UNIT I

INTRODUCTION TO IMAGE FORMATION AND PROCESSING

Computer Vision - Geometric primitives and transformations - Photometric image formation - The digital camera - Point operators - Linear filtering - More neighborhood operators - Fourier transforms - Pyramids and wavelets - Geometric transformations - Global optimization.

1.1 COMPUTER VISION

Computer Vision is a multidisciplinary field that enables computers to interpret and understand visual information from the world, just like humans do with their eyes and brains. It involves processing, analyzing, and making sense of images and videos to extract meaningful insights, recognize objects, understand scenes, and even perform actions based on visual data. Computer Vision combines techniques from various domains, such as computer science, mathematics, physics, and psychology, to enable machines to perceive and interpret visual content.

1.1 Key Concepts in Computer Vision:

- Image Acquisition: Computer Vision starts with obtaining visual data. This can be achieved through various devices like cameras, scanners, drones, or sensors. Images are captured as grids of pixels, each pixel representing a tiny unit of color or intensity.
- Image Processing: This involves manipulating and enhancing images to improve their quality or extract relevant features. Common techniques include noise reduction, image sharpening, and contrast adjustment.
- Feature Extraction: In order to understand and identify objects in an image, relevant features must be extracted. Features can be edges, corners, textures, colors, shapes, or more complex patterns.
- Image Analysis and Understanding: This stage involves interpreting the extracted features to understand the content of the image. For instance, detecting objects, recognizing faces, or identifying specific patterns within an image.
- Object Detection: Identifying and localizing objects within an image or video stream. This is commonly used in applications like self-driving cars, where the system needs to detect pedestrians, vehicles, traffic signs, etc.
- Image Classification: Assigning a label or category to an image. For example, recognizing whether an image contains a cat or a dog.
- Object Tracking: Following the movement of an object over a sequence of frames in a video. This is crucial in applications like surveillance and robotics.

- Semantic Segmentation: Assigning a semantic label (e.g., "road," "building," "tree") to each pixel in an image.
- 3D Reconstruction: Creating a three-dimensional model of a scene or object from multiple images taken from different viewpoints.
- Motion Analysis: Analyzing the motion of objects in videos to understand patterns, behaviors, or anomalies. This is used in applications like action recognition and video surveillance.
- Deep Learning: In recent years, deep neural networks, particularly Convolutional Neural Networks (CNNs), have revolutionized computer vision. They can automatically learn hierarchical features from raw pixel data, enabling better performance in tasks like image recognition and object detection.
- Applications: Computer Vision has a wide range of applications across industries. Some examples include facial recognition, medical image analysis (like diagnosing diseases from medical scans), autonomous vehicles, augmented reality, quality control in manufacturing, and more.
- Challenges: Computer Vision faces challenges such as handling variations in lighting, viewpoint, and occlusions (parts of objects being hidden). Building models that generalize well across different conditions is a complex task.

Computer Vision has progressed significantly due to advancements in hardware capabilities, algorithm development, and the availability of large datasets. It's a rapidly evolving field with continuous research and innovation, making machines more capable of perceiving and understanding visual data, and bringing us closer to achieving tasks that were once exclusive to human vision.

1.2 GEOMETRIC PRIMITIVES AND TRANSFORMATION

Geometric primitives and transformations are fundamental concepts in computer vision that enable the representation, manipulation, and analysis of visual data, such as images and videos. These concepts help to describe the structure and relationships within visual scenes and facilitate tasks like object recognition, tracking, and 3D reconstruction.

1.2.1 Geometric Primitives

Geometric primitives are basic shapes or components that are used to represent objects and structures in a visual scene. They serve as building blocks for more complex shapes and allow us to describe the world in terms of simple elements. Some common geometric primitives include:

- Points: Represented by their coordinates (x, y), points are fundamental elements that define positions in a 2D or 3D space.
- Lines and Line Segments: Lines are defined by two points and extend infinitely in both directions. Line segments are portions of lines bounded by two points.
- Curves: Curves can be represented by mathematical equations and can

include circles, ellipses, and Bézier curves.

- Polygons: Closed shapes formed by connecting multiple line segments.
- Planes: Defined by three points or a point and a normal vector, planes are flat surfaces in 3D space.
- Solids: Represent three-dimensional objects, such as cubes, spheres, and cylinders.

1.2.2 Geometric Transformations:

Geometric transformations involve altering the position, size, orientation, or shape of geometric primitives in a visual scene. These transformations are used to manipulate visual data and enable various computer vision tasks. Some common types of geometric transformations include:

- Translation: Moves an object from one location to another by adding a constant value to its coordinates. Translation is often represented as (x + dx, y + dy) in 2D.
- Rotation: Rotates an object around a specified point or axis. The rotation angle determines the amount of rotation.
- Scaling: Changes the size of an object by multiplying its coordinates by scaling factors. Scaling can be uniform (equal in all directions) or non-uniform.
- Shearing: Skews or distorts an object along one axis, causing it to change shape.
- Reflection: Creates a mirror image of an object by reversing its coordinates along a specified axis.
- Affine Transformation: Combines translation, rotation, scaling, and shearing to perform more complex transformations while preserving parallel lines.
- Projective Transformation: Handles more general transformations that include perspective effects, such as foreshortening and vanishing points.

These geometric transformations are crucial for various computer vision tasks:

- Image Registration: Aligning multiple images or frames to a common reference frame.
- Object Recognition: Matching features or shapes in different images despite variations in viewpoint or scale.
- Camera Calibration: Determining the parameters of a camera's intrinsic and extrinsic properties to enable accurate mapping of 3D world points to 2D image points.
- Augmented Reality: Overlaying virtual objects onto the real world by transforming them according to the camera's perspective.
- 3D Reconstruction: Estimating the 3D structure of a scene by transforming 2D image points into 3D space.

In summary, geometric primitives and transformations play a crucial role in

computer vision by providing the tools to describe, manipulate, and analyze visual data in both 2D and 3D space, enabling a wide range of applications in the field.

1.3 PHOTOMETRIC IMAGE FORMATION

Photometric image formation, also known as the imaging process or image formation model, refers to the process by which light from a scene interacts with a camera's optics and sensor to create a digital image. Understanding this process is essential in computer vision for tasks such as image analysis, object recognition, and 3D reconstruction. The photometric image formation process involves several stages:

- Illumination: The scene is illuminated by a light source, either natural (such as sunlight) or artificial (like indoor lighting). Illumination affects the appearance of objects in the scene, their colors, and the intensity of light they reflect or transmit.
- Reflection and Transmission: When light interacts with objects in the scene, it can be reflected, absorbed, or transmitted. The properties of the materials in the scene determine how they respond to different wavelengths of light, leading to variations in appearance.
- Camera Optics: The light from the scene enters the camera through its lens system. The camera optics focus the incoming light onto the camera's sensor (or film in traditional cameras). The lens characteristics, including aperture size and focal length, affect how the light is captured.
- Sensor Response: The camera's sensor consists of individual photosensitive elements (pixels) that detect the incoming light. Each pixel responds to the intensity of light falling on it. The sensor's response can vary based on its sensitivity to different wavelengths and the exposure settings.
- Quantization: The analog signal generated by the sensor needs to be digitized for storage and processing. This involves converting the continuous range of light intensities into discrete values (usually in the form of digital numbers), a process known as quantization.
- Noise and Distortions: During the entire imaging process, various sources of noise and distortions can affect the captured image. These can include sensor noise, lens aberrations, and atmospheric effects.
- Color Representation: For color images, information about the distribution of different wavelengths of light is captured. This can be achieved using different techniques, such as using multiple sensors or filters to capture red, green, and blue channels, or using a single sensor with a color filter array (e.g., Bayer filter).

The entire process of photometric image formation is complex and influenced by numerous factors, such as the properties of the camera, the characteristics of the scene, the lighting conditions, and the interaction of light with different materials. Because of these complexities, computer vision researchers have developed mathematical models to

simulate and understand the image formation process accurately.

These models help in tasks like:

- Camera Calibration: Estimating the intrinsic and extrinsic parameters of a camera, which are essential for mapping 3D world points to 2D image coordinates.
- Image Enhancement: Correcting for noise, distortions, and other artifacts introduced during the imaging process.
- Color Correction: Adjusting colors to account for differences in lighting conditions or sensor characteristics.
- Material and Lighting Estimation: Inferring properties of materials and lighting in the scene based on captured images.
- Scene Understanding: Interpreting the observed images to infer the structure, content, and characteristics of the scene.

In summary, photometric image formation is a fundamental concept in computer vision that encompasses the entire process from illumination to the creation of a digital image. Understanding this process is essential for accurate analysis and interpretation of visual data in various applications.

1.4 THE DIGITAL CAMERA

A digital camera is a device that captures and records images in a digital format. It plays a significant role in computer vision as it serves as the primary tool for capturing visual data that can be analyzed, processed-, and understood by computers. The working principle of a digital camera involves several components and processes.

1.4.1 Components of a Digital Camera

- Lens: The lens gathers light from the scene and focuses it onto the camera's sensor. The quality and characteristics of the lens affect the overall image quality.
- Aperture: The aperture is an adjustable opening that controls the amount of light entering the camera. It affects the depth of field (the range of distances in focus) and the exposure of the image.
- Shutter: The shutter controls the duration of time that light is allowed to hit the sensor. It opens and closes to control the exposure of the image. A fast shutter speed freezes motion, while a slow shutter speed captures motion blur.
- ADC (Analog-to-Digital Converter): The analog signal generated by the sensor needs to be converted into digital form for processing and storage. The ADC assigns a digital value to each analog signal, which represents the intensity of light.
- Processor: The camera's processor handles various tasks, including image processing, compression, and the application of camera settings like white

balance and color correction.

