Lecture 1

Course Overview and Introduction

- This course on computer vision has been redesigned with a greater emphasis on **machine learning** and practical applications, reducing the focus on theory and complex mathematics.
- The course is delivered entirely online, offering both live sessions and recorded materials. Active participation in these sessions is encouraged.
- The institute of computer graphics specializes in **visual computing**, using computer science to address problems using visual data.

Labs

- The computer vision labs have been redesigned to be completed **online** with no physical presence required.
- The format includes recordings, slides, coding assignments, and model solutions.
- The labs are divided into two parts:
    - **Onboarding labs**: These are designed to teach or refresh basic Python programming skills, particularly for image processing and machine learning.
    - **Individual assignments**: These assignments involve implementing Python code related to the lecture topics. They are designed to provide hands-on experience with tools and techniques applicable to real-world problems.
- Students get one week to complete the assignment, and generally have two weeks to complete the assignment.
- **Open lab sessions** offer opportunities for students to ask questions and address any problems they encounter.

Assignments

- There are four individual assignments that are related to the lecture topics.
- The aim of the assignments is to give students practical skills relevant to industry applications, rather than focusing on research.

Assessment

- The lecture component is assessed through a **Moodle exam** at the end of the semester.

- The lab component is graded based on four assignments, each contributing 25% to the final grade.
- Physical presence is mandatory for the exams.

Prerequisites

- To succeed in this course, students should have:
    - Proficiency in **Python programming**
    - A basic understanding of **machine learning** concepts
    - Knowledge of **linear algebra**

Course Topics

- The course will cover the following topics:
    - Image capturing
    - Digital image processing
    - Machine learning
    - Feature extraction
    - Segmentation
    - Optical flow
    - Object detection
    - Multi-view geometry
    - 3D vision
- The final lecture will discuss current trends in computer vision.

Light Fields

- Light fields are a central research tool, and can be combined with computer vision, machine learning, and optics to solve unsolved problems.
- The goal of both computer vision and computer graphics is to describe how light is transported in 3D environments.
- The **plane optic function** describes light traveling along a straight line as a 3-dimensional vector.

- Light fields simplify the plane optic function, reducing it to a four-dimensional function that is easier to record and reproduce.

Human vs. Computer Vision

- Human vision involves a 3D environment, light, and the eye's lens projecting an image onto the retina. The brain then processes these visual signals.
- In computer vision, computers need intelligent methods and reasoning to interpret image information.
- The course will address classical computer vision tasks like classification, tracking, motion estimation, and scene reconstruction.
- It's worth noting that computer vision is not always easy, and even seemingly simple tasks can be challenging for computers.

Historical Context

- The field of computer vision began with the invention of digital cameras.
- Early approaches were model-based, relying on algorithms and mathematical equations.
- The introduction of AlexNet in 2012 marked a turning point, demonstrating the advantages of machine learning and data-driven approaches.

Machine Learning in Computer Vision

- Machine learning involves learning a function that represents data as accurately as possible.
- Deep neural networks, with millions or billions of parameters, enable more complex and sophisticated functions compared to simpler models.
- The course includes a machine learning class that teaches the fundamentals of machine learning applicable to computer vision.

Related Fields

- Computer vision is related to various fields, including:
  - AI and machine learning
  - Robotics
  - Cognitive science

    - Neuroscience
    - Computer graphics
    - Image processing
    - Signal processing

Lecture 2

Here are detailed lecture notes incorporating information from the sources, "Capturing Digital Images: Camera Optics, Sensors, and Calibration" and "Lec2_Capturing_Digital_Images.pdf":

Computer Vision: Capturing Digital Images

Camera Optics and the Pinhole Camera Model

- **Camera Obscura/Pinhole Camera** The pinhole camera, or camera obscura, projects light from a 3D scene through a small pinhole onto a canvas, creating a flipped image. The scaling of the image depends on the distance between the pinhole and the canvas.
- **Lenses vs. Pinhole** While the pinhole camera serves as a basic model, real-world cameras use lenses (or trains of lenses) and digital image sensors (or analog film).
    - The primary drawback of pinhole cameras is the trade-off between image sharpness and brightness. A smaller pinhole increases sharpness but reduces light, resulting in a dark image. A larger pinhole allows more light but causes blur.
    - **Lenses** Lenses focus incoming light to a single point on the image sensor, allowing for brighter and sharper images. Lenses, made of optically dense material like glass, use refraction to focus light.
- **Focal Distance** The focal distance is the distance from the lens at which light is focused when it comes from an infinite distance. It is determined by the lens's thickness, surface radii, and material.
- **Aperture** An aperture is used to control the amount of light entering the camera. It can be adjusted to change the effective area of the lens.
    - A smaller aperture (stopping down the lens) reduces the amount of light collected, but increases the **depth of field**, making objects at different distances appear sharper.

    - A larger aperture allows more light, which is useful in low light conditions to reduce exposure time and avoid motion blur, but it results in a shallower depth of field.

- **f-number** The f-number represents the ratio of the focal length to the aperture diameter. A higher f-number means a smaller aperture and a larger depth of field. Each "f-stop" changes the amount of light by a factor of two.

Drawbacks and Aberrations of Lenses

- Lenses are not perfect and introduce optical artifacts due to imperfect manufacturing, non-homogeneous material, and other factors.
- **Chromatic Aberrations** Imperfect lenses cause light rays to not perfectly intersect at one point, resulting in slight defocus. The refraction of light within the lens depends on the wavelength (color) of the light, causing different colors to be differently defocused.
- **Radial Distortion** Wide field-of-view lenses, like fisheye lenses, can cause geometric distortions such as barrel distortion, where the image appears warped. Conversely, lenses with a narrow field of view can cause pincushion distortion.
- **Vignetting** Vignetting occurs when the edges of an image appear darker than the center due to light clipping by multiple lenses within the camera.

Thin Lens Model

- The thin lens model simplifies the camera model by assuming an infinitely thin lens.
- Despite using lenses, the position where a 3D point is projected onto the image plane is the same as in a pinhole camera model.
- Basic trigonometry can be used to compute the projection of a 3D point onto the 2D image plane, using the focal distance.

Digital Image Sensors

- Digital cameras use image sensors like CCD or CMOS sensors to record incoming light. These sensors consist of a grid of photosensors that measure the amount of light.
- The images are discrete 2D functions due to the finite resolution and digital nature of the sensor.

- Sensors are susceptible to noise. The **signal-to-noise ratio** is more important than the amount of noise itself. Techniques like averaging multiple images or filtering can suppress noise.
- The relationship between the amount of incoming light and the sensor values is non-linear in consumer cameras, mimicking human visual perception. **Linearization** is needed for computer vision to ensure a linear relationship between incoming light and measured light.

