can extend to 12 months for part-time study