- Memory Card: The digital image data is stored on a memory card, typically in formats like JPEG or RAW. This storage allows for easy transfer of images to other devices.
- Viewfinder or LCD Screen: The viewfinder (optical or electronic) allows the photographer to compose the shot, while the LCD screen displays the captured image and camera settings

1.4.2 Working Principle:

- Light Capture: When you press the shutter button, the camera's aperture opens, allowing light to pass through the lens. The lens focuses this light onto the image sensor.
- Sensor Exposure: The light falling on the sensor generates an electrical charge at each pixel site. The amount of charge corresponds to the intensity of light.
- Analog-to-Digital Conversion: The analog charges from the sensor are converted into digital values using the ADC. Each pixel's charge is quantized into a specific digital number, representing its brightness.
- Image Processing: The camera's processor applies various adjustments to the digital image, including white balance, color correction, noise reduction, and sharpening.
- Storage: The processed image data is stored on a memory card in a specific file format (such as JPEG or RAW). This allows easy transfer and sharing of the images.
- Display: The captured image can be displayed on the camera's LCD screen for review.

Digital cameras provide a flexible and efficient way to capture visual data for computer vision applications. They allow for precise control over exposure settings, image quality, and other parameters, making them valuable tools for capturing images that can be analyzed by computers to extract information, recognize objects, and understand scenes.

1.5 POINTER OPERATORS

In computer vision, a point operator, often referred to as a point processing or point-wise operation, is a type of image processing operation that operates on individual pixels of an image independently, without considering their neighbors. The operation modifies the pixel values based on a predefined rule or function, transforming the pixel values one by one. Point operators are simple yet powerful tools used for tasks like image enhancement,

contrast adjustment, and intensity transformation.

1.5.1 Key Concepts of Point Operators

- Pixel Transformation: In a point operator, each pixel in the input image is transformed into a new value in the output image based on a specific rule or function. The transformation function defines how the pixel values are modified.
- Function Mapping: Point operators involve mapping the original pixel values to new values using a mathematical function. The function's behavior determines how the pixel values change.
- Local Operation: Since point operators consider only the individual pixel being processed, they are local operations. They don't take into account the surrounding pixels, which makes them computationally efficient.
- Intensity Transformation: Point operators are commonly used for intensity transformations, such as adjusting the brightness, contrast, or gamma correction of an image.

1.5.2 Common Point Operator Examples

- Brightness Adjustment: A simple point operator involves adding or subtracting a constant value from all pixel values in the image. This operation shifts the intensity levels without changing the overall contrast.
- Contrast Enhancement: By applying a linear scaling factor to the pixel values, you can expand or compress the range of intensity values, enhancing the image's contrast.
- Gamma Correction: Adjusting the intensity levels using a power function can correct the gamma characteristics of an image and improve its overall appearance.
- Thresholding: Creating a binary image by assigning one value to pixels above a certain threshold and another value to pixels below the threshold.
- Negative Image: Inverting the pixel values, resulting in a negative image.
- Histogram Equalization: A more advanced point operator that redistributes the intensity values to enhance the image's contrast and reveal details.

1.5.3 Applications:

Point operators are widely used in various computer vision tasks and image processing applications:

- Image Enhancement: Modifying pixel values to improve the overall visual quality of an image.
- Histogram Manipulation: Changing the distribution of pixel intensities in the image to improve contrast or match a desired histogram.

- Image Correction: Adjusting images to account for variations in lighting conditions or sensor characteristics.

- Preprocessing: Preparing images for further analysis by normalizing intensity ranges or reducing noise.
- Artistic Effects: Creating stylistic effects by manipulating pixel values, such as generating sepia tones or applying filters.

In summary, point operators are basic but essential tools in image processing and computer vision. They enable quick adjustments to pixel values, allowing for image enhancement and transformation without the need for complex neighborhood-based operations.

1.6 LINEAR FILTERING

Linear filtering is a fundamental concept in computer vision and image processing that involves applying a convolution operation to an image using a filter kernel. This operation is used to perform tasks like noise reduction, edge detection, and image smoothing. Linear filtering plays a crucial role in enhancing images, extracting features, and preparing images for further analysis.

1.6.1 Key Concepts of Linear Filtering

- Filter Kernel: A filter kernel, also known as a convolution kernel or mask, is a small matrix of numeric values. The size of the kernel determines the extent of the neighborhood considered during the filtering operation. The values within the kernel define the coefficients used in the convolution operation.
- Convolution Operation: The convolution operation involves placing the filter kernel over each pixel in the image and calculating a weighted sum of the pixel values and their neighbors according to the values in the kernel. The result of this sum becomes the new pixel value in the output image.
- Pixel-wise Operation: Linear filtering is a pixel-wise operation, meaning that each pixel's value in the output image is calculated independently based on its corresponding neighborhood in the input image.

1.6.2 Types of Linear Filtering

- Smoothing (Low-pass) Filters: These filters are used to reduce noise and smooth out image details. Common smoothing filters include the Gaussian filter, which assigns higher weights to central pixels and lower weights to neighboring pixels, and the mean filter, which replaces each pixel value with the average of its neighbors.
- Edge Detection (High-pass) Filters: These filters emphasize the sharp changes or edges in an image. The Sobel, Prewitt, and Roberts operators are commonly used edge detection filters.

- Gradient Filters: These filters highlight intensity changes in different directions. They are used for tasks like edge detection and texture analysis.

- Embossing Filters: These filters create a 3D effect by highlighting the edges in the image. They can make objects appear raised or engraved.
- Box Blur and Motion Blur Filters: These filters apply uniform blurring in a specific direction, simulating motion blur or smoothing.

1.6.3 Application of Linear Filtering

- Noise Reduction: Smoothing filters are used to reduce random noise in images, making them clearer and easier to analyze.
- Edge Detection: High-pass filters are employed to highlight edges, which are important features in image analysis and object detection.
- Image Enhancement: Filtering can enhance specific image details or structures to improve the visual quality or prepare images for further processing.
- Feature Extraction: Certain filters highlight textures, patterns, or other characteristics, which can be useful for identifying objects or regions of interest.
- Preprocessing: Filtering can be a crucial step before applying more complex computer vision algorithms, improving their accuracy and performance.

In summary, linear filtering is a foundational technique in computer vision that involves convolving an image with a filter kernel to achieve various effects such as noise reduction, edge detection, and image enhancement. It's a versatile tool used in various image processing tasks to extract meaningful information and prepare images for analysis.

1.7 MORE NEIGHBOURHOOD OPERATORS

Neighborhood operators, also known as local operators or spatial filters, are techniques used in computer vision and image processing to manipulate the pixel values of an image based on the values of their neighboring pixels. Unlike point operators, which only consider individual pixel values, neighborhood operators consider the context of nearby pixels to perform operations that involve local patterns and structures. These operators are particularly useful for tasks like image enhancement, edge detection.

1.7.1 Key Concepts of Neighborhood Operators

- Neighborhood: A neighborhood refers to a specific region around a pixel in an image. It's defined by a certain number of rows and columns (usually an odd number) that determines the size of the local region. The pixels within this region are used to compute the output value for the central pixel.
- Kernel or Mask: The kernel, also referred to as the mask or window, is a small matrix that defines the weights or coefficients for each pixel within the neighborhood. These weights determine how much influence each pixel has on

the computation of the central pixel's new value.

- Convolution Operation: The process involves sliding the kernel over each pixel in the image. At each position, the elements of the kernel are multiplied with the corresponding pixel values in the neighborhood, and the sum of these products becomes the new value for the central pixel.

1.7.2 Common Neighborhood Operators

- Blurring (Smoothing) Filters: Neighborhood operators can perform blurring or smoothing to reduce noise and make images more visually coherent. Gaussian blur and median filters are examples of smoothing operators.
- Edge Detection Filters: These operators emphasize the differences in intensity between neighboring pixels, making edges more visible. Examples include the Sobel, Prewitt, and Laplacian of Gaussian filters.
- Gradient Filters: These operators compute the gradient of intensity changes in different directions, helping to identify edges and boundaries. The Sobel and Scharr filters are popular gradient operators.
- Noise Removal Filters: Some neighborhood operators, like the median filter, are effective at removing salt-and-pepper noise or other types of noise from images.
- Morphological Operators: These operators, such as erosion and dilation, are used for tasks like image segmentation and noise reduction by modifying the shapes of objects in an image.

1.7.3 Application of Neighborhood Operators

- Image Enhancement: By considering local patterns and structures, neighborhood operators can enhance image features, improve contrast, and reveal hidden details.
- Edge Detection: Operators like Sobel and Prewitt are used to detect edges by highlighting rapid intensity changes between neighboring pixels.
- Feature Extraction: Neighborhood operators can help extract textures, patterns, and other features from images, aiding in object recognition and classification.
- Image Restoration: These operators can restore images by reducing noise, smoothing, and correcting artifacts.
- Morphological Operations: Erosion, dilation, opening, and closing are used in image segmentation and shape analysis.

Neighborhood operators are essential tools in computer vision and image processing, as they allow us to analyze and manipulate images based on local patterns and context, which is often crucial for understanding the content of images and extracting meaningful information.

1.8 FOURIER TRANSFORMS

Fourier Transforms are fundamental mathematical tools used in various fields, including computer vision, to analyze signals and images in the frequency domain. They enable the decomposition of complex signals into simpler sinusoidal components, revealing underlying patterns and structures that might not be immediately apparent in the spatial domain.

1.8.1 Key Concepts of Fourier Transforms

- Spatial Domain vs. Frequency Domain: In the spatial domain, data is represented as intensity values in an image. The frequency domain, on the other hand, represents data in terms of its frequency components, indicating how much each frequency contributes to the overall signal.
- Complex Exponential Basis: Fourier Transforms use complex exponential functions as the basis to represent signals. These basis functions are sinusoidal waves of various frequencies.
- Transform and Inverse Transform: A Fourier Transform converts a signal from the spatial domain to the frequency domain. The Inverse Fourier Transform converts it back from the frequency domain to the spatial domain.
- Magnitude and Phase: The magnitude of the Fourier Transform represents the strength of each frequency component, while the phase indicates the position of each component in time.
- **1.8.2 Applications of Fourier Transforms in Computer Vision:**
- Image Compression: Fourier Transforms help compress images by representing them in the frequency domain and removing high-frequency components that contribute less to human perception.
- Pattern Recognition and Texture Analysis: Frequency information can be vital for detecting patterns and analyzing textures in images. Certain frequency components are characteristic of specific patterns and textures.
- Edge Detection: High-frequency components in the Fourier Transform can help identify edges and abrupt changes in the image.
- Image Registration: Fourier Transforms are used for aligning and registering images by analyzing their frequency content.
- Holography and 3D Imaging: Fourier Transforms are utilized in holography to record and reconstruct 3D images.
- Image Filtering and Enhancement: Fourier Transforms are used to design and

apply filters in the frequency domain, enabling tasks such as noise reduction, sharpening, and blurring.

