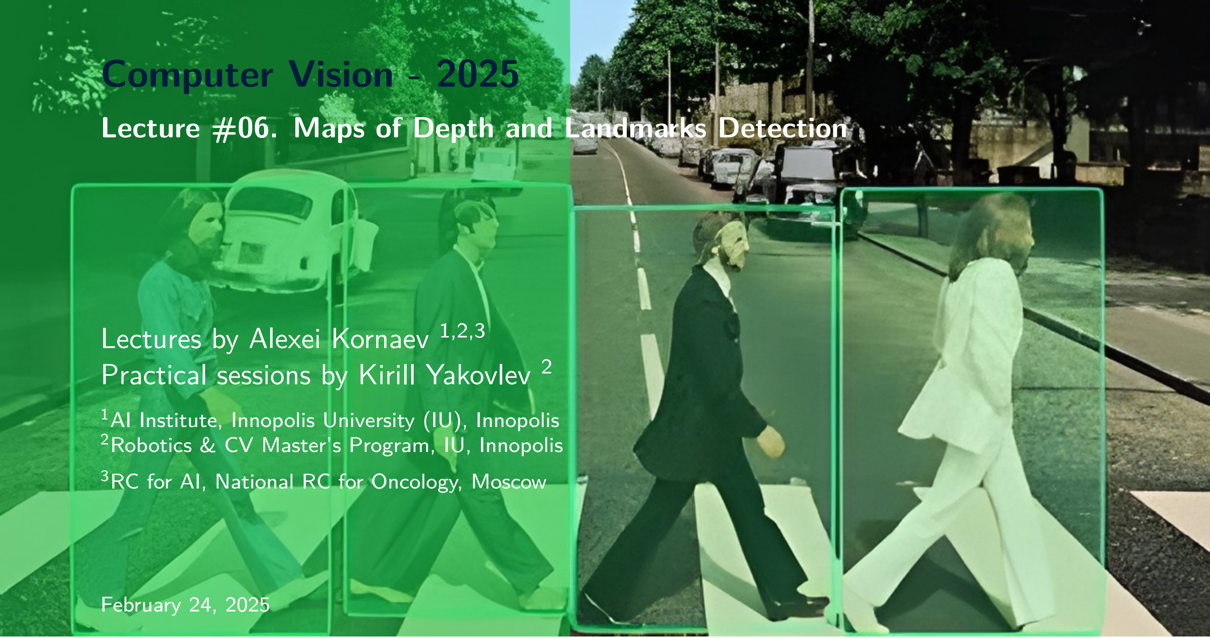

Computer Vision - 2025

Lecture #06. Maps of Depth and Landmarks Detection

Lectures by Alexei Kornaev [1,2,3]
Practical sessions by Kirill Yakovlev [2]

[1] AI Institute, Innopolis University (IU), Innopolis
[2] Robotics & CV Master's Program, IU, Innopolis
[3] RC for AI, National RC for Oncology, Moscow

February 24, 2025

Agenda

1. Outcomes
2. Introduction
   - Monocular Depth Estimation
   - LiDAR-based Depth Estimation
3. Landmark Detection
4. Pose Estimation & Keypoint Detection
5. Practical Implementation
6. Conclusion & Discussion

Section 1. Outcomes

Outcomes

This week's lecture and seminar on Depth Estimation and Landmark Detection aim to provide an understanding of methods for estimating depth from images and detecting keypoints in images. By the end of this week, students will be able to:

1. Understand the principles of depth estimation and landmark detection.
2. Describe techniques such as stereo vision, monocular depth estimation, and LiDAR-based depth sensing.
3. Implement monocular and stereo depth estimation using DL models.
4. Detect facial landmarks and keypoints using pre-trained models like dlib and MediaPipe.
5. Build a real-time depth estimation and landmark detection pipeline.

Key Takeaway: Depth estimation and landmark detection are fundamental for 3D perception in computer vision applications, from autonomous systems to augmented reality.

Section 2. Introduction

What is Depth Estimation?

Depth Estimation Overview

Depth estimation is the process of inferring the distance of objects from a camera using images. It is crucial for robotics, autonomous driving, and AR/VR applications.

- **Stereo Depth Estimation:** Uses two images from slightly different viewpoints to compute disparity and infer depth.
- **Monocular Depth Estimation:** Predicts depth from a single image using learned priors and deep networks.
- **LiDAR-based Depth Sensing:** Uses laser pulses to measure distances with high accuracy.

Figure: Sample images from the 3D Movies dataset.

What is Landmark Detection?

Landmark Detection Overview

Landmark detection involves identifying key points on objects (e.g., human faces, hands, or bodies) for tasks like facial recognition, motion tracking, and medical imaging.

