Computer Vision Final Year Project Ideas PDF

A good computer vision final year project goes beyond image classification to detection, tracking, segmentation or OCR, runs on real video, and reports standard metrics such as mAP or IoU. Traffic sign detection, helmet detection, number plate reading and attendance by face recognition are practical choices with public data or easy data collection.

Key takeaways

- Use a pretrained detector (a YOLO model or Faster R-CNN) and fine-tune it on your data.
- Report mAP for detection and IoU or Dice for segmentation, not only accuracy.
- Test on your own video from the campus or street, with permission where people appear.
- Face projects need consent and careful data handling.
- The PDF gives plans and steps, not finished source code.

Who this PDF is for

This PDF is for final-year B.Tech, MCA and M.Tech students who want a computer vision project with a live camera or video demo. The ideas use OpenCV, PyTorch and common detection models. Each plan names a public dataset or tells you how to collect your own, and lists modules a team can split: data and labelling, model training, tracking or post-processing, and the app.

How to choose your project

- **Detection, segmentation or tracking** sounds more final-year than plain classification.
- **Check labels.** Bounding boxes take time; prefer datasets that already have them.
- **Plan hardware.** A laptop webcam or a phone video is enough for most demos.
- **Think about privacy** if faces or number plates appear.

Standard level projects

1. Traffic Sign Detection and Recognition

- **Problem:** Find and read traffic signs in road images.
- **Dataset:** German Traffic Sign Recognition Benchmark (GTSRB) and its detection set (GTSDB).
- **Tech stack:** Python, PyTorch, a YOLO model, OpenCV.
- **Main modules:** Detector, classifier, video pipeline, results viewer.
- **Steps:** Train a classifier on GTSRB, a detector on GTSDB, chain them on dashcam-style video.

- **Show in the viva:** mAP of the detector and examples of missed signs.
- **Extensions:** Test on Indian road video you record yourself.

2. Helmet Detection for Two-Wheeler Riders

- **Problem:** Detect riders without helmets in traffic video.
- **Dataset:** Collect and label your own frames (with permission), or use a public helmet dataset you can verify on Kaggle or Roboflow Universe.
- **Tech stack:** Python, a YOLO model, OpenCV, LabelImg or CVAT.
- **Main modules:** Labelling, detector training, rider-helmet association, alert log.
- **Steps:** Label riders and helmets, fine-tune YOLO, link each helmet to a rider box, log violations.
- **Show in the viva:** How you linked helmets to riders.
- **Extensions:** Add number plate reading for flagged riders.

3. Face Recognition Attendance System

- **Problem:** Mark class attendance by recognising students' faces.
- **Dataset:** Your own consented photos of classmates.
- **Tech stack:** Python, OpenCV, face embedding library, SQLite, Flask.
- **Main modules:** Enrolment, face detection, embedding match, attendance database, report export.
- **Steps:** Enrol each student with a few photos, detect faces in class photos, match embeddings, save attendance.
- **Show in the viva:** Your consent process and false-match rate.
- **Extensions:** Add liveness detection against photo spoofing.

4. Face Mask Detection in Crowds

- **Problem:** Detect faces with, without, or incorrectly worn masks.
- **Dataset:** Kaggle "Face Mask Detection" (andrewmvd, with bounding boxes).
- **Tech stack:** Python, PyTorch, a YOLO model.
- **Main modules:** Annotation conversion, detector training, per-class metrics, video demo.
- **Steps:** Convert the XML labels to YOLO format, fine-tune, check per-class mAP.
- **Show in the viva:** Why the "incorrect" class is hardest.
- **Extensions:** Count people per frame for crowd monitoring.

Advanced level projects

5. Vehicle Counting and Speed Estimation

- **Problem:** Count vehicles and estimate speed from a fixed camera.
- **Dataset:** Your own recorded road video, or public traffic videos with a clear licence.
- **Tech stack:** Python, a YOLO model, a tracker such as SORT or ByteTrack, OpenCV.
- **Main modules:** Detection, tracking, line-crossing counter, speed from pixel distance, dashboard.
- **Steps:** Detect vehicles, assign track IDs, count crossings, calibrate pixels to metres, estimate speed.

- **Show in the viva:** How calibration affects speed error.
- **Extensions:** Add vehicle type classification.

6. Automatic Number Plate Recognition

- **Problem:** Read number plates from vehicle images.
- **Dataset:** Collect your own images in a parking area with permission; label plates.
- **Tech stack:** Python, a YOLO model, an OCR engine such as Tesseract or EasyOCR.
- **Main modules:** Plate detector, crop and clean, OCR, format check, log.
- **Steps:** Detect plates, deskew and threshold, run OCR, validate against the Indian plate format.
- **Show in the viva:** Character-level error rate and failure cases.
- **Extensions:** Add a gate-access whitelist.

7. Human Pose Estimation for Exercise Counting

- **Problem:** Count push-ups or squats and check form from video.
- **Dataset:** MPII Human Pose for testing; your own exercise videos for the demo.
- **Tech stack:** Python, MediaPipe or OpenPose, OpenCV.
- **Main modules:** Keypoint extraction, joint angle calculation, repetition counter, feedback overlay.
- **Steps:** Extract keypoints per frame, compute joint angles, count reps with thresholds, show tips.
- **Show in the viva:** The angle rules used for each exercise.
- **Extensions:** Add more exercises and a progress log.