- Image Deconvolution: Fourier Transforms play a role in image deconvolution, a technique used to recover the original image from a blurred or distorted version.
- Frequency Analysis: Fourier Transforms provide insight into the frequency content of images, which is important for tasks like speech and audio analysis.

18.3 Fast Fourier Transforms

Fast Fourier Transforms is an algorithm that efficiently computes the Fourier Transform and its inverse. It significantly speeds up the process and is widely used due to its computational efficiency.

In summary, Fourier Transforms are powerful tools in computer vision that allow us to analyze images and signals in terms of their frequency components. They find applications in various tasks, including filtering, enhancement, compression, and pattern recognition, helping us gain insights into the underlying structure of visual data.

1.9 PYRAMIDS AND WAVELETS

Pyramids and wavelets are two techniques used in computer vision and image processing for multi-scale analysis of images. They allow for the representation of images at different scales, which is particularly useful for tasks like image compression, object detection, texture analysis, and feature extraction. Both pyramids and wavelets provide a way to capture information at various levels of detail, helping to analyze images efficiently.

1.9.1 Pyramids

Pyramids are hierarchical representations of images, organized into multiple levels or scales. Each level contains a version of the image at a different resolution. There are two main types of pyramids:

- Gaussian Pyramid: In a Gaussian pyramid, images are repeatedly filtered with a Gaussian filter to reduce their size and resolution. This process effectively blurs the image and reduces high-frequency details.
- Laplacian Pyramid: A Laplacian pyramid is constructed by taking the difference between consecutive levels of the Gaussian pyramid. This pyramid represents the details that are removed during the blurring process of the Gaussian pyramid. It captures the high-frequency information and can be used to reconstruct the original image

1.9.2 Wavelets

Wavelets are mathematical functions that can represent signals and images in both time and frequency domains. Wavelet transformations break down an image into

different frequency components at different scales. The concept of wavelets is similar to pyramids, but they use a different mathematical approach.

- Continuous Wavelet Transform (CWT): This transform analyzes an image using a continuous wavelet function that varies in scale and position. It provides a detailed view of the frequency content of an image at various scales.
- Discrete Wavelet Transform (DWT): DWT divides an image into approximate and detailed components at different scales. It operates on discrete levels, making it suitable for computer applications.

1.9.3 Applications of Pyramids and Wavelets:

- Image Compression: Both pyramids and wavelets are used in image compression algorithms like JPEG2000. They allow for efficient representation of images by discarding less significant details at lower scales.
- Texture Analysis: Multi-scale analysis helps in capturing texture features at different levels, which can be useful for texture classification.
- Object Detection: Multi-scale analysis aids in detecting objects of different sizes within images. Objects at different scales are often represented more clearly in different levels of the pyramid or wavelet transformation.
- Feature Extraction: Pyramids and wavelets can help extract features at various scales, improving the accuracy of feature-based techniques.
- Image Denoising: Multi-scale analysis can help in removing noise by isolating high-frequency details, allowing for effective noise reduction.
- Image Restoration: Wavelet and pyramid techniques can be used to restore images by enhancing the important features while suppressing noise.

Both pyramids and wavelets offer valuable tools for multi-scale analysis in computer vision, enabling the extraction of information from images at different levels of detail and contributing to various image processing and analysis tasks.

1.10 GEOMETRIC TRANSFORMATIONS

Geometric transformations in computer vision involve manipulating the spatial properties of images, such as their position, orientation, scale, and perspective. These transformations are used to align images, correct distortions, and project 3D scenes onto 2D images. Geometric transformations play a crucial role in various computer vision applications, including image registration, object tracking, augmented reality, and 3D reconstruction.

1.10.1 Common Geometric Transformations:

- Translation: Shifting an image by a certain amount in the horizontal and vertical directions. This is often represented as (x + dx, y + dy), where (dx, dy) are the

translation parameters.

- Rotation: Rotating an image around a specified point by a given angle.
- Scaling: Changing the size of an image by applying scaling factors independently along the horizontal and vertical axes.
- Shearing: Distorting an image by slanting it along one axis while keeping the other axis unchanged.
- Reflection: Flipping an image along a specified axis, creating a mirror image.
- Affine Transformation: A combination of translation, rotation, scaling, and shearing. It preserves straight lines but not necessarily angles and lengths.
- Projective Transformation (Homography): A more general transformation that includes perspective effects, such as mapping 3D scenes onto 2D images. It preserves collinearity (straight lines) but not necessarily angles, lengths, or parallelism.

1.10.2 Application of Geometric Transformations

- Image Registration: Aligning multiple images from different viewpoints or time instances. This is crucial in applications like medical imaging, satellite imagery, and image mosaicking.
- Object Tracking: Adjusting the position and orientation of a tracking window as an object moves in a video sequence.
- Augmented Reality: Overlaying virtual objects onto the real world by transforming them according to the camera's perspective.
- Image Warping: Deforming images for artistic effects or simulating changes in viewing angles.
- Camera Calibration: Determining the intrinsic and extrinsic parameters of a camera to map 3D points to 2D image coordinates.
- 3D Reconstruction: Estimating the 3D structure of a scene by mapping 2D points from multiple images to a common 3D coordinate system.

Geometric transformations are essential for adapting images to different contexts, correcting distortions introduced by camera optics, and aligning images for further analysis. They are a fundamental part of computer vision algorithms that involve spatial relationships between images and scenes.

1.11 GLOBAL OPTIMIZATION

Global optimization in computer vision refers to the process of finding the optimal solution across an entire search space, considering all possible solutions and avoiding local minima or maxima. It's a challenging problem since many real-world optimization tasks involve complex, multi- dimensional, and often nonlinear objective functions. Global

optimization methods aim to find the best possible solution regardless of the initial starting point.

1.11.1 Key Challenges in Global Optimization:

- High-Dimensional Spaces: Many optimization problems in computer vision involve a large number of variables, making the search space high-dimensional. Traditional optimization methods struggle to handle such complexities.
- Nonlinearity: Objective functions in computer vision can be highly nonlinear, with multiple local optima and global optima scattered throughout the search space.
- Non-Convexity: The presence of multiple local optima that are not necessarily convex adds complexity to the optimization problem.
- Noise and Uncertainty: In real-world scenarios, objective functions often include noise or uncertainty, making it difficult to distinguish true optima from spurious ones.
- Computationally Intensive: Finding a global optimum can be computationally expensive, especially in high-dimensional spaces.

1.11.2 Methods for Global Optimization in Computer Vision:

- Simulated Annealing: This probabilistic technique mimics the annealing process in metallurgy. It allows the optimization algorithm to explore the search space by accepting uphill moves with decreasing probability. This helps escape local minima and eventually converge to a global optimum.
- Genetic Algorithms: Inspired by natural evolution, genetic algorithms use concepts of selection, mutation, and crossover to evolve a population of potential solutions. They explore the search space broadly, enabling them to find global optima.
- Differential Evolution (DE): DE is a population-based optimization algorithm that creates new candidate solutions by combining differences between existing solutions. It balances exploration and exploitation to find global optima.
- Bayesian Optimization: Bayesian optimization uses a probabilistic model to predict the value of the objective function at different points. It aims to optimize the acquisition function, which balances exploration and exploitation, to find the best solution.
- Random Search: Despite its simplicity, random search can be surprisingly effective for global optimization, especially in high-dimensional spaces. It involves randomly sampling points from the search space and evaluating the objective function at those points.
- Evolutionary Strategies: These algorithms employ strategies such as mutation, crossover, and selection to evolve a population of solutions. They can be adapted

to handle complex, noisy, and high-dimensional optimization problems.

- Metaheuristic Algorithms: Metaheuristic algorithms, like ant colony optimization, harmony search, and cuckoo search, are designed to tackle complex optimization problems by simulating natural or social behaviors.

Global optimization methods in computer vision are crucial for parameter tuning, model selection, and solving optimization tasks where local optima are not sufficient. These methods help in finding solutions that are closer to the true global optimum across a wide range of applications, from image processing to machine learning.

PART A

1. Define geometric primitives in computer vision.
2. What is the pinhole camera model in photometric image formation?
3. Name two common point operators.
4. State the effect of histogram equalization.
5. What kernel is used for Gaussian linear filtering?
6. Define edge detection in the context of neighborhood operators.
7. What does the Fourier transform do in image processing?
8. Explain the concept of image pyramid.
9. What is an affine transformation?
10. Give one example of a global optimization method in vision.

PART B

1. Explain how geometric primitives like points, lines, and circles are represented mathematically.
2. Describe the photometric image formation pipeline from scene irradiance to pixel value.
3. Compare contrast stretching vs histogram equalization.
4. Derive the 2D Gaussian filter equation and its significance.
5. Discuss two neighborhood operators for edge detection and their response characteristics.
6. Explain how the 2D Fourier transform is used to analyze image frequency content.
7. Illustrate the construction and use of a Laplacian pyramid in image compression.
8. Show mathematically how scaling and rotation are combined in a single affine transformation matrix.
9. Discuss how RANSAC performs robust fitting of geometric primitives.
10. Explain belief propagation as used in global optimization for stereo vision.

PART C

1. Design an algorithm to detect and parameterize circles in an image using geometric primitives and Hough Transform.
2. Analyze errors introduced by lens distortion during photometric image formation and propose correction methods.
3. Derive, implement, and compare point operators for contrast adjustment and gamma correction.