Digital Color Images

- Color filters (red, green, and blue) are placed on top of the image sensor to capture color information.
- The **Bayer filter** is a common configuration with twice as many green pixels as red or blue, mimicking the higher sensitivity of human vision to green light.
- Demosaicing is used to interpolate the missing color components for each pixel, since each pixel only captures one color natively.

Mathematical Camera Models and Projection

- The goal is to find a mathematical representation of a camera that maps a 3D object to a 2D image. This mapping can be represented as a transformation matrix.
- **Perspective Projection** Most cameras use perspective projection, where 3D points are projected onto the image plane by computing rays from each point to the projection center (pinhole). This results in perspective distortion, where parallel lines converge at vanishing points.
- **Parallel (Orthographic) Projection** In contrast to perspective projection, parallel projection projects points parallel onto the image plane without perspective distortion.

Homogeneous Coordinates

- Homogeneous coordinates are used to describe transformations (translations, scaling, rotations) using matrix multiplications.
- A 2D point (x, y) becomes (x, y, 1), and a 3D point (x, y, z) becomes (x, y, z, 1).
- After applying transformations, homogeneous coordinates are converted back to heterogeneous coordinates by dividing by the homogeneous coefficient (w).
-

Camera Parameters and Calibration

- The transformation from 3D world coordinates to 2D image coordinates is influenced by camera parameters and pose.
- **Intrinsic Parameters** These parameters describe the internal characteristics of the camera, including focal length, scaling factors, and principal point (the center of the image).
- **Extrinsic Parameters** These parameters describe the camera's pose (position and orientation) in the world coordinate system, including translation and rotation.
- **Camera Calibration** Camera calibration is the process of estimating the intrinsic and extrinsic parameters of a camera. It involves using images of known calibration patterns (e.g., checkerboards) and optimization techniques to minimize the reprojection error. Libraries like OpenCV provide functions for camera calibration.
- Why is it needed? Camera calibration allows to accurately map 3D points to 2D images, which is essential for applications like 3D reconstruction.

Lecture 3

Here are detailed lecture notes on digital image processing, incorporating explanations and information from the sources:

Computer Vision: Digital Image Processing

Digital Image Processing Domains

Digital image processing involves different domains, each offering unique ways to analyze and modify images. The primary goal of image processing is often **feature detection**, where features are pixels or image regions containing meaningful information about the image's content.

The most common domains include:

- **Spatial Domain**: This is the most intuitive domain, where image processing is performed directly on the pixels of the image, analyzing and modifying their intensities and colors.
- **Frequency Domain**: In this domain, the image is transformed into its frequency components using techniques like Fourier transformation. Processing involves

modifying the spectrum of the image, which can be useful for tasks like noise removal or sharpening.

- **Gradient Domain**: This domain involves analyzing and manipulating the gradients (changes in pixel intensity) within an image. It is useful for tasks such as image blending, High Dynamic Range (HDR) imaging, and removing specular reflections.

Spatial Domain Processing

Spatial domain processing involves working directly with the pixels of an image. Key operations in this domain include:

- **Image Gradients**: Computing the partial derivatives of an image in the horizontal (x) and vertical (y) directions can reveal edges and other features. This is done by considering the image as a 2D discrete function and calculating the rate of change in pixel intensities.
    - **Horizontal and Vertical Edge Detection**: By computing gradients in the x and y directions, vertical and horizontal edges can be detected, respectively.
    - **Gradient Magnitude and Orientation**: Combining the gradient responses in the x and y directions allows for the computation of the magnitude (strength) and orientation (angle) of edges. The gradient magnitude indicates the edge strength, while the orientation reveals the edge's direction.
- **Convolution**: Convolution is a fundamental operation in image processing that involves applying a filter kernel to an image. A filter kernel is a matrix of weights that determines the effect of the convolution.
    - **The Convolution Operation**: The kernel is滑移 across the image, and at each position, a weighted sum of the pixel values under the kernel is computed. This weighted sum becomes the new value of the central pixel in the output image.
    - **Shift Invariance**: Convolution is a shift-invariant operation, meaning that the result of the convolution is independent of the position in the image.
    - **Zero Padding**: To handle border pixels, where the kernel extends beyond the image boundaries, zero-padding is often used. This involves adding a border of zeros around the image before applying the convolution.
    - **Blurring (Box Filter)**: Applying a box filter, which averages the pixel values in a neighborhood, results in a blurred image. This is a low-pass filter that removes high-frequency components (details) from the image.

    - **Impulse Filter**: An impulse filter, with a value of 1 in the center and 0 elsewhere, does not change the image when convolved with it.

Frequency Domain Processing

Frequency domain processing involves transforming the image into its frequency components using a Fourier transformation.

- **Fourier Transformation**: This transformation decomposes the image into its constituent frequencies, representing the image as a spectrum.
    - **Spatial vs. Frequency Domain**: A function in the spatial domain, denoted as *f[n, m]*, is transformed into the frequency domain as *F[u, v]*, where *n, m* are spatial coordinates and *u, v* are frequency coordinates.
    - **Frequency Components**: Each point in the frequency domain represents a particular frequency component in the image. The magnitude of the component indicates its amplitude, and the phase indicates its shift.
    - **High vs. Low Frequencies**: High frequencies correspond to fine details and sharp transitions in the image, while low frequencies correspond to smooth regions and overall shapes.
- **Convolution Theorem**: Convolution in the spatial domain is equivalent to multiplication in the frequency domain. This means that convolving an image with a kernel can be done by multiplying their respective spectra and then transforming back to the spatial domain.

Filter Types in Frequency Domain

- **Low-Pass Filter**: Allows low-frequency components to pass through while blocking high-frequency components. This results in a blurred image with reduced details.
    - **Box Filter**: A simple low-pass filter that averages pixel values in a neighborhood.
    - **Gaussian Filter**: A more sophisticated low-pass filter that uses a Gaussian function to weight the pixel values, resulting in a smoother blur compared to the box filter.
- **High-Pass Filter**: Allows high-frequency components to pass through while blocking low-frequency components. This enhances edges and details in the image.
    - **Laplacian Filter**: A high-pass filter that can be implemented by subtracting a Gaussian-blurred image from the original image.

Image Pyramids (Scale Invariance)

Image pyramids are used to process images at multiple scales, allowing for the detection of features of different sizes.

- **Gaussian Pyramid**: Created by repeatedly applying a Gaussian filter and downsampling the image. Each level of the pyramid represents the image at a different scale, with progressively lower frequencies.
- **Laplacian Pyramid**: Created by subtracting adjacent levels of the Gaussian pyramid. Each level represents a different frequency subband, corresponding to different scales of details in the image.