- **Facial Landmark Detection:** Identifies key facial features (e.g., eyes, nose, mouth).
- **Pose Estimation:** Estimates human body keypoints for action recognition and motion tracking.
- **Keypoint Detection in Objects:** Used in robotics and AR applications for shape understanding.

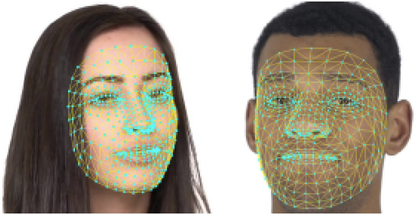

Figure: Facial landmark detection.

Depth Estimation Basics

Definition: Depth estimation is the process of inferring the 3D structure of a scene from 2D images.

Mathematical Formulation

Given a 2D image $I(x, y)$, the goal is to estimate the depth function $D(x, y)$ that maps pixel coordinates to depth values:

$$D(x, y) = f(I(x, y))$$

where f can be a geometric function (stereo vision) or a learned function (monocular depth estimation).

Applications:

- **Robotics:** Obstacle avoidance, SLAM.
- **Self-Driving Cars:** Scene understanding, distance estimation.

Stereo Vision: Basic Concept

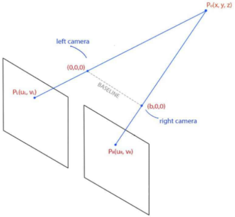

Figure: Image formation using 2 cameras.

Depth from Disparity.

Key Idea: Use two images from slightly different viewpoints to compute depth using disparity.The disparity d is the pixel shift between the left and right images. The depth D is given by:

$$D = \frac{fB}{d}$$

where: f = focal length of the camera, B = baseline distance between cameras, d = disparity (difference in pixel position)

Stereo Vision: Epipolar Geometry

Epipolar Constraint: A point in the left image must lie on the corresponding epipolar line in the right image.

Fundamental Matrix

The relationship between corresponding points in two images is given by:

$$x_r^T F x_l = 0$$

where:

- F is the fundamental matrix.
- x_l, x_r are homogeneous coordinates of corresponding points.

Stereo Rectification: Transforms image pairs so that corresponding points lie on the same scanline, simplifying disparity computation.

Stereo Vision: Depth Estimation Algorithms

Traditional Methods:

- **Block Matching (BM):** Matches small image patches across stereo images.
- **Semi-Global Matching (SGM):** Uses dynamic programming to smooth disparities.

Deep Learning Approaches:

- **Deep Stereo Matching Networks (e.g., PSMNet, RAFT-Stereo)**
- **Self-Supervised Depth Estimation (e.g., Monodepth2)**

Why is Monocular Depth Estimation Hard?

Ambiguity in Depth. Given an image $I(x, y)$, multiple 3D scenes $S_1, S_2, \ldots$ can generate the same 2D projection.

Traditional Cues:

- Perspective cues (vanishing points).
- Texture gradients (closer objects have more detail).
- Shape-from-shading.

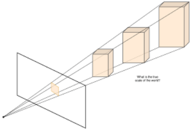

Figure: Depth ambiguity in monocular images. A single image does not contain explicit depth information.

Deep Learning-Based Monocular Depth Estimation

Key Idea: Train deep networks to predict depth from a single image using learned priors.

Learning-Based Depth Prediction

Given an image I, a deep model predicts $D = f(I)$.

Popular Architectures:

- **MiDaS:** Transformer-based depth estimation.
- **DPT:** Depth Prediction Transformer.

How LiDAR Works

Key Idea: LiDAR uses laser pulses to measure distances based on time-of-flight.

Distance Calculation

$$D = \frac{cT}{2}$$

where: c = speed of light, T = time taken for the laser to return.

Advantages: High accuracy, works in low light. **Limitations:** Sparse depth maps, expensive sensors.

Figure: LiDAR depth estimation example.

Fusion of LiDAR and Cameras

Motivation: LiDAR provides sparse but accurate depth, while RGB images contain rich details.

Sensor Fusion

$$D_{\text{final}} = w_1 D_{\text{LiDAR}} + w_2 D_{\text{RGB}}$$

where w_1 and w_2 are learned weights.

Examples:

- RGB-D cameras (e.g., Kinect, Intel RealSense).
- Deep learning fusion networks (e.g., DeepLiDAR, FusionNet).

Section 3. Landmark Detection

Formulation of the Problem

Given: A dataset of 2D landmarks for objects (e.g., facial keypoints).