4. Develop a complete edge-preserving smoothing filter using non-linear neighborhood operators (e.g., bilateral filter), explaining each step.
5. Design and analyze a frequency-domain filter (e.g., notch or bandpass) via Fourier transforms to remove periodic noise.
6. Build and evaluate a multiscale object detection method using image pyramids and sliding windows.
7. Formulate the mathematical framework for combining rotation, translation, and scaling in 2D/3D geometric transformations, and demonstrate with an example.
8. Design a graph-cut optimization for image segmentation: define energy terms, construct the graph, and solve a sample case.
9. Compare and contrast wavelet vs pyramid decompositions for image compression—include design criteria, transform steps, and a performance evaluation.
10. Propose a global optimization framework for stereo depth estimation: include energy model (data + smoothness), minimization method (e.g., belief propagation, graph cuts), and discuss computational complexity and accuracy trade-offs.

UNIT II

FEATURE DETECTION, MATCHING AND SEGMENTATION

Points and patches - Edges - Lines - Segmentation - Active contours - Split and merge - Mean shift and mode finding - Normalized cuts - Graph cuts and energy-based methods.

2.1 POINTS AND PATCHES

2.1.1 Points

Points in the context of computer vision typically refer to specific locations or coordinates within an image.

Usage: Points are often used as key interest points or landmarks. These can be locations with unique features, such as corners, edges, or distinctive textures.

Applications: Points are crucial in various computer vision tasks, including feature matching, image registration, and object tracking. Algorithms often detect and use points as reference locations for comparing and analyzing images.

2.1.2 Patches:

Definition: Patches are small, localized regions or segments within an image.

Usage: In computer vision, patches are often extracted from images to focus on specific areas of interest. These areas can be defined by points or other criteria.

Applications: Patches are commonly used in feature extraction and representation. Instead of analyzing entire images, algorithms may work with patches to capture detailed information about textures, patterns, or structures within the image. Patches are also utilized in tasks like image classification and object recognition.

While "points" usually refer to specific coordinates or locations within an image, "patches" are small, localized regions or segments extracted from images. Both concepts are fundamental in various computer vision applications, providing essential information for tasks such as image analysis, recognition, and understanding. Points and patches play a crucial role in the extraction of meaningful features that contribute to the overall interpretation of visual data by computer vision systems.

2.2 EDGES

In image processing and computer vision, "edges" refer to significant changes in intensity or color within an image. Edges often represent boundaries or transitions between different objects or regions in an image. Detecting edges is a fundamental step in various computer vision tasks.

Definition: - An edge is a set of pixels where there is a rapid transition in intensity or color. This transition can occur between objects, textures, or other features in an image.Importance:

Edges are crucial for understanding the structure of an image. They represent boundaries between different objects or regions, providing valuable information for object recognition and scene understanding.

2.2.1 Edge Detection:

Edge detection is the process of identifying and highlighting edges within an image. Various edge detection algorithms, such as the Sobel operator, Canny edge detector, and Laplacian of Gaussian (LoG), are commonly used for this purpose.

Applications: Object Recognition: Edges help in defining the contours and shapes of objects, facilitating their recognition.

Image Segmentation: Edges assist in dividing an image into meaningful segments or regions.

Feature Extraction: Edges are important features that can be extracted and used in higher-level analysis.

Image Compression: Information about edges can be used to reduce the amount of data needed to represent an image.

2.2.1 Types of Edges:

- Step Edges: Sharp transitions in intensity.
- Ramp Edges: Gradual transitions in intensity.
- Roof Edges: A combination of step and ramp edges.

Challenges: Edge detection may be sensitive to noise in the image, and selecting an appropriate edge detection algorithm depends on the characteristics of the image and the specific application.

2.3 LINES

In the context of image processing and computer vision, "lines" refer to straight or curved segments within an image.Detecting and analyzing lines is a fundamental aspect of Image understanding and is important in various computer vision

applications. Here are key points about lines:

Definition: A line is a set of connected pixels with similar characteristics, typically representing a continuous or approximate curve or straight segment within an image.

Line Detection: Line detection is the process of identifying and extracting lines from an image. Hough Transform is a popular technique used for line detection, especially for straight lines.

2.3.1 Types of Lines:

- Straight Lines: Linear segments with a constant slope.
- Curved Lines: Non-linear segments with varying curvature.
- Line Segments: Partial lines with a starting and ending point.

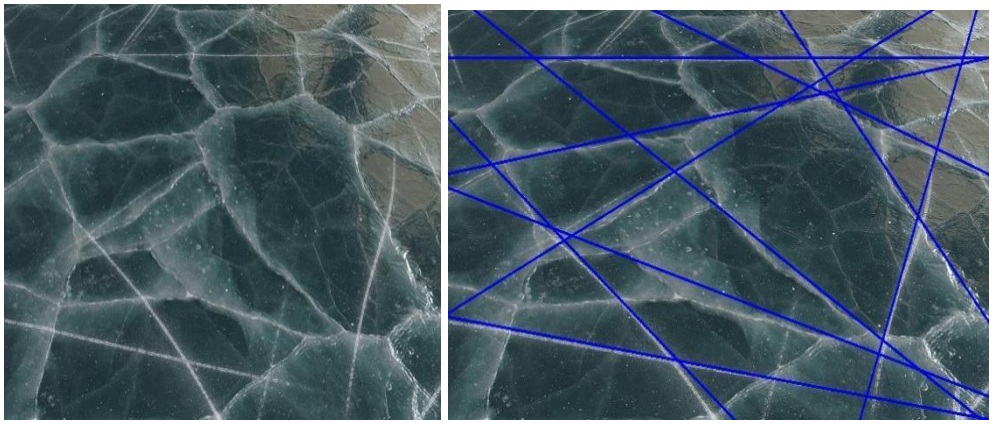

Fig 2.1 Line Detection

2.4 SEGMENTATION

Image segmentation is a computer vision task that involves partitioning an image into meaningful and semantically coherent regions or segments. The goal is to group together pixels or regions that share similar visual characteristics, such as color, texture, or intensity. Image segmentation is a crucial step in various computer vision applications as it provides a more detailed and meaningful understanding of the content within an image. Here are key points about image segmentation:

Definition: Image segmentation is the process of dividing an image into distinct and meaningful segments. Each segment typically corresponds to a region or object in the image.

Purpose: Segmentation is used to simplify the representation of an image, making it easier to analyze and understand. It helps in identifying and delineating different objects or regions within the image.

2.4.1 Types of Segmentation:

- Semantic Segmentation: Assigning a specific class label to each pixel in the image, resulting in a detailed understanding of the object categories present.
- Instance Segmentation: Identifying and delineating individual instances of objects within the image. Each instance is assigned a unique label.
- Boundary or Edge-based Segmentation: Detecting edges or boundaries between different regions in the image.
- Region-based Segmentation: Grouping pixels into homogeneous regions based on similarity criteria.Algorithms:

Various algorithms are used for image segmentation, including region-growing methods, clustering algorithms (e.g., K-means), watershed algorithms, and deep learning-based approaches using convolutional neural networks (CNNs).

2.4.2 Applications:

- Object Recognition: Segmentation helps in isolating and recognizing individual objects within an image.
- Medical Imaging: Identifying and segmenting structures or anomalies in medical images.
- Autonomous Vehicles: Segmenting the environment to detect and understand objects on the road.
- Satellite Image Analysis: Partitioning satellite images into meaningful regions for land cover classification.
- Robotics: Enabling robots to understand and interact with their environment by segmenting objects and obstacles.

2.4.3 Challenges:

Image segmentation can be challenging due to variations in lighting, complex object shapes, occlusions, and the presence of noise in the image.

2.4.4 Evaluation Metrics:

Common metrics for evaluating segmentation algorithms include Intersection over Union (IoU), Dice coefficient, and Pixel Accuracy.

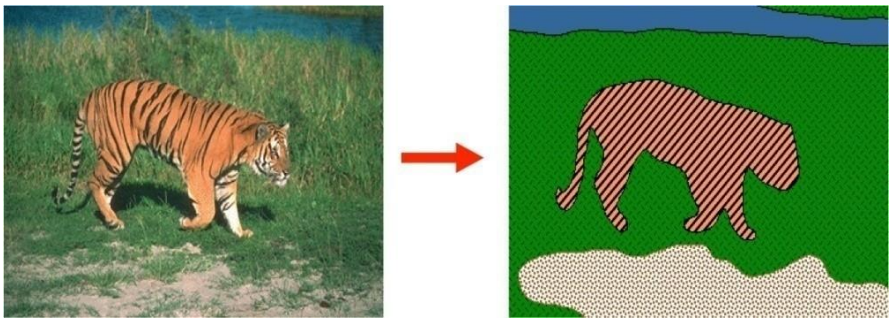

Fig 2.2 Image Segmentation

Image segmentation is a fundamental task in computer vision that involves dividing an image into meaningful segments to facilitate more advanced analysis and understanding. The choice of segmentation method depends on the specific characteristics of the images and the requirements of the application.

2.5 ACTIVE CONTOURS

Active contours, also known as snakes, are a concept in computer vision and image processing that refer to deformable models used for image segmentation. The idea behind active contours is to evolve a curve or contour within an image in a way that captures the boundaries of objects or regions of interest. These curves deform under the influence of internal forces .

2.5.1 Key features of active contours include:

- Initialization: Active contours are typically initialized near the boundaries of the objects to be segmented. The initial contour can be a closed curve or an open curve depending on the application.

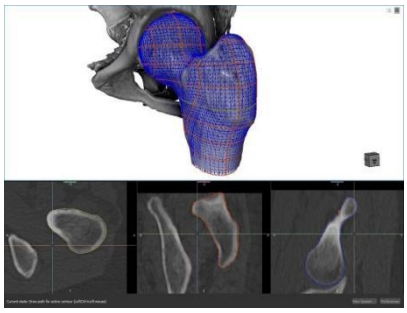

Fig 2.3 Active Contours

- Energy Minimization: The evolution of the active contour is guided by an energy function that combines internal and external forces. The goal is to minimize this energy to achieve an optimal contour that fits the boundaries of the object.
- Internal Forces: Internal forces are associated with the deformation of the contour itself. They include terms that encourage smoothness and continuity of the curve. The internal energy helps prevent the contour from oscillating or exhibiting unnecessary deformations.
- External Forces: External forces are derived from the image data and drive the contour toward the boundaries of objects. These forces are attracted to features such as edges, intensity changes, or texture gradients in the image.
- Snakes Algorithm: The snakes algorithm is a well-known method for active contour modeling. It was introduced by Michael Kass, Andrew Witkin, and Demetri Terzopoulos in 1987. The algorithm involves iterative optimization of the energy function to deform the contour.