Deconvolution/Inverse Filtering

Deconvolution is the process of reversing the effects of convolution, such as blurring.

- **Inverse Filtering**: In theory, deconvolution can be achieved by dividing the spectrum of the blurred image by the spectrum of the blurring kernel. However, this is highly sensitive to noise and can lead to artifacts.
- **Regularized Techniques**: To address the limitations of inverse filtering, regularized techniques such as Wiener filtering, Lucy-Richardson algorithm, and blind deconvolution are used. These techniques use optimization algorithms and prior knowledge to estimate the original image.

Gradient Domain Processing

Gradient domain processing involves manipulating the gradients of an image.

- **Gradient Domain**: The gradient domain represents an image as a vector field, where each pixel has a vector representing the horizontal and vertical gradients.
- **Gradient Modification**: Image processing operations are performed by modifying this vector field.
- **Poisson Equation**: To reconstruct the image from the modified gradient field, a Poisson equation is solved. This involves finding an image whose Laplacian (second-order derivative) matches the divergence of the modified gradient field.
- **Applications**: Gradient domain processing is useful for tasks such as removing specular reflections, image stitching, and High Dynamic Range (HDR) imaging.

Convolutional Neural Networks (CNNs)

CNNs are a type of neural network that uses convolution as a fundamental operation.

- **Learned Filter Kernels**: Unlike traditional image processing, where filter kernels are handcrafted, CNNs learn the filter kernel weights from data. This allows the network to automatically learn features that are relevant to the task.
- **Transposed Convolution**: Transposed convolution is used for upsampling in CNNs. It involves adding padding and striding to the input image before performing convolution, resulting in an upscaled output.

Lecture 4

Here are detailed lecture notes on machine learning for computer vision, incorporating explanations and information from the sources:

Machine Learning for Computer Vision

Course Overview

The lecture is part of a computer vision course. The course covers a range of topics, including:

- Digital image processing
- Machine learning
- Feature extraction
- Segmentation
- Optical flow
- Object detection
- Multi-view geometry
- 3D vision

Machine Learning Fundamentals

Machine learning involves using data to create models that can abstract and generalize from that data. The goal is to develop a model during a training phase that can accurately process new data during a testing phase.

Classes of Machine Learning:

- **Unsupervised Learning**: Involves finding structure in unlabeled data. The algorithm identifies patterns, such as clusters, without human input.

- **Supervised Learning**: Uses labeled data to make predictions. The algorithm learns to classify new data based on the provided labels. Tasks include classification and regression.
- **Semi-Supervised Learning**: A combination of labeled and unlabeled data is used to solve prediction tasks.
- **Self-Supervised Learning**: Uses the properties of the data itself to create supervisory signals, eliminating the need for external labels.

Dimensionality Reduction

Dimensionality reduction is a key concept in machine learning. It involves reducing the number of dimensions needed to represent data without losing critical information. This is essential for efficient data processing and generalization.

- **Goal**: To convert data points in d dimensions to k dimensions, where $k < d$, with minimal information loss.
- **Example**: Reducing a 3D dataset to 2D by projecting the data points onto a 2D plane.
- **Importance**: It helps in creating a more general and efficient representation of training data.

Principle Component Analysis (PCA):

PCA is a technique used to reduce the dimensionality of data by finding the directions (principal components) along which the data varies the most.

- **Variance**: PCA identifies the vector that captures the maximum variance in the data.
- **Eigenvectors and Eigenvalues**: PCA computes eigenvectors and eigenvalues of the covariance matrix of the data. Eigenvectors represent the directions of the principal components, and eigenvalues represent the amount of variance along those directions.
- **Selecting Components**: The eigenvectors with the largest eigenvalues are chosen to represent the data, effectively reducing its dimensionality.

Face Recognition with Eigenfaces:

Eigenfaces is an example of using PCA for face recognition.

- **Face Space**: Each face image is treated as a point in a high-dimensional space. For example, a 100x100 image is a point in a 10,000-dimensional space.

- **Dimensionality Reduction**: PCA is used to find a lower-dimensional subspace that represents the set of faces.
- **Eigenfaces**: PCA extracts a set of vectors (eigenfaces) that span the space of faces. These eigenfaces are the eigenvectors of the covariance matrix of the face images.
- **Face Reconstruction**: Any face can be represented as a linear combination of the eigenfaces.
- **Recognition**: Face recognition is performed by projecting a new face onto the eigenfaces and finding the nearest neighbor in the training set.

Supervised Learning: Linear Regression

Linear regression is a supervised learning method used to model the relationship between a dependent variable and one or more independent variables.

- **Hypothesis Space**: The relationship between the input (X) and output (Y) is assumed to be roughly linear.
- **Objective**: To find the parameters that best fit the data by minimizing the squared error between predictions and target values.
- **Loss Function**: Measures the error between predicted and actual values. The least-squares objective is a common loss function.
- **Limitations**: Limited in terms of generalization compared to more complex models like neural networks.

Deep Learning

Deep learning uses neural networks to model complex, non-linear functions.

- **Training, Validation, and Testing**: The dataset is divided into training, validation, and testing sets. The training set is used to train the model parameters, the validation set is used to tune hyperparameters, and the testing set is used to evaluate the final accuracy.

Convolutional Neural Networks (CNNs)

CNNs are a type of neural network commonly used for image classification and computer vision tasks.

- **Convolutional Layers**: These layers apply convolutional kernels to the input image to extract features. The kernel computes a weighted sum of the pixels in the patch.

- **Activation Maps**: Each convolutional layer produces a set of feature maps, also known as channels, which represent the responses of the convolutional kernels.
- **ReLU Activation**: The Rectified Linear Unit (ReLU) activation function is used to introduce non-linearity, mitigating the vanishing gradient problem. ReLU sets all negative values to zero.
- **Pooling**: Pooling layers down-sample the feature maps, reducing the spatial dimensions while retaining the most important information. Max pooling is a common technique that selects the maximum value from each local region.
- **Receptive Field**: The receptive field is the region of the input image that a particular neuron in a convolutional layer is "looking at". It depends on the kernel size, stride, and pooling layers.
- **Classification**: The final layers of a CNN typically consist of fully connected layers that map the learned features to class labels.

Feature Extraction

Feature extraction involves identifying and isolating the most salient aspects of an image which most succinctly describes its contents. These features, which are often responses in the activation layers of a convolutional layer, are then used in subsequent machine learning models.

Lecture 5

Here are detailed lecture notes on feature extraction in computer vision, incorporating explanations and information from the sources:

Feature Extraction in Computer Vision

Role of Image Features

- **Image understanding** is a primary goal in computer vision, achieved by algorithms analyzing images to extract meaningful information.
- **Not all pixels are equally important**; some contain more information than others for understanding the image.
- **Features are specific pixels or patterns** that are crucial for scene understanding. Examples include edges that define object boundaries and textures that provide information about surface properties.