Notation:

- $\mathbf{y}^{(i)} = (y_1^{(i)}, y_2^{(i)})$: True landmark position for sample i.
- $\mathbf{h}^{(i)} = (h_1^{(i)}, h_2^{(i)})$: Predicted landmark position.
- $\mathbf{\Sigma}^{(i)} \in \mathbb{R}^{2\times 2}$: Covariance matrix modeling uncertainty.

Goal: Learn a probabilistic model to estimate $\mathbf{h}^{(i)}$ and $\mathbf{\Sigma}^{(i)}$, then minimize the Negative Log-Likelihood (NLL) under a Multivariate Normal Distribution.

Multivariate Normal Likelihood

The likelihood of observing a landmark $\mathbf{y}^{(i)}$ given prediction $\mathbf{h}^{(i)}$ follows:

$$p(\mathbf{y}^{(i)}|\mathbf{h}^{(i)}, \boldsymbol{\Sigma}^{(i)}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}^{(i)}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{y}^{(i)} - \mathbf{h}^{(i)})^\top (\boldsymbol{\Sigma}^{(i)})^{-1} (\mathbf{y}^{(i)} - \mathbf{h}^{(i)})\right)$$

where:

- $d = 2$ (for 2D landmarks).
- $|\boldsymbol{\Sigma}^{(i)}|$: Determinant of the covariance matrix.
- $(\boldsymbol{\Sigma}^{(i)})^{-1}$: Precision matrix.

Loss: The Negative Log-Likelihood (NLL) to minimize is:

$$\mathcal{L}_{\text{NLL}} = \sum_i \frac{1}{2}\left(\log|\boldsymbol{\Sigma}^{(i)}| + (\mathbf{y}^{(i)} - \mathbf{h}^{(i)})^\top (\boldsymbol{\Sigma}^{(i)})^{-1} (\mathbf{y}^{(i)} - \mathbf{h}^{(i)})\right) + C$$

e C is a constant independent of predictions.

Simplification: Spherical Covariance

If we assume a **spherical covariance matrix**, meaning the variance is isotropic (same for all directions), we set:

$$\mathbf{\Sigma}^{(i)} = \sigma^2 \mathbf{I}$$

where σ^2 is a scalar variance, shared across all dimensions. Then:

$$|\mathbf{\Sigma}^{(i)}| = \sigma^4, \quad (\mathbf{\Sigma}^{(i)})^{-1} = \frac{1}{\sigma^2}\mathbf{I}$$

Substituting into the NLL loss:

$$\mathcal{L}_{\text{NLL}} = \sum_i \left(\log \sigma^2 + \frac{1}{2\sigma^2} \|\mathbf{y}^{(i)} - \mathbf{h}^{(i)}\|^2 \right) + C$$

This loss function balances prediction accuracy and uncertainty estimation.

Key Takeaways

- **Landmark detection** can be modeled probabilistically with a multivariate normal distribution.
- The NLL loss consists of two terms:
  1. $\|\mathbf{y}^{(i)} - \mathbf{h}^{(i)}\|^2$: Measures prediction error.
  2. $\log \sigma^2$: Encourages learning an adaptive uncertainty.
- **Spherical covariance** simplifies the loss and assumes uncertainty is isotropic.
- This formulation is useful for **uncertainty-aware deep learning models**, where σ^2 can be learned dynamically.

Hands-on coding: A landmark detection task

A task where we want to predict points in a picture. For this, we will use the Biwi Kinect Head Pose Dataset.

Paper reading

LUVLi Face Alignment: Estimating Landmarks' Location, Uncertainty, and Visibility Likelihood

Modern face alignment methods have become quite accurate at predicting the locations of facial landmarks, but they do not typically estimate the uncertainty of their predicted locations nor predict whether landmarks are visible. In this paper, we present a novel framework for jointly predicting landmark locations, associated uncertainties of these predicted locations, and landmark visibilities. We model these as mixed random variables and estimate them using a deep network trained with our proposed Location, Uncertainty, and Visibility Likelihood (LUVLi) loss. In addition, we release an entirely new labeling of a large face alignment dataset with over 19,000 face images in a full range of head poses. Each face is manually labeled with the ground-truth locations of 68 landmarks, with the additional information of whether each landmark is unoccluded, self-occluded (due to extreme head poses), or externally occluded. Not only does our joint estimation yield accurate estimates of the uncertainty of predicted landmark locations, but it also yields state-of-the-art estimates for the landmark locations themselves on multiple standard face alignment datasets. Our method's estimates of the uncertainty of predicted landmark locations could be used to automatically identify input images on which face alignment fails, which can be critical for downstream tasks Kumar et al. [2020].

Facial Landmark Detection: Dlib Model

Dlib's 68 Facial Landmark Model: A pre-trained model that detects 68 keypoints across facial features.