2.5.2 Applications:

Active contours are used in various image segmentation applications, such as medical image analysis, object tracking, and computer vision tasks where precise delineation of object boundaries is required.

2.5.3 Challenges:

Active contours may face challenges in the presence of noise, weak edges, or complex object structures. Careful parameter tuning and initialization are often required.

2.5.4 Variations:

There are variations of active contours, including geodesic active contours and level-set methods, which offer different formulations for contour evolution and segmentation.

Active contours provide a flexible framework for interactive and semi-automatic segmentation by allowing users to guide the evolution of the contour. While they have been widely used, the choice of segmentation method depends on the specific characteristics of the images and the requirements of the application.

2.6 SPLIT AND MERGE

Split and Merge is a recursive image segmentation algorithm that divides an image into homogeneous regions based on certain criteria. The primary idea behind the algorithm is to recursively split an image into smaller blocks until certain conditions are met, and then merge those blocks if they are sufficiently homogeneous. This process continues iteratively until the desired level of segmentation is achieved.

2.6.1 Algorithm:

- Splitting Phase: The algorithm starts with the entire image as a single block. It evaluates a splitting criterion to determine if the block is sufficiently homogeneous or should be split further.
- Merging Phase: Once the recursive splitting reaches a certain level or the splitting criterion is no longer satisfied, the merging phase begins.
- Homogeneity Criteria: The homogeneity of a block or region is determined based on certain criteria, such as color similarity, intensity, or texture. For example, blocks may be considered homogeneous if the variance of pixel values within the block is below a certain threshold.
- Recursive Process: The splitting and merging phases are applied recursively, leading to a hierarchical segmentation of the image.

2.6.2 Applications:

Split and Merge can be used for image segmentation in various applications, including object recognition, scene analysis, and computer vision tasks where delineation of regions is essential.

The Split and Merge algorithm provides a way to divide an image into regions of homogeneous content, creating a hierarchical structure. While it has been used historically, more recent image segmentation methods often involve advanced techniques, such as machine learning-based approaches (e.g., convolutional neural networks) or other region-growing algorithms.

2.7 MEAN SHIFT AND MODE FINDING

Mean Shift is a non-parametric clustering algorithm commonly used for image segmentation and object tracking. The algorithm works by iteratively shifting a set of data points towards the mode or peak of the data distribution. In the context of image processing, MeanShift can be applied to group pixels with similar characteristics into coherent segments.

2.7.1 Algorithm:

- Kernel Density Estimation: The algorithm begins by estimating the probability density function (PDF) of the input data points. This is typically done using a kernel function, such as a Gaussian kernel.
- Initialization: Each data point is considered as a candidate cluster center.
- Mean Shift Iterations: For each data point, a mean shift vector is computed. The mean shift vector points towards the mode or peak of the underlying data distribution.
- Convergence Criteria: The algorithm converges when the mean shift vectors become very small or when the points reach local modes in the data distribution.
- Cluster Assignment: After convergence, data points that converge to the same mode are assigned to the same cluster.

Mean Shift has been successfully applied to image segmentation, where it effectively groups pixels with similar color or intensity values into coherent segments.

2.7.2 Mode Finding:

In statistics and data analysis, a "mode" refers to the value or values that appear most frequently in a dataset. Mode finding, in the context of Mean Shift or other clustering algorithms, involves identifying the modes or peaks in the data distribution.

2.7.3 Mean Shift:

- The mean shift process involves iteratively shifting towards the modes of the underlying data distribution.
- Each cluster is associated with a mode, and the mean shift vectors guide the data points toward these modes during the iterations.

Mean Shift is an algorithm that performs mode finding to identify clusters in a dataset. In image processing, it is often used for segmentation by iteratively shifting towards modes in the color or intensity distribution, effectively grouping pixels into coherent segments.

2.8 NORMALIZED CUTS

Normalized Cuts is a graph-based image segmentation algorithm that seeks to divide an image into meaningful segments by considering both the similarity between pixels and the dissimilarity between different segments. It was introduced by Jianbo Shi and Jitendra Malik in 2000 and has been widely used in computer vision and image processing.

2.8.1 Algorithm:

- Graph Representation: The image is represented as an undirected graph, where each pixel is a node in the graph, and edges represent relationships between pixels. Edges are weighted based on the similarity between pixel values.

- Affinity Matrix: An affinity matrix is constructed to capture the similarity between pixels. The entries of this matrix represent the weights of edges in the graph, and the values are determined by a similarity metric, such as color similarity or texture similarity.
- Segmentation Objective: The goal is to partition the graph into two or more segments in a way that minimizes the dissimilarity between segments and maximizes the similarity within segments.
- Normalized Cuts Criteria: The algorithm formulates the segmentation problem using a normalized cuts criteria, which is a ratio of the sum of dissimilarities between segments to the sum of similarities within segments.
- Eigenvalue Problem: The optimization problem involves solving an eigenvalue problem derived from the affinity matrix. The eigenvectors corresponding to the smallest eigenvalues provide information about the optimal segmentation.
- Recursive Approach: To achieve multi-segmentation, the algorithm employs a recursive approach. After the initial segmentation, each segment is further divided into sub-segments by applying the same procedure recursively.
- Advantages: Normalized Cuts is capable of capturing both spatial and color information in the segmentation process.
- Challenges: The computational complexity of solving the eigenvalue problem can be a limitation, particularly for large images.

Normalized Cuts has been widely used in image segmentation tasks, especially when capturing global structures and relationships between pixels is essential. It has applications in computer vision, medical image analysis, and other areas where precise segmentation is crucial.

2.9 GRAPH CUTS AND ENERGY-BASED METHODS

Graph cuts and energy-based methods are widely used in computer vision and image processing for solving optimization problems related to image segmentation. These methods often leverage graph representations of images and use energy functions to model the desired properties of the segmentation.

2.9.1 Graph Cuts:

Graph cuts involve partitioning a graph into two disjoint sets such that the cut cost (the sum of weights of edges crossing the cut) is minimized. In image segmentation, pixels are represented as nodes, and edges are weighted based on the dissimilarity between pixels.

- Graph Representation: Each pixel is a node, and edges connect adjacent pixels. The weights of edges reflect the dissimilarity between pixels (e.g., color, intensity).

- Energy Minimization: The problem is formulated as an energy minimization task, where the energy function includes terms encouraging similarity within segments and dissimilarity between segments.
- Binary Graph Cut: In the simplest case, the goal is to partition the graph into two sets (foreground and background) by finding the cut with the minimum energy.
- Multiclass Graph Cut: The approach can be extended to handle multiple classes or segments by using techniques like the normalized cut criterion.

2.9.1.1 Applications:

Graph cuts are used in image segmentation, object recognition, stereo vision, and other computer vision tasks.

2.9.2 Energy-Based Methods:

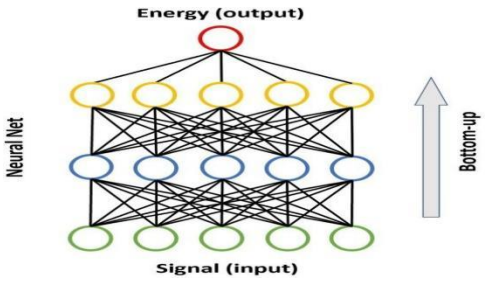

Fig 2.4 Energy-Based Methods

Energy-based methods involve formulating an energy function that measures the quality of a particular configuration or assignment of labels to pixels. The optimization process aims to find the label assignment that minimizes the energy.

- Energy Function: The energy function is defined based on factors such as data terms (measuring agreement with observed data) and smoothness terms (encouraging spatial coherence).
- Unary and Pairwise Terms: Unary terms are associated with individual pixels and capture the likelihood of a pixel belonging to a particular class. Pairwise terms model relationships between neighboring pixels and enforce smoothness.
- Markov Random Fields (MRFs) and Conditional Random Fields (CRFs): MRFs and CRFs are common frameworks for modeling energy-based methods. MRFs consider local interactions, while CRFs model dependencies more globally.

2.9.2.1 Applications:

Energy-based methods are applied in image segmentation, image denoising, image restoration, and various other vision tasks.

PART A

1. Define a point feature and a patch in computer vision.
2. What is an edge in an image, and why is it important?
3. Name two common edge detectors.
4. Define a line in the context of image analysis.
5. What is image segmentation?
6. What is the principle behind active contours (or snakes)?
7. Briefly describe the split-and-merge segmentation method.
8. What is mean shift used for?
9. Define normalized cuts in image segmentation.
10. What are graph cuts used for in computer vision?

PART B

1. Explain how point features are detected and described using patches.
2. Compare Sobel vs. Canny edge detectors, detailing strengths and limitations.
3. Describe the Hough Transform for line detection, including its parameterization and voting scheme.
4. Outline the split-and-merge algorithm step-by-step with a simple example.
5. Explain active contours (snake model): include energy terms, internal vs. external forces, and balloon model variant.
6. Detail the mean shift algorithm: explain kernel density estimation, the mean shift vector, and an application.
7. Describe normalized cuts: how an image is represented as a graph, the cut criterion, and the role of eigen-decomposition.
8. Explain graph cuts for segmentation: mapping energy minimization to min-cut/max-flow and submodularity.
9. Compare normalized cuts vs. graph cuts in terms of optimization goals, computational cost, and segmentation quality.
10. Detail gradient vector flow (GVF) snakes: how edge gradients are extended and why this improves active contour convergence.

PART C

1. Design an end-to-end system that uses point-based descriptors (e.g., SIFT/SURF) for feature matching, followed by RANSAC-based line fitting. Explain each stage and evaluate failure modes.
2. Implement and compare edge detectors (Sobel, Canny, Laplacian of Gaussian) on noisy images. Include derivations, parameter tuning, and performance metrics (precision, recall).
3. Build a line-detection pipeline using Hough Transform—derive the voting space, thresholding, and post-processing to filter false positives.