- **Feature extraction** is the process of computing where the important features are, describing them, and using them to make inferences about the image content.

Computing Image Features

- Feature extraction involves computing and describing the key elements in an image to facilitate understanding and interpretation.
- **Partial derivatives** can be used to compute gradients that show horizontal and vertical edges. These can be implemented using convolution with specific filter kernels.
- **Filter kernels** can be designed to identify intensity changes in different directions, effectively detecting edges.
- **Convolution** is a key operation of filter kernels and is shift invariant. Shift invariance means that the kernels perform the same operation regardless of their location in the image.

Model-Based Feature Extraction

- **Model-based feature extraction** involves manually designing filter kernels to extract specific features.
- Historically, these kernels were designed based on an understanding of how the human visual perception system works, mimicking the brain's initial processing stages such as edge and point detection.
- **Filter banks**, which are sets of filter kernels, can be used to detect various types of features, such as points and edges at different orientations.
- Examples of hand-tuned kernels include Gaussian kernels (for blurring) and Laplacian kernels (for detail extraction).

Learning-Based Feature Extraction

- **Learning-based approaches** use deep neural networks (DNNs) to automatically learn filter kernels from training data.
- **Deep neural networks** can learn the best filter kernels for specific image content by training on a particular dataset. For instance, a network trained on cat images will develop kernels that are highly effective at extracting features relevant to cats.
- This approach allows the feature extraction to be specialized and optimized for the application at hand.

- **Convolutional Neural Networks (CNNs)** are a common architecture for learning-based feature extraction. The kernel weights in the convolutional layers are learned during training to extract relevant features.

Scale Invariance

- **Scale invariance** is the ability to detect features regardless of their size in the image.
- **Image pyramids** are a classical tool for achieving scale invariance by applying filters and convolutions at multiple scales of the image.
- **Gaussian and Laplacian pyramids** are types of image pyramids. Gaussian pyramids involve progressive blurring and down-sampling, while Laplacian pyramids decompose the image into different frequency sub-bands.
- The **Scale-Invariant Feature Transform (SIFT)** is a model-based approach that uses a combination of Gaussian pyramids and feature descriptors to achieve scale and shift invariance.

Scale-Invariant Feature Transform (SIFT)

- SIFT involves identifying key points in an image that are distinctive and can be reliably matched across different views of the same object or scene.
- **Key steps in SIFT:**
    - **Scale-space construction**: Generating a Gaussian pyramid by convolving the image with Gaussian kernels at different scales.
    - **Difference of Gaussians (DoG)**: Computing the Laplacian pyramid by subtracting adjacent layers in the Gaussian pyramid.
    - **Local extrema detection**: Identifying local extrema in the DoG pyramid as potential feature points.
    - **Filtering**: Eliminating features in low-contrast regions or along edges.
    - **Orientation assignment**: Assigning an orientation to each key point based on the dominant gradient direction in its neighborhood.
    - **Feature descriptor**: Computing a descriptor for each key point based on the gradient orientations in its neighborhood. This descriptor is typically a 128-dimensional vector.

Feature Usage in Computer Vision Applications

- **Image classification**: Extracted features serve as input for classifiers, such as those in CNNs, to categorize images based on their content.
- **Image registration**: Features are used to align multiple images, such as in creating panoramic images. This involves finding corresponding points in different images and transforming them to match.
- **Other applications**: Features can be used for tasks beyond classification, such as image fusion and 3D reconstruction.

Advantages of Learning-Based Approaches

- **Adaptability**: Learning-based methods can adapt to specific tasks and datasets, resulting in more effective feature extraction.
- **Automation**: Features are learned automatically, reducing the need for manual design and tuning.
- **Improved performance**: Deep learning models have achieved state-of-the-art results in many computer vision tasks due to their ability to learn complex and high-level features.

AlexNet Architecture

- **AlexNet** was a breakthrough CNN architecture that demonstrated the power of learning-based feature extraction for image classification.
- It consists of five convolutional layers followed by fully connected layers.
- The convolutional layers learn different types of features, ranging from low-level edges and textures to high-level object parts.
- The depth of the network and the use of ReLU activation functions contributed to its success.

Modern CNN Architectures

- **VGGNet and ResNet** are deeper CNN architectures that have further improved the performance of image classification and other computer vision tasks.
- **ResNet** employs skip connections to address the vanishing gradient problem, allowing for the training of very deep networks.
- Deeper networks can learn more representative features and achieve better performance.

Using Features Beyond Classification

- Features learned through classification tasks can be used for other applications, such as image fusion and 3D reconstruction.
- **Image fusion** combines information from multiple images to create a single, more informative image. VGGNet features can be used to identify salient regions in different images and merge them.
- **Depth from defocus** uses ResNet features to estimate the depth of objects in a scene from a focal stack of images.
- These applications demonstrate the versatility and generalizability of learned features.

Conclusion

- Feature extraction is a critical step in computer vision, enabling algorithms to understand and interpret images.
- Both model-based and learning-based approaches have their strengths, but learning-based methods, particularly those using deep neural networks, have become dominant due to their adaptability and performance.
- The features learned through these methods can be applied to a wide range of computer vision tasks beyond classification, demonstrating their importance and versatility.

Lecture 6

Here are detailed lecture notes about image segmentation in computer vision, incorporating explanations and information from the sources:

Image Segmentation in Computer Vision

What is Image Segmentation?

- **Segmentation** is the process of identifying and grouping pixels that belong together in an image. The decision of whether pixels belong together can be application-dependent.
- **The goal** is to determine which pixels belong to the same segment because they relate to the same object or semantic surface.
- **This involves** defining a function to assess pixel similarity based on factors like distance, color, or texture, or learning this function through machine learning.

- **There are two main approaches** to solving the segmentation problem:
  - **Model-based**: Creating an algorithm or equation that defines the problem's solution.
  - **Learning-based**: Using machine learning to learn how to solve the problem.

Types of Segmentation

- **Semantic Segmentation:** Pixels are assigned to segments belonging to the same object class, without differentiating between instances of that class. For example, all car pixels are labeled the same, but individual cars are not distinguished.
- **Instance Segmentation:** First, object detection is performed to detect individual objects. Then, segmentation identifies the pixels within each bounding box that belong to a specific object instance. For example, individual cars are detected and segmented separately.
- **Panoptic Segmentation:** Combines semantic and instance segmentation to segment both object instances and background pixels not part of a detected object. This approach segments different objects of the same class and segments pixels not part of any trained object detector.