Landmark Indices

- Jawline: Points 1-17
- Eyebrows: Points 18-27
- Nose: Points 28-36
- Eyes: Points 37-48
- Mouth: Points 49-68

Pipeline:

1. Detect face using a Haar Cascade or CNN-based face detector.
2. Apply Dlib's shape predictor to extract landmarks.

Facial Landmark Detection: MediaPipe

Google's MediaPipe: A real-time, deep-learning-based landmark detector.

Advantages of MediaPipe

- Lightweight and fast: Works on mobile devices in real-time.
- Predicts 468 facial landmarks, enabling high-precision applications.
- Uses a deep learning pipeline optimized for efficiency.

Applications:

- Real-time face tracking (Snapchat filters, AR effects).
- Face mesh reconstruction for 3D modeling.

Section 4. Pose Estimation & Keypoint Detection

Pose Estimation: 2D vs. 3D

Key Idea: Predict key human joints to track body movement.

2D vs. 3D Pose Estimation

- 2D Pose Estimation: Estimates joint locations in image space (x, y).
- 3D Pose Estimation: Adds depth information (x, y, z)

Applications:

- Motion tracking in sports and rehabilitation.
- Sign language recognition.

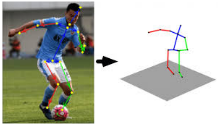

Figure: Comparison of 2D and 3D pose estimation.

Human Pose Estimation: OpenPose

OpenPose: A real-time multi-person keypoint detection system.

Key Features

- Detects 135 keypoints (body, hands, face).
- Uses Part Affinity Fields (PAFs) to connect detected joints.
- Supports multi-person tracking in crowded scenes.

Pipeline:

1. Extract keypoints using CNN-based feature extraction.
2. Compute PAFs to determine joint connections.
3. Assemble human skeleton models from keypoints.

Human Pose Estimation: BlazePose & HRNet

BlazePose (Google MediaPipe):

- Lightweight pose detection for mobile devices.
- Uses 33 keypoints, optimized for real-time tracking.
- Enables applications like yoga tracking and gesture recognition.

HRNet (High-Resolution Network):

- Maintains high-resolution feature maps for accurate keypoint detection.
- Outperforms other models in complex poses (e.g., sports, dance).

Section 5. Practical Implementation

Implementing Depth Estimation

Stereo Vision with OpenCV

- Use `cv2.StereoBM_create()` for block matching.
- Pre-process images: grayscale conversion, rectification.
- Compute disparity map: $d(x, y) = I_L(x, y) - I_R(x, y)$.

Monocular Depth with MiDaS

- Use a pre-trained MiDaS model for single-image depth prediction.
- Works with RGB input:

$$D = f_\theta(I), \quad D : \text{predicted depth}, \quad I : \text{input image}$$

- Supports various backbones (ResNet, Vision Transformers).

Real-Time Depth Estimation

- Load a MiDaS model with PyTorch.
- Process video frames from a webcam.
- Visualize real-time depth maps.

Implementing Landmark Detection

Dlib for Facial Landmark Detection

- Load a pre-trained shape predictor model.
- Detect facial keypoints using:

$$\{(x_1, y_1), ..., (x_N, y_N)\}$$

- Overlay keypoints on the face.

Real-Time Tracking with MediaPipe

- Uses deep learning-based regression for landmark detection.
- Outputs 468 high-precision keypoints.
- Optimized for real-time applications.

Hands-on Demo: Webcam-Based Facial Landmark Detection

1. Capture video using OpenCV.
2. Process frames using MediaPipe's Face Mesh model.
3. Visualize real-time facial landmarks.

Section 6. Conclusion & Discussion

Conclusion & Discussion

Strengths and Limitations

- **Stereo Vision:** Good accuracy but needs two cameras.
- **Monocular Depth Estimation:** Works with single images, but lacks absolute depth accuracy.
- **LiDAR:** High precision but expensive and sparse.

Future Trends

- Neural Radiance Fields (NeRF) for photo-realistic depth estimation Mildenhall et al. [2020].
- Transformer-based Keypoint Detection for more robust landmark estimation.

Open Challenges

- Depth estimation under low-light or occlusions.
- Real-time landmark detection with high precision in occluded faces.

Bibliography

Abhinav Kumar, Tim K Marks, Wenxuan Mou, Ye Wang, Michael Jones, Anoop Cherian, Toshiaki Koike-Akino, Xiaoming Liu, and Chen Feng. Luvli face alignment: Estimating landmarks' location, uncertainty, and visibility likelihood. In **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**, pages 8236–8246, 2020.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. URL https://arxiv.org/abs/2003.08934.