4. Develop a segmentation algorithm combining split-and-merge with region merging heuristics. Present pseudocode, complexity analysis, and visualize results.
5. Design an active contour (snake) framework: formulate the energy functional, discretize the curve, implement gradient descent, and demonstrate on medical imagery. Address initialization and parameter tuning.
6. Extend snake with GVF: derive GVF equations, implement force field computation, and compare segmentation robustness against standard snakes.
7. Formulate mean shift segmentation: include bandwidth selection, convergence analysis, and apply to color and spatial domains. Evaluate contour preservation.
8. Apply normalized cuts to segment real-world scenes: derive Laplacian matrix, compute eigenvectors, thresholding method, and discuss segmentation quality vs. spectral graph clustering.
9. Design an energy-based segmentation using graph cuts: define unary and pairwise terms, build the graph, solve via min-cut, and test on foreground-background tasks (e.g., GrabCut).
10. Compare normalized cuts, graph cuts, and mean shift: discuss theoretical underpinnings (spectral vs. combinatorial vs. density-based), computational complexity, segmentation accuracy, and practical application scenarios

*

UNIT III

FEATURE-BASED ALIGNMENT & MOTION ESTIMATION

2D and 3D feature-based alignment – Pose estimation – Geometric intrinsic calibration – Triangulation – Two-frame structure from motion – Factorization – Bundle adjustment – Constrained structure and motion – Translational alignment – Parametric motion – Spline- based motion – Optical flow – Layered motion.

3.1 2D AND 3D FEATURE-BASED ALIGNMENT

Feature-based alignment is the problem of estimating the motion between two or more sets of matched 2D or 3D points.

Transform	Matrix	Parameters p	Jacobian J
translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	(t_x, t_y)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Euclidean	$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	(t_x, t_y, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$
similarity	$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	(t_x, t_y, a, b)	$\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$
affine	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	$(t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})$	$\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$
projective	$\begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$	$(h_{00}, h_{01}, \ldots, h_{21})$	(see Section 8.1.3)

Table 3.1 2D Coordinate

2D alignment using least squares Given a set of matched feature points {(xi , x 0 i)} and a planar parametric transformation1 of the form how can we produce the best estimate of the motion parameters p? The usual way to do this is to use least squares, i.e., to minimize the sum of squared residuals.

The primary goals of 2D feature-based alignment in computer vision presented in points:

- Accurate Alignment: The primary aim is to accurately align two images by finding the transformation parameters that best match corresponding features between

$$E_{\mathrm{LS}} = \sum_i \|\mathbf{r}_i\|^2 = \sum_i \|\mathbf{f}(\mathbf{x}_i; \mathbf{p}) - \mathbf{x}'_i\|^2,$$

them.

- Feature Detection: Identify distinctive key points or features in each image that are robust to changes in scale, rotation, and illumination.

- Feature Description: Describe the detected key points in a way that allows for reliable matching across images despite variations in viewpoint or lighting conditions.
- Feature Matching: Match corresponding features between the images to establish correspondences, which are used to estimate the transformation that aligns the images.
- Transformation Estimation: Estimate the transformation parameters (such as translation, rotation, scale, etc.) that align one image with another based on the matched feature correspondences.
- Robustness: Ensure that the alignment process is robust to noise, occlusions, and other artifacts present in the images.
- Efficiency: Perform the alignment process efficiently to allow for real-time or near-real-time performance in applications such as augmented reality or video processing.
- Applicability: Enable alignment for various computer vision tasks such as image stitching, object recognition, image registration, and augmented reality.

$$
\begin{aligned}
E_{\mathrm{LLS}} &= \sum_i \|\mathbf{J}(\mathbf{x}_i)\mathbf{p} - \Delta\mathbf{x}_i\|^2 \\
&= \mathbf{p}^T \left[\sum_i \mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i)\right] \mathbf{p} - 2\mathbf{p}^T \left[\sum_i \mathbf{J}^T(\mathbf{x}_i)\Delta\mathbf{x}_i\right] + \sum_i \|\Delta\mathbf{x}_i\|^2 \\
&= \mathbf{p}^T\mathbf{A}\mathbf{p} - 2\mathbf{p}^T\mathbf{b} + c.
\end{aligned}
$$

The minimum can be found by solving the symmetric positive definite (SPD) system of normal Equations

$$\mathbf{A}\mathbf{p} = \mathbf{b},$$

$$\mathbf{A} = \sum_i \mathbf{J}^T(\mathbf{x}_i)\mathbf{J}(\mathbf{x}_i)$$

Where

is called the Hessian and b $\mathbf{b} = \sum_i \mathbf{J}^T(\mathbf{x}_i)\Delta\mathbf{x}_i$ For the case of pure translation, the resulting equations have a particularly simple form, i.e., the translation is the average translation between corresponding points or, equivalently, the translation of the point centroids.

Uncertainty weighting. The above least squares formulation assumes that all feature points are matched with the same accuracy. This is often not the case, since certain points may fall into more textured regions than others. If we associate a scalar variance estimate with each correspondence, we can minimize the weighted least squares problem instead

Fig 3.1 Panography

$$E_{\mathrm{WLS}} = \sum_i \sigma_i^{-2} \|\mathbf{r}_i\|^2.$$

$$E_{\mathrm{CWLS}} = \sum_i \|\mathbf{r}_i\|^2_{\mathbf{\Sigma}_i^{-1}} = \sum_i \mathbf{r}_i^T \mathbf{\Sigma}_i^{-1} \mathbf{r}_i = \sum_i \sigma_n^{-2} \mathbf{r}_i^T \mathbf{A}_i \mathbf{r}_i.$$

3.1.1 Application: Panography

One of the simplest (and most fun) applications of image alignment is a special form of image stitching called panography. In a panograph, images are translated and optionally rotated and scaled before being blended with simple averaging.

In most of the examples seen on the web, the images are aligned by hand for best artistic effect.

Consider a simple translational model. We want all the corresponding features in different images to line up as best as possible. Let tj be the location of the jth image coordinate frame in the global composite frame and xij be the location of the ith matched feature in the jth image. In order to align the images, we wish to minimize the least squares error

$$E_{\mathrm{PLS}} = \sum_{ij} \|(\mathbf{t}_j + \mathbf{x}_{ij}) - \mathbf{x}_i\|^2,$$

To minimize the non-linear least squares problem, we iteratively find an update _p to the current parameter estimate p by minimizing

$$
\begin{aligned}
E_{\mathrm{NLS}}(\Delta\mathbf{p}) &= \sum_i \|\mathbf{f}(\mathbf{x}_i;\mathbf{p}+\Delta\mathbf{p}) - \mathbf{x}_i'\|^2 \\
&\approx \sum_i \|\mathbf{J}(\mathbf{x}_i;\mathbf{p})\Delta\mathbf{p} - \mathbf{r}_i\|^2 \\
&= \Delta\mathbf{p}^T \left[\sum_i \mathbf{J}^T\mathbf{J}\right] \Delta\mathbf{p} - 2\Delta\mathbf{p}^T \left[\sum_i \mathbf{J}^T\mathbf{r}_i\right] + \sum_i \|\mathbf{r}_i\|^2 \\
&= \Delta\mathbf{p}^T\mathbf{A}\Delta\mathbf{p} - 2\Delta\mathbf{p}^T\mathbf{b} + c,
\end{aligned}
$$

For the other 2D motion models, the derivatives in Table 8.1 are all fairly straightforward, except for the projective 2D motion (homography), which arises in image-stitching applications

$$
x' = \frac{(1+h_{00})x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + 1} \quad \text{and} \quad y' = \frac{h_{10}x + (1+h_{11})y + h_{12}}{h_{20}x + h_{21}y + 1}.
$$

The Jacobian is therefore

$$
\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{p}} = \frac{1}{D}\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y \end{bmatrix},
$$

3.1.2 3D Alignment

Instead of aligning 2D sets of image features, many computer vision applications require the alignment of 3D points. In the case where the 3D transformations are linear in the motion parameters, e.g., for translation, similarity, and affine, regular least squares can be used.

$$
E_{\mathrm{R3D}} = \sum_i \|\mathbf{x}_i' - \mathbf{R}\mathbf{x}_i - \mathbf{t}\|^2,
$$

The case of rigid (Euclidean) motion, which arises more frequently and is often called the absolute orientation problem, requires slightly different techniques. If only scalar weightings are being used (as opposed to full 3D per-point anisotropic covariance estimates), the weighted centroids of the two point clouds

One commonly used technique is called the orthogonal Procrustes algorithm and involves computing the singular value decomposition (SVD) of the 3 * 3 correlation matrix.

The rotation matrix is then obtained as (Verify this for yourself when ^x0 = R^x.) Another technique is the absolute orientation algorithm for estimating the unit quaternion corresponding to the rotation matrix R, which involves forming a 4 _ 4 matrix from the entries in C and then finding the eigenvector associated with its largest positive eigenvalue.

3.2 POSE ESTIMATION

This pose estimation problem is also known as extrinsic calibration, as opposed to the intrinsic calibration of internal camera parameters such as focal length. The problem of recovering pose from three correspondences, which is the minimal amount of information necessary, is known as the perspective-3-point-problem (P3P), 2 with extensions to larger numbers of points collectively known as PnP.

In this section, we look at some of the techniques that have been developed to solve such problems, starting with the direct linear transform (DLT), which recovers a 3 _ 4 camera matrix, followed by other "linear" algorithms, and then looking at statistically optimal iterative algorithms.

3.2.1 Linear Algorithms

The simplest way to recover the pose of the camera is to form a set of rational linear equations analogous to those used for 2D motion estimation from the camera matrix form of perspective projection.

$$x_i = \frac{p_{00}X_i + p_{01}Y_i + p_{02}Z_i + p_{03}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}}$$
$$y_i = \frac{p_{10}X_i + p_{11}Y_i + p_{12}Z_i + p_{13}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}},$$

where (xi; yi) are the measured 2D feature locations and (Xi; Yi;Zi) are the known 3D feature locations. As with, this system of equations can be solved in a linear fashion for the unknowns in the camera matrix P by multiplying the denominator on both sides of the equation. Because P is unknown up to a scale, we can either fix one of the entries, e.g., p23 = 1, or find the smallest singular vector of the set of linear equations. The resulting algorithm is called the direct linear transform (DLT) and is commonly attributed to (For a more in-depth discussion, To compute the unknowns in P, at least six correspondences between 3D and 2D locations must be known.