Applications of Image Segmentation

- **Image Editing**: Removing backgrounds or isolating objects.
- **Robotics**: Segmenting cars, street lanes, and obstacles for autonomous driving.
- **Medical Imaging**: Segmenting tissue types, tumors, and other structures in CT and MRI scans.

Segmentation by Clustering

- **Concept**: Pixels are treated as points in a high-dimensional space, such as a 3D color space (RGB) or a 5D space that includes color and location.
- **Clustering**: Pixels are grouped into segments based on their proximity in this feature space.
- **K-Means Clustering**: A common algorithm used to cluster pixels.
  - **Process**:
    - Choose *k* points randomly as initial cluster centers.
    - Allocate each point to the nearest cluster.

        - Replace each cluster center with the mean of the new cluster points.
        - Repeat until cluster centers no longer change.
    - **Similarity**: Determined by a defined function, such as color similarity or spatial location.
    - **Limitation**: Requires predefining the number of clusters (*k*).

Segmentation with Graphs

- **Concept**: Pixels are represented as nodes in a graph, with edges connecting pairs of pixels.
- **Affinity Weight**: The edge weight measures the similarity between pixels.
- **Clustering**: Pixels are clustered into segments using methods like graph cuts or clustering by graph eigenvectors.
- **Affinity Matrix**: A matrix *A* (size *N*x*N* for *N* elements) represents the graph, where matrix coefficients indicate pixel similarity.
- **Clustering by Graph Eigenvectors**:
    - Goal: To find clusters where elements are strongly connected and have high affinity values.
    - Weight Vector: An assignment is made for cluster *n* using a weight vector *w_n*, with high values indicating strong connectivity to the cluster.
    - Mathematical Objective: Maximize the expression *w_n*^T *A* *w_n* to determine if the assignment is good.
    - PCA: Using Principal Component Analysis (PCA) to find the most important eigenvectors.
    - Algorithm:
        - Construct the affinity matrix *A*.
        - Compute eigenvalues and eigenvectors of *A*.
        - Iteratively assign elements to clusters based on the eigenvector corresponding to the next largest eigenvalue and zero out clustered elements in *A*.

Segmentation by Fitting

- **Concept**: Utilizing knowledge about the shapes of segments.
- **Parameterization**: Segments are described using parametric equations.
- **Hough Transform**: A method to transform pixels into a parameter space.
    - **Line Parameterization**: Lines can be described using parameters such as angle (θ) and distance from the origin (r).
    - **Process**:
        - Transform the image from the spatial domain (*x*, *y*) to the Hough space (θ, r).
        - Increment counters in the Hough space for each line that goes through a pixel.
        - Find peaks in the Hough space, corresponding to the strongest lines in the image.

Segmentation by Learning

- **Supervised Learning**: Uses labeled training data where each pixel is assigned to a semantic category.
- **CNNs**: Convolutional Neural Networks are trained to classify each pixel in an image.
- **Issue with Standard CNNs**: Classification architectures reduce feature spatial sizes, which is not ideal for semantic segmentation that requires output size to match the input size.
- **Solutions**:
    - **Convolutional Layers Without Downsampling**: Design a network with only convolutional layers to make pixel-wise predictions, but this is computationally expensive.
    - **Encoder-Decoder Architecture**: Use downsampling (pooling, strided convolution) and upsampling (transposed convolution or unpooling) to reduce computational cost.
    - **Autoencoders**: Architectures with an encoder to reduce dimensionality and a decoder to reconstruct the original resolution.
    - **Unconnected Autoencoders**: Can lead to imprecise feature prediction due to loss of spatial information during downsampling.

    - **Connected Autoencoders (U-Nets)**: Overcome the problem of detail loss by using skip connections between corresponding encoder and decoder layers.

U-Nets

- **Skip Connections**: Concatenate the output of an encoder level with the input of the corresponding decoder level to preserve spatial details.
- **1x1 Convolution**: Used for linear projection of feature stacks (dimensionality reduction).

Course Topics Recap

- The lecture covered segmentation, following previous topics such as image processing, machine learning, and feature extraction. The next lecture will cover optical flow.
- **Assignments**: The second assignment focuses on feature extraction using VGG-19 and ResNet-152 architectures, as well as crack segmentation using Yolov8.

Lecture 7

Here are detailed lecture notes about optical flow in computer vision, incorporating explanations and information from the sources:

Optical Flow

What is Optical Flow?

- **Definition**: Optical flow describes the motion of points or features in an image over time. It represents the apparent movement of pixels between consecutive frames in a video sequence.
- **Motion Vectors**: Optical flow is represented by motion vectors, which indicate the direction and speed of movement for each point.
- **Time Series Analysis**: It involves analyzing a time series of images (a video) to estimate pixel motion from one image *I(x, y, t)* to the next *I(x, y, t+1)*.
- **Two Types**:
    - **Sparse Optical Flow**: Motion vectors are computed for a discrete set of points or features.
    - **Dense Optical Flow**: A motion vector is computed for every pixel in the image.

- **Visualization**:
  - **Sparse**: Flow vectors are visualized as arrows, with the direction and length indicating motion direction and speed.
  - **Dense**: Flow vectors are color-coded to represent direction and magnitude. The color indicates direction, and the intensity (brightness) indicates the vector's length.

Assumptions Underlying Optical Flow Computation

- **Brightness Constancy**: Assumes that the brightness or color of a point remains approximately constant as it moves from one frame to the next. A point at time t looks the same at time $t+1$.
  - Expressed mathematically as: $I(x + u, y + v, t + 1) = I(x, y, t)$.
- **Small Motion**: Assumes that points do not move very far between consecutive frames. The displacement (u, v) is small.

The Optical Flow Equation

- **Derivation**: Based on the brightness constancy assumption and Taylor expansion.
- **Taylor Expansion**: A mathematical tool used to approximate a function as a series of terms.
- **Gradients**: The equation involves spatial and temporal image intensity gradients.
  - I_x: First-order partial derivative of image intensity in the x-direction (gradient in x).
  - I_y: First-order partial derivative of image intensity in the y-direction (gradient in y).
  - I_t: First-order partial derivative of image intensity in time (gradient in time).
- **The Equation**:
  - $I_x * u + I_y * v + I_t = 0$.
  - This equation represents the brightness constancy constraint.
- **Challenge**: One equation with two unknowns (u, v) for each pixel.

Overcoming the Aperture Problem

- **Aperture Problem**: The optical flow equation alone cannot uniquely determine the motion vector because it has two unknowns (u, v) but only one equation per pixel. It can only identify motion along the gradient direction.
- **Assumption of Spatial Coherence**: Pixels within a local neighborhood have similar motion vectors.
- **Neighborhood Consideration**: By considering a neighborhood (e.g., 5x5 window), multiple equations can be generated. For example, a 5x5 window provides 25 equations.
- **Solving the System**: This results in an over-constrained linear equation system that can be solved numerically to estimate *u* and *v*.