8. Industrial Defect Detection

- **Problem:** Find scratches, cracks and dents on products.
- **Dataset:** MVTec AD (anomaly detection) dataset.
- **Tech stack:** Python, PyTorch, anomaly detection methods.
- **Main modules:** Normal-only training, anomaly scoring, heatmaps, threshold selection.
- **Steps:** Train on good samples only, score test images, produce heatmaps, report AUROC.
- **Show in the viva:** Why the model trains without defect images.
- **Extensions:** Run on a Raspberry Pi with a camera.

Research-oriented projects

9. Semantic Segmentation of Road Scenes

- **Problem:** Label each pixel as road, car, person and so on.
- **Dataset:** Cityscapes (registration needed) or CamVid.
- **Tech stack:** Python, PyTorch, U-Net or DeepLabV3.
- **Main modules:** Data pipeline, segmentation model, IoU per class, colour overlay video.
- **Steps:** Train a pretrained DeepLabV3, report mean IoU, run on your own road video.
- **Show in the viva:** Per-class IoU and hardest classes.
- **Extensions:** Compare a lightweight model for speed.

10. Object Detection Benchmark: YOLO vs Faster R-CNN

- **Problem:** Compare a one-stage and a two-stage detector on the same data.

- **Dataset:** Pascal VOC 2012.
- **Tech stack:** Python, PyTorch, torchvision.
- **Main modules:** Shared data split, two training pipelines, mAP and FPS benchmarks.
- **Steps:** Train both on the same split, compute mAP at the same IoU, measure frames per second.
- **Show in the viva:** The speed versus accuracy table.
- **Extensions:** Add a test on low-light images.

11. Low-light Image Enhancement

- **Problem:** Make dark photos clearer before detection.
- **Dataset:** LOL (Low-Light) dataset.
- **Tech stack:** Python, PyTorch, OpenCV.
- **Main modules:** Classic baseline (CLAHE), learned model, PSNR and SSIM, detection-after-enhancement test.
- **Steps:** Run CLAHE, train or fine-tune an enhancement network, measure PSNR/SSIM and detector gain.
- **Show in the viva:** Whether enhancement helps detection, with numbers.
- **Extensions:** Apply to night CCTV footage.

12. Document Layout and Table Extraction

- **Problem:** Find text blocks, tables and figures in scanned pages.
- **Dataset:** PubLayNet dataset.
- **Tech stack:** Python, PyTorch, a layout model, Tesseract OCR.
- **Main modules:** Layout detection, table cropping, OCR, export to CSV.
- **Steps:** Fine-tune a detector on PubLayNet, crop tables, OCR cells, write CSV.
- **Show in the viva:** Example pages and extraction errors.
- **Extensions:** Support Hindi text with an Indic OCR model.

Report structure

A final-year report is marked on clarity as much as on code. A structure most universities accept:

1. **Abstract** (150 to 250 words): problem, method, main result.
2. **Introduction:** background, problem statement, objectives and scope.
3. **Literature review:** 8 to 15 papers or existing systems, and the gap your project fills.
4. **Requirements and design:** functional and non-functional requirements, architecture diagram, the camera or video setup, model choice and labelling plan.
5. **Implementation:** modules, tools, key algorithms and screenshots.
6. **Testing and results:** mAP, IoU or error rate on a held-out set, frames per second, and examples of failures.
7. **Conclusion and future scope.**
8. **References** in the format your department asks for (IEEE style is common).

Follow your own university's template if it gives one; it overrides this list.

Common mistakes to avoid

- Testing only on images that look like the training set.
- Recording people or plates without permission.
- Quoting accuracy for detection instead of mAP.
- Skipping FPS measurement for a "real-time" claim.

Frequently asked questions

Which computer vision project is best for a final year?

Detection or tracking projects with a live demo, such as traffic sign detection, helmet detection or vehicle counting. They show more skill than plain image classification.

Do I need to label my own images?

Not always. GTSRB, Pascal VOC, MVTec AD and the Kaggle face mask set come labelled. For India-specific projects you may need to label a few hundred images with CVAT or LabelImg.

Is OpenCV enough for a final year project?

OpenCV is good for video handling and classic methods. Most final-year projects pair it with a deep learning model in PyTorch or TensorFlow.

Can face recognition projects be risky?

Yes, if you collect faces without consent. Take written consent, store data securely, and delete it after evaluation.

Does this PDF include full source code?

No. It gives the problem, data, stack, modules and steps for each project. You write the code yourself, which is what examiners test in the viva.

Is this PDF free?

Yes. Fill in the short form on this page and the PDF downloads right away.

Sources and further reading

- OpenCV documentation
- PyTorch torchvision detection models
- MVTec AD dataset
- Cityscapes dataset
- MediaPipe solutions guide

About this PDF

Prepared by the Uncodemy training team as practice project ideas. Datasets and tools named are public; check each dataset's licence before use. We update it every January. Corrections: info@uncodemy.com.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550