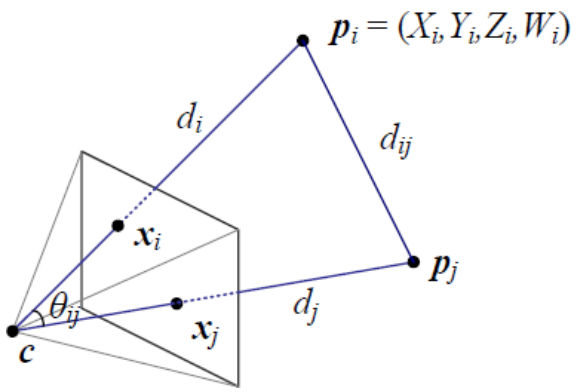

Fig 3.2 Pose estimation by the direct linear transform and by measuring visual angles and distances between pairs of points.

As with the case of estimating homographies , more accurate results for the entries in P can be obtained by directly minimizing the set of Equations using non-linear least squares with a small number of iterations. Note that instead of taking the ratios of the X=Z and Y=Z values as in it is also possible to take a cross product of the 3-vector (xi; yi; 1) image measurement and the 3-D ray (X; Y;Z) and set the three elements of this cross-product to 0. The resulting three equations, when interpreted as a set of least squares constraints, in effect compute the squared sine of the angle between the two rays.

3.2.2 CNN-based Pose Estimation

As with other areas on computer vision, deep neural networks have also been applied to pose estimation. Some representative papers include for object pose estimation, and papers such as and discussed in Section on location recognition. There is also a very active community around estimating pose from RGB-D images, with the most recent papers evaluated on the BOP (Benchmark for 6DOF Object Pose)

3.2.3 Iterative Non-Linear Algorithms

The most accurate and flexible way to estimate pose is to directly minimize the

$$E_{\mathrm{NLP}} = \sum_i \rho\left(\frac{\partial \mathrm{f}}{\partial \mathrm{R}}\Delta\mathrm{R} + \frac{\partial \mathrm{f}}{\partial \mathrm{t}}\Delta\mathrm{t} + \frac{\partial \mathrm{f}}{\partial \mathrm{K}}\Delta\mathrm{K} - \mathrm{r}_i\right),$$

squared (or robust) reprojection error for the 2D points as a function of the unknown

$$\mathbf{x}_i = \mathbf{f}(\mathbf{p}_i; \mathbf{R}, \mathbf{t}, \mathbf{K})$$

pose parameters in (R; t) and optionally K using non-linear least squares. We can write the projection equations as and iteratively minimize the robustified linearized reprojection errors.

$$\mathbf{r}_i = \tilde{\mathbf{x}}_i - \hat{\mathbf{x}}_i$$

Where is the current residual vector (2D error in predicted position) and the partial derivatives are with respect to the unknown pose parameters (rotation, translation, and optionally calibration). The robust loss function is used to reduce the influence of outlier correspondences. The resulting projection equations can be written as

$$\begin{aligned}
\mathbf{y}^{(1)} &= \mathbf{f}_{\mathrm{T}}(\mathbf{p}_i; \mathbf{c}_j) = \mathbf{p}_i - \mathbf{c}_j, \\
\mathbf{y}^{(2)} &= \mathbf{f}_{\mathrm{R}}(\mathbf{y}^{(1)}; \mathbf{q}_j) = \mathbf{R}(\mathbf{q}_j)\,\mathbf{y}^{(1)}, \\
\mathbf{y}^{(3)} &= \mathbf{f}_{\mathrm{P}}(\mathbf{y}^{(2)}) = \frac{\mathbf{y}^{(2)}}{z^{(2)}}, \\
\mathbf{x}_i &= \mathbf{f}_{\mathrm{C}}(\mathbf{y}^{(3)}; \mathbf{k}) = \mathbf{K}(\mathbf{k})\,\mathbf{y}^{(3)}.
\end{aligned}$$

Note that in these equations, we have indexed the camera centers cj and camera rotation quaternions qj by an index j, in case more than one pose of the calibration object is being used.

The advantage of this chained set of transformations is that each one has a simple partial derivative with respect both to its parameters and to its input. Thus, once the predicted value of ~xi has been computed based on the 3D point location pi and the current values of the pose parameters (cj ; qj ; k), we can obtain all of the required partial derivatives using the chain rule

$$\frac{\partial \mathbf{r}_i}{\partial \mathbf{p}^{(k)}} = \frac{\partial \mathbf{r}_i}{\partial \mathbf{y}^{(k)}} \frac{\partial \mathbf{y}^{(k)}}{\partial \mathbf{p}^{(k)}},$$

Where p(k) indicates one of the parameter vectors that is being optimized. (This same “trick” is used in neural networks as part of back propagation. The one special case in this formulation that can be considerably simplified is the computation of the rotation update. Instead of directly computing the derivatives of the 3 _ 3 rotation matrix R(q) as a function of the unit quaternion entries, you can prepend the incremental rotation matrix $\Delta \mathbf{R}(\boldsymbol{\omega})$ given in to the current rotation matrix and compute the partial derivative of the transform with respect to these parameters, which results in a simple cross product of the backward chaining partial derivative and the outgoing 3D vector,

The inference of human pose (head, body, and limb locations and attitude) from a single images can be viewed as yet another kind of segmentation task. We have already discussed some pose estimation techniques on pedestrian detection section. Starting with the seminal work 2D and 3D pose detection and estimation rapidly developed as an active research area, with important advances and datasets

3.3 GEOMETRIC INTRINSIC CALIBRATION

Intrinsic calibration estimates the camera's internal parameters that map 3D world points onto the 2D image plane. These core parameters include focal lengths, principal point, skew, and lens distortion coefficients (radial + tangential). The output is typically the camera intrinsic matrix:

3.3.1 Image acquisition with calibration patterns

Patterns like checkerboards or circle grids are photographed from multiple angles

3.3.2 Detect feature correspondences

Extract precise points in 2D images and their known 3D coordinates on the pattern

3.4 TRIANGULATION

The problem of determining a point’s 3D position from a set of corresponding image. Locations and known camera positions is known as triangulation. This problem is the converse of the pose estimation problem.

One of the simplest ways to solve this problem is to find the 3D point p that lies closest to all of the 3D rays corresponding to the 2D matching feature locations observed by cameras The nearest point to p on this ray, which we denote as minimizes the distance.

$$\|\mathbf{q}_j - \mathbf{p}\|^2 = \|\mathbf{c}_j + d_j\hat{\mathbf{v}}_j - \mathbf{p}\|^2,$$

$$d_j = \hat{\mathbf{v}}_j \cdot (\mathbf{p} - \mathbf{c}_j).$$

which has a minimum at Hence,

$$\mathbf{q}_j = \mathbf{c}_j + (\hat{\mathbf{v}}_j\hat{\mathbf{v}}_j^T)(\mathbf{p} - \mathbf{c}_j) = \mathbf{c}_j + (\mathbf{p} - \mathbf{c}_j)_{\|},$$

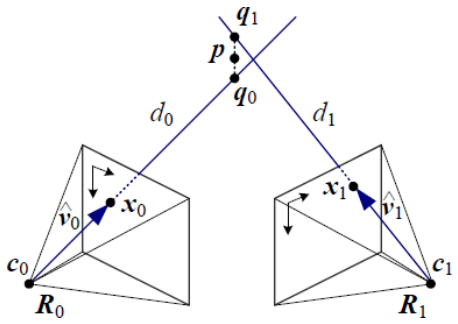

Fig 3.3 Triangulations

in the notation of Equation, and the squared distance between p and qj is

$$r_j^2 = \|(\mathbf{I} - \hat{\mathbf{v}}_j\hat{\mathbf{v}}_j^T)(\mathbf{p} - \mathbf{c}_j)\|^2 = \|(\mathbf{p} - \mathbf{c}_j)_{\perp}\|^2.$$

The optimal value for p, which lies closest to all of the rays, can be computed as a regular least squares problem by summing over all the r2 j and finding the optimal value of p,

$$\mathbf{p} = \left[\sum_j (\mathbf{I} - \hat{\mathbf{v}}_j\hat{\mathbf{v}}_j^T)\right]^{-1} \left[\sum_j (\mathbf{I} - \hat{\mathbf{v}}_j\hat{\mathbf{v}}_j^T)\mathbf{c}_j\right].$$

An alternative formulation, which is more statistically optimal and which can produce significantly better estimates if some of the cameras are closer to the 3D point than others, is to minimize the residual in the measurement equations

$$x_j = \frac{p_{00}^{(j)}X + p_{01}^{(j)}Y + p_{02}^{(j)}Z + p_{03}^{(j)}W}{p_{20}^{(j)}X + p_{21}^{(j)}Y + p_{22}^{(j)}Z + p_{23}^{(j)}W}$$

$$y_j = \frac{p_{10}^{(j)}X + p_{11}^{(j)}Y + p_{12}^{(j)}Z + p_{13}^{(j)}W}{p_{20}^{(j)}X + p_{21}^{(j)}Y + p_{22}^{(j)}Z + p_{23}^{(j)}W},$$

$$\{p_{00}^{(j)} \ldots p_{23}^{(j)}\}$$

where (xj ; yj) are the measured 2D feature locations and are the known entries in camera matrix Pj. As with, this set of non-linear equations can be converted into a linear least squares problem by multiplying both sides of the denominator, again resulting in the direct linear transform (DLT) formulation.

Note that if we use homogeneous coordinates p = (X; Y;Z;W), the resulting set of equations is homogeneous and is best solved as a singular value decomposition (SVD) or eigenvalue problem (looking for the smallest singular vector or eigenvector). If we set W = 1, we can use regular linear least squares, but the resulting system may be singular or poorly conditioned, i.e., if all of the viewing rays are parallel, as occurs for points far away from the camera.