Addressing Large Motions

- **Problem**: The basic optical flow computation works best when motion between frames is small (on the order of one pixel).
- **Gaussian Image Pyramids**: Used to handle larger motions by reducing the image resolution.
- **Process**:
    - **Downsampling**: The image is downsampled to create multiple layers in a pyramid, each with a lower resolution.
    - **Flow Computation**: Optical flow is computed at the lower resolution levels where motion is smaller.
    - **Warping and Upsampling**: The flow field is warped and upsampled to the original resolution.

Optical Flow Methods

- **Lucas-Kanade Optical Flow**:
    - A classical method that combines the optical flow equation, spatial coherence, and Gaussian image pyramids.
    - Often used for sparse optical flow fields.
    - Involves solving a system of equations derived from a local neighborhood of pixels to estimate the optical flow vector (u, v).
- **Horn-Schunck Optical Flow**:

- A dense optical flow method that enforces both brightness constancy and smoothness constraints.
- **Smoothness Constraint**: Assumes that flow vectors change smoothly over a local area.
- **Objective Function**: Minimizes an objective function that combines the brightness constancy constraint and the smoothness constraint.
    - $E = \Sigma [E\_c(i, j) + \lambda E\_s(i, j)]$.
    - $E\_c(i, j)$ represents the brightness constancy constraint.
    - $E\_s(i, j)$ represents the smoothness constraint.
    - λ is a regularization constant that weights the smoothness term.
- **Numerical Solution**: Solved using numerical optimization techniques to find the flow vectors that minimize the objective function.

Optical Flow and Machine Learning

- **Deep Learning**: CNNs, particularly autoencoder architectures, can be trained to estimate optical flow.
- **Encoder-Decoder Architectures**: Common architectures used for optical flow estimation.
    - **Input**: Two consecutive RGB images.
    - **Output**: A dense flow field.
- **Training**: Networks are trained using supervised learning with ground truth flow fields.
    - **Training Data**: Often generated using computer graphics techniques to create realistic image sequences with known motion.
- **Examples**:
    - **FlowNetS**: A simple autoencoder architecture for optical flow estimation.
    - **FlowNetCorr**: A more complex architecture that uses a correlation layer to compare feature maps from the two input images.

Applications of Optical Flow

- **MPEG Compression**: Used to reduce data by storing keyframes and motion vectors.

- **Autonomous Driving**: Helps in understanding the motion of objects, such as cars and pedestrians.
- **Traffic Monitoring**: Used to measure the speed and movement patterns of objects.
- **Object Tracking**: A fundamental component in tracking objects over time.

Key Considerations

- **Illumination Changes**: Optical flow computation is sensitive to changes in lighting conditions, which can violate the brightness constancy assumption.
- **Occlusion**: Occlusion (where a point disappears or is covered) can also cause errors in optical flow estimation.
- **Parameter Selection**: The choice of window size (for spatial coherence) and the regularization constant (for smoothness) can impact the accuracy and stability of the results.

These notes provide a comprehensive overview of optical flow, covering its theoretical foundations, computational methods, and applications.

Lecture 8

Here are detailed lecture notes on object detection, incorporating explanations and information from the sources.

Object Detection

Overview

- **Definition**: Object detection involves identifying and localizing multiple objects within an image, assigning each object to a specific class and determining its position, usually using bounding boxes.
- **Difference from Other Tasks**:
    - **Classification**: Determines if an object is present in an image but does not provide location information or detect multiple objects.
    - **Semantic Segmentation**: Identifies the class of each pixel but does not distinguish between separate objects of the same class.

    - **Instance Segmentation**: Extends segmentation by differentiating between individual objects of the same class, providing a pixel-level mask for each object.

Object Detection Approaches

- **Challenges**: Object detection architectures must handle a varying number of outputs depending on the number of objects detected.
- **Naive Approach**: Applying a Convolutional Neural Network (CNN) to different crops of the image to classify each crop as either an object or background.
    - **Problem**: Computationally expensive due to the need to apply the CNN to a huge number of locations, scales, and aspect ratios.
- **Region Proposal**: A strategy to determine regions of interest in the image where objects are likely to be located, reducing the number of regions to be processed.

Regional-Based CNN (R-CNN)

- **Process**:
    - **Region Proposal**: Extracts Regions of Interest (RoIs) from a proposal method such as Selective Search (~2k regions).
    - **ConvNet Forward Pass**: Warps each region and forwards it through a ConvNet.
    - **Classification**: Classifies regions with Support Vector Machines (SVMs).
    - **Bounding Box Regression**: Adjusts bounding boxes using regression.
- **Problem**: Very slow due to the need to perform approximately 2,000 independent forward passes for each image.

Fast R-CNN

- **Process**:
    - **ConvNet**: Runs the whole image through a ConvNet to extract feature maps.
    - **RoI Projection**: Projects Regions of Interest (RoIs) onto the feature maps.
    - **CNN Per-Region Network**: Uses a CNN per region to classify the object category and refine bounding box offsets.

Faster R-CNN

- **Region Proposal Network (RPN)**: Inserts a Region Proposal Network (RPN) to predict proposals from features.
    - **Anchor Boxes**: The RPN imagines anchor boxes of fixed sizes at each point in the feature map.
    - **Object Detection**: Predicts whether each anchor contains an object (binary classification).
    - **Bounding Box Correction**: For positive boxes, predicts corrections from the anchor to the ground-truth box (regression of 4 numbers per pixel).
    - **Multiple Anchors**: Uses K different anchor boxes of different sizes/scales at each point.
    - **Proposal Selection**: Sorts the K*20*15 boxes by their "objectness" score and takes the top ~300 as proposals.
- **Advantage**: Makes CNN generate proposals.

Instance Segmentation: Mask R-CNN

- **Extension of Faster R-CNN**: Adds a small mask network that operates on each RoI and predicts a 28x28 binary mask.

Single-Shot Object Detectors: YOLO/SSD/RetinaNet

- **YOLO (You Only Look Once)**:
    - **Process**: Divides the image into a grid (e.g., 7x7).
    - **Base Boxes**: Uses a set of base boxes centered at each grid cell (B = 3).
    - **Regression**: Regresses from each of the B base boxes to a final box with 5 numbers (dx, dy, dh, dw, confidence).
    - **Classification**: Predicts scores for each of C classes (including background as a class).
    - **Single Stage**: Performs classification and bounding box prediction in a single stage.

YOLO Steps

1. **Image Splitting**: Splits the image into a grid of SxS equal cells.
2. **Bounding Box Regression**: For each cell, predicts bounding box parameters:

    - *Y = [pc, bx, by, bh, bw, c1, c2, ...]*.
    - *pc*: Probability of containing an object (confidence score).
    - *bx,by,bh,bw*: Bounding box coordinates with respect to the cell center.
    - *c1,c2,...*: Contained classes.
3. **Intersection over Union (IoU)**: Discards cells with too-small IoU values.
4. **Non-Maximum Suppression (NMS)**: Reduces noisy bounding boxes per object.
    - Selects the bounding box with the highest confidence score and suppresses all other bounding boxes with high IoU with respect to the selection.
    - Repeats until no bounding box can be selected anymore.

Evaluating Classification Performance

- **Confusion Matrix**:
    - **True Positives (TP)**: Correctly classified positives.
    - **False Negatives (FN)**: Actual positives classified incorrectly as negatives.
    - **False Positives (FP)**: Actual negatives classified incorrectly as positives.
    - **True Negatives (TN)**: Correctly classified negatives.
- **Metrics**:
    - **Precision**: *TP/(TP+FP)* - Proportion of all positive classifications that are actually positive.
    - **Recall**: *TP/(TP+FN)* - Proportion of all actual positives that were classified correctly as positives.
    - **Accuracy**: *(TP+TN)/(TP+TN+FP+FN)* - Proportion of all classifications that were correct.
    - **False Positive Rate**: *FP/(FP+TN)* - Proportion of all actual negatives that were classified incorrectly as positives.
- **Precision over Recall (P-R) Curve**:
    - Plots precision against recall for many confidence score thresholds.
    - A well-trained classifier has a large area under the P-R curve for all classes.

    - **Average Precision (AP)**: The area under the P-R curve (0..1).
    - **Mean Average Precision (mAP)**: The mean over AP for each class (0..1).

Vision Transformers

- **Transformers**: Used for sequences, can handle long-distance dependencies better than CNNs or RNNs.
- **Vision Transformers (ViTs)**: Used for computer vision tasks like classification and semantic labeling.
- **Process**:
    - Sequence of words = sequence of image patches.
    - Only the encoder part is used.
    - Flattened patches = reshape to 1D vector.
    - Linear projection = flattened patches * weight vector + bias (weight vector + bias has lower dimension and is learned).
    - Position encoded is added to ensure the transformer knows about the order of patches (= tokens).
    - For classification, add class token.
    - Attention is determined/learned between all tokens in the transformer encoder network (results are feature vectors).
    - Example classification: Output of transformer encoder (e.g., only class feature vector) = input of MPL / softmax for classification.
    - Key idea: Not only features are relevant (like in CNNs) but their relationship (context) to each other.

These notes cover the fundamental concepts, architectures, and evaluation methods in object detection.

Lecture 9

Here are detailed lecture notes on multi-view geometry, incorporating explanations and information from the sources:

Multi-View Geometry

Overview

- **Goal:** Estimate depth from multiple images. This is crucial for 3D scene understanding.
- **Depth from Stereo (Stereoscopic Images):** Achieved through using two views with slightly different perspectives.
- **Depth Map:** A representation where every pixel encodes the distance from the camera. Brighter pixels indicate closer points, while darker pixels indicate distant points.

Two Problems in Depth Estimation

1. **Finding Matching Features:** Identifying corresponding features (pixels/coordinates) in two or more images that represent the same surface point. Because scenery is recorded from different perspectives, corresponding pixels will be at different locations.
2. **Computing Depth of Corresponding Scene Points:** Computing the depth or 3D coordinate of the identified point using the matched features. Once matching is done for every pixel, a depth value can be assigned to each pixel.

Camera Model Recap

- A mathematical model that describes our camera is needed for depth reconstruction.
- **Pixel Coordinates**: Represented as (x, y) in an image.
- **3D Coordinates**: Represented as (X, Y, Z) in the 3D scene.
- **Camera Calibration**: Provides information about the camera.
  - **Intrinsic Parameters (K)**: Include focal length, field of view.
  - **Extrinsic Parameters**: Describe the camera's position and orientation in the world. They consist of:
    - **Rotation**
    - **Translation**
- **Projection Matrix (M)**: A combination of intrinsic and extrinsic parameters used to map 3D points (P) to 2D pixel coordinates (p).
  - $M = K[R|T]$

- **Perspective Projection**: Requires perspective division to account for the distance of the scene point.
- **Non-Linear Distortion**: Lens distortion, especially radial distortion, can make the model non-linear, requiring extended camera models.

Camera Calibration with OpenCV

- **Process**: Capturing images of a known calibration pattern.
- **Function**: cv.calibrateCamera() returns the camera matrix, distortion coefficients, rotation, and translation vectors.
- **Importance**: Essential when using multiple cameras for depth reconstruction to ensure correct scene point reconstruction in a common 3D coordinate system.

Epipolar Geometry: Feature Matching Problem

- **Epipolar Constraint**: A geometric constraint that simplifies the search for corresponding points from a 2D domain to a 1D domain. This is key for real-time depth estimation.
- **Epipoles (eL, eR)**: The intersection points of the line connecting the optical centers of the two cameras with the image planes.
- **Epipolar Line**: For a point in one image, the corresponding point in the other image must lie on a line called the epipolar line.
    - This constraint transforms the 2D search problem into a 1D search problem.

Mathematical Representation of Epipolar Constraints

- **Calibrated Case**:
    - Points (p, p') are on the normalized image plane.
    - Intrinsic parameters (K) are known.
    - Essential Matrix (ε): A 3x3 matrix that defines the epipolar constraint. The equation $l = \varepsilon p'$ describes the epipolar line.
- **Uncalibrated Case**:
    - Points (p, p') are on the physical image plane.
    - Intrinsic parameters (K) are unknown.

- Fundamental Matrix (F): A 3x3 matrix that defines the epipolar constraint. The equation *l = Fp'* describes the epipolar line.

Computing the Fundamental or Essential Matrix

- Requires a sufficient number of known corresponding point-pairs (>8).
- To solve for eight unknowns, you need eight or more point pairs.
- With the essential or fundamental matrix, the parameters of the epipolar line on the other side can be calculated for every point.

Image Rectification

- **Purpose**: Transforms images so that epipolar lines become horizontal scanlines.
- **Advantage**: Simplifies the search for matches to searching along horizontal lines (scanlines) with the same y-coordinate.

Stereoscopic Depth Reconstruction

- **Process**:
    1. **Calibrated Cameras**: Using two calibrated cameras with known intrinsic and extrinsic parameters.
    2. **Feature Matching**: Finding corresponding points (XL and XR) in the left and right images.
    3. **Depth Calculation**: Applying trigonometry using similar triangles to calculate the depth (Z) of the point.
- **Disparity**: The difference in pixel coordinates between matching points in the left and right images (XL - XR).
- **Depth Proportionality**: Depth (Z) is inversely proportional to disparity.

Measuring Disparity

- Can be done through:
    - **Manual Feature Matching**: Selecting corresponding points manually.
    - **SIFT Features**: Computing SIFT features and matching descriptors.
    - **Dense Matching**: Computing disparity for every pixel.
- **Disparity Maps**: Represent disparity values for each pixel.

    - These maps are used interchangeably with depth maps since depth can be derived from disparity given camera parameters.

Structure from Motion (SfM)

- **Goal**: Simultaneously estimate depth (structure) and camera pose (motion) from multiple images.
- **Advantage**: Does not require pre-calibration of cameras.
- **Process**:
    1. **Input**: A set of images captured from different viewpoints.
    2. **Simultaneous Estimation**: Solves for 3D structure and camera poses in a single step.
- **Projection Equation**: Extends the basic projection equation to multiple cameras and 3D points.
- SfM formulates a linear equation system to solve for camera poses and 3D point coordinates.

Solving Structure from Motion

- **Unknowns**:
    - 12*m coefficients in all m projection matrices (M).
    - 3*n coefficients in all n 3D points (P).
- **Equations**: 2 equations for all nm points.
- **Solvability Condition**: The system is solvable when 2*nm > 12*m + 3*n (e.g., m=2 and n=25).
- **Applications**: 3D reconstruction of cities and buildings from tourist photos.

These notes cover the core concepts of multi-view geometry, providing a foundation for understanding advanced topics and techniques in 3D scene reconstruction.

Lecture 10

Here are detailed lecture notes on 3D Vision, incorporating explanations and information from the sources:

3D Vision

Overview

- **Goal**: To enable computers to "see" and interpret the world in three dimensions.
- **Applications**: Object classification, segmentation, detection and instance segmentation.
- **Challenges**: Requires understanding and reconstructing depth information, which is not directly available in 2D images.

Depth Estimation Techniques

- **Depth from Stereo (and Multiple Views)**:
    - **Key Idea**: Use differences in perspective to estimate depth.
    - **Solution**: Develop models for feature extraction, matching, and depth computation.
- **Depth from Shading**:
    - **Key Idea**: Use differences in shading to estimate depth.
    - **Solution**: Develop models for inverse shading.
- **Depth from Defocus**:
    - **Key Idea**: Use differences in focus to estimate depth.
    - **Solution**: Develop models for focus estimation.
- **Depth from X**: General term for depth estimation techniques, where "X" can be shading, defocus, stereo, or multi-view.
    - Each approach has its own advantages and disadvantages.
- **Why Learn Depth Estimation?** Matching is hard due to the ill-posed problem. Factors such as reflectance, shape, and shading influence pixel colors, making it difficult to determine correct matches.

Representing Depth

- **Depth Map**: A 2D matrix where each pixel value represents the distance from the camera to the object at that pixel.
    - A depth map gives the distance from the camera to the object in the world at each pixel.

- **Voxel Grid**: A 3D matrix where each voxel stores properties of a point in space, such as color or opacity.
    - Represents a shape with a V x V x V grid of occupancies.
    - Conceptually simple, but requires high spatial resolution to capture fine structures.
- **Point Cloud**: A set of 3D points representing the shape of an object.
    - Can represent fine structures without a huge number of points.
- **Meshes**: Represent a 3D shape as a set of triangles.
    - Standard representation for graphics, explicitly represents 3D shapes, and can attach data on vertices.
- **Implicit Surfaces**: Learn a function to classify arbitrary 3D points as inside or outside the shape.
    - Application: Novel view synthesis.

Depth Maps in Detail

- **RGB-D Image**: An RGB image combined with a depth map.
    - This type of data can be recorded directly for some types of 3D sensors (e.g., Microsoft Kinect).
- **Predicting Depth Maps**:
    - Use a Convolutional Neural Network (CNN) to predict a depth image from an RGB input image.
    - Per-pixel loss (L2 distance) is used to train the network.
- **Scale-Depth Ambiguity**: A small, close object can look the same as a large, far object in a single image.
- **Scale Invariant Error**: A loss function that considers the global scale of a scene to address the scale-depth ambiguity.

Surface Normals

- For each pixel, the surface normal is the normal vector (unit vector perpendicular to the tangential surface) of the pixel.

- Surface normals can be computed from given depth (gradients) and vice versa (integration).
- **Predicting Surface Normals**:
  - Use a fully convolutional network to predict surface normals from an RGB input image.
  - Per-pixel loss function considers the solid angle difference between estimated and ground truth surface normals.

Voxel Grids in Detail

- Conceptually simple: just a 3D grid.
- Need high spatial resolution to capture fine structures.
- Scaling to high resolutions is nontrivial.
- **Generating Voxel Shapes**:
  - Use a 2D CNN to extract 2D features from an input image.
  - Use a 3D CNN to convert 2D features into voxels.
  - Train with per-voxel cross-entropy loss.

Point Clouds in Detail

- Can represent fine structures without huge numbers of points.
- Requires new architectures and losses.
- Doesn't explicitly represent the surface of the shape.
- **Generating Point Clouds**:
  - Use a 2D CNN to extract image features.
  - Use convolutional and fully connected branches to generate points.

Meshes in Detail

- Standard representation for graphics.
- Explicitly represents 3D shapes.
- Adaptive: Can represent flat surfaces efficiently and allocate more faces to areas with fine detail.

- Can attach data on vertices and interpolate over the whole surface.
- **Predicting Meshes**:
    - Use iterative refinement and graph convolution.
    - Project vertex-aligned features to image space.

Implicit Surfaces in Detail

- Learn a function to classify arbitrary 3D points as inside/outside the shape.
- Application: Novel View Synthesis.
- **Volume Rendering**:
    - Abstract away light sources and objects.
    - For each point in space, determine how much light it emits and how opaque it is.
    - Parameterize each ray as origin plus direction.
- **Neural Radiance Fields (NeRFs)**:
    - Use a fully connected network to input position and direction and output volume density and RGB color.
    - Train a neural network to input position and direction, and output opacity and color.
    - The network is retrained for every scene.

Pinhole Cameras (Recap)

- Light sources emit light.
- Objects in the world reflect light.
- Opaque objects block light.
- Transparent objects transmit light.
- Camera sensors record light through a pinhole.

These notes cover the core concepts of 3D vision, providing a foundation for understanding advanced topics and techniques in 3D scene reconstruction.