For this reason, it is generally preferable to parameterize 3D points using homogeneous coordinates, especially if we know that there are likely to be points at greatly varying distances from the cameras. Of course, minimizing the set of observations using non-linear least squares is preferable to using linear least squares, regardless of the representation chosen

3.5 TWO-FRAME STRUCTURE FROM MOTION

So far in our study of 3D reconstruction, we have always assumed that either the 3D point positions or the 3D camera poses are known in advance. In this section, we take our first look at structure from motion, which is the simultaneous recovery of 3D structure and pose from image correspondences.

In particular, we examine techniques that operate on just two frames with point correspondences. We divide this section into the study of classic “n- point” algorithms, special (degenerate) cases, projective (uncalibrated) reconstruction, and self-calibration for cameras whose intrinsic calibrations are unknown.

3.5.1 Eight, seven and five-point algorithms

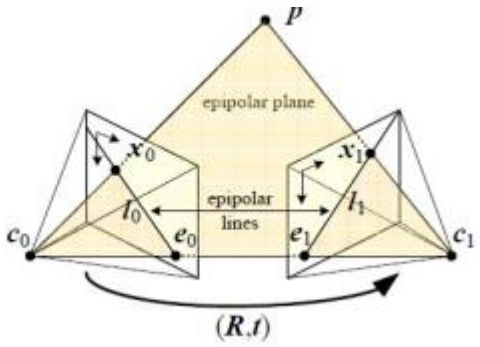

Fig 3.4 Two-Frame Structure From Motion

Consider which shows a 3D point p being viewed from two cameras whose relative position can be encoded by a rotation R and a translation t. As we do not know anything about the camera positions, without loss of generality, we can set the first camera at the origin c0 = 0 and at a canonical orientation R0 = I. The 3D point p0 =

d0^x0 observed in the first image at location ^x0 and at a z distance of d0 is mapped into the second image by the transformation.

$$d_1\hat{\mathbf{x}}_1 = \mathbf{p}_1 = \mathbf{R}\mathbf{p}_0 + \mathbf{t} = \mathbf{R}(d_0\hat{\mathbf{x}}_0) + \mathbf{t},$$

$$d_1[\mathbf{t}]_\times\hat{\mathbf{x}}_1 = d_0[\mathbf{t}]_\times\mathbf{R}\hat{\mathbf{x}}_0.$$

Where $\hat{\mathbf{x}}_j = \mathbf{K}_j^{-1}\mathbf{x}_j$ are the (local) ray direction vectors. Taking the cross product of the two (Interchanged) sides with t in order to annihilate it on the right-hand side yields.

$$\mathbf{E} = \mathbf{T}_1^T\tilde{\mathbf{E}}\mathbf{T}_0.$$

Taking the dot product of both sides with ^x1 yields,

$$d_0\hat{\mathbf{x}}_1^T([\mathbf{t}]_\times\mathbf{R})\hat{\mathbf{x}}_0 = d_1\hat{\mathbf{x}}_1^T[\mathbf{t}]_\times\hat{\mathbf{x}}_1 = 0,$$

because the right-hand side is a triple product with two identical entries. (Another way to say this is that the cross product matrix [t]_ is skew symmetric and returns 0 when pre- and post-multiplied by the same vector.).

3.5.2 Self-calibration

The results of structure from motion computation are much more useful if a metric reconstruction is obtained, i.e., one in which parallel lines are parallel, orthogonal walls are at right angles, and the reconstructed model is a scaled version of reality.

Over the years, a large number of self-calibration (or auto-calibration) techniques have been developed for converting a projective reconstruction into a metric one, which is equivalent to recovering the unknown calibration matrices Kj associated with each image.In situations where additional information is known about the scene, different methods may be employed.

For example, if there are parallel lines in the scene, three or more vanishing points, which are the images of points at infinity, can be used to establish the homography for the plane at infinity, from which focal lengths and rotations can be recovered. If two or more finite orthogonal vanishing points have been observed, the single-image calibration method based on vanishing points can be used instead

3.5.3 Application: View morphing

An interesting application of basic two-frame structure from motion is view morphing (also known as view interpolation,), which can be used to generate a smooth 3D animation from one view of a 3D scene to another. To create such a transition, you must first smoothly interpolate the camera matrices, i.e., the camera positions, orientations, and focal lengths. While simple linear interpolation can be used (representing rotations as quaternions.a more pleasing effect is obtained by easing in

and easing out the camera parameters,

To generate in-between frames, either a full set of 3D correspondences needs to be established or 3D models (proxies) must be created for each reference view. describes several widely used approaches to this problem. One of the simplest is to just triangulate the set of matched feature points in each image, e.g., using Delaunay triangulation. As the 3D points are re- projected into their intermediate views, pixels can be mapped from their original source images to their new views using affine or projective mapping.

3.6 FACTORIZATION

While two-frame techniques are useful for reconstructing sparse geometry from stereo image pairs and for initializing larger-scale 3D reconstructions, most applications can benefit from the much larger number of images that are usually available in photo collections and videos of scenes.

In this section, we briefly review an older technique called factorization, which can provide useful solutions for short video sequences, and then turn to the more commonly used bundle adjustment approach, which uses non-linear least squares to obtain optimal solutions under general camera configurations

When processing video sequences, we often get extended feature tracks from which it is possible to recover the structure and motion using a process called factorization. Consider the tracks generated by a rotating ping pong ball, which has been marked with dots to make its shape and motion more discernable .

Once the rotation matrices and 3D point locations have been recovered, there still exists a bas-relief ambiguity, i.e., we can never be sure if the object is rotating left to right or if its depth reversed version is moving the other way. (This can be seen in the classic rotating Necker Cube visual illusion.) Additional cues, such as the appearance and disappearance of points, or perspective effects, both of which are discussed below, can be used to remove this ambiguity.

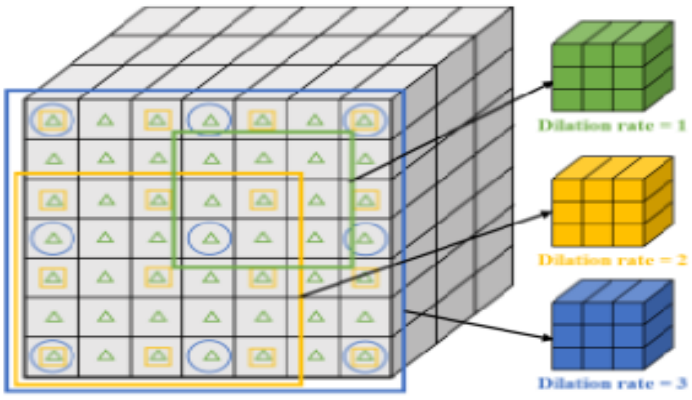

Fig 3.5 Factorization

For motion models other than pure orthography, e.g., for scaled orthography or paraperspective, the approach above must be extended in the appropriate manner. Such techniques are relatively straightforward to derive from first principles; more details can be found in papers that extend the basic factorization approach to these more flexible models. Additional extensions of the original factorization algorithm include multi-body rigid motion, sequential updates to the factorization, the addition of lines and planes and re-scaling the measurements to incorporate individual location uncertainties.

A disadvantage of factorization approaches is that they require a complete set of tracks, i.e., each point must be visible in each frame, for the factorization approach to work and deal with this problem by first applying factorization to smaller denser subsets and then using known camera (motion) or point (structure) estimates to hallucinate additional missing values, which allows them to incrementally incorporate more features and cameras.

Perspective and projective factorization Another disadvantage of regular factorization is that it cannot deal with perspective cameras. One way to get around this problem is to perform an initial affine (e.g., orthographic) reconstruction and to then correct for the perspective effects in an iterative manner (Christy and Horaud 1996). This algorithm usually converges in three to five iterations, with the majority of the time spent in the SVD computation.

3.7 BUNDLE ADJUSTMENT

As we have mentioned several times before, the most accurate way to recover structure and motion is to perform robust non-linear minimization of the measurement (re-projection) errors, which is commonly known in the photogrammetry (and now computer vision) communities as bundle adjustment.

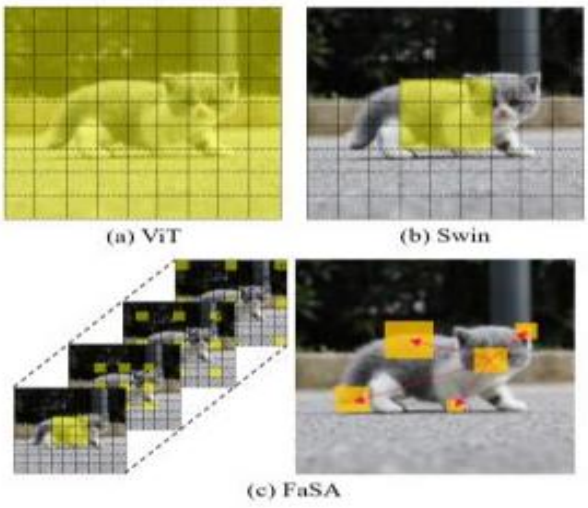

Fig 3.6 Bundle Adjustment

The biggest difference between these formulas and full bundle adjustment is that our feature location measurements xij now depend not only on the point (track) index i but also on the camera pose index j,

$$\begin{aligned}\mathbf{r}_{ij} &= \tilde{\mathbf{x}}_{ij} - \hat{\mathbf{x}}_{ij}, \\ s_{ij}^2 &= \mathbf{r}_{ij}^T \mathbf{\Sigma}_{ij}^{-1} \mathbf{r}_{ij}, \\ e_{ij} &= \hat{\rho}(s_{ij}^2),\end{aligned}$$

$$\begin{aligned}\frac{\partial e_{ij}}{\partial s_{ij}^2} &= \hat{\rho}'(s_{ij}^2), \\ \frac{\partial s_{ij}^2}{\partial \tilde{\mathbf{x}}_{ij}} &= \mathbf{\Sigma}_{ij}^{-1} \mathbf{r}_{ij}.\end{aligned}$$

While most of the boxes (transforms) have previously been explained , the leftmost box has not. This box performs a robust comparison of the predicted and measured 2D locations xˆij and ˜xij after re-scaling by the measurement noise covariance Σij . In more detail, this operation can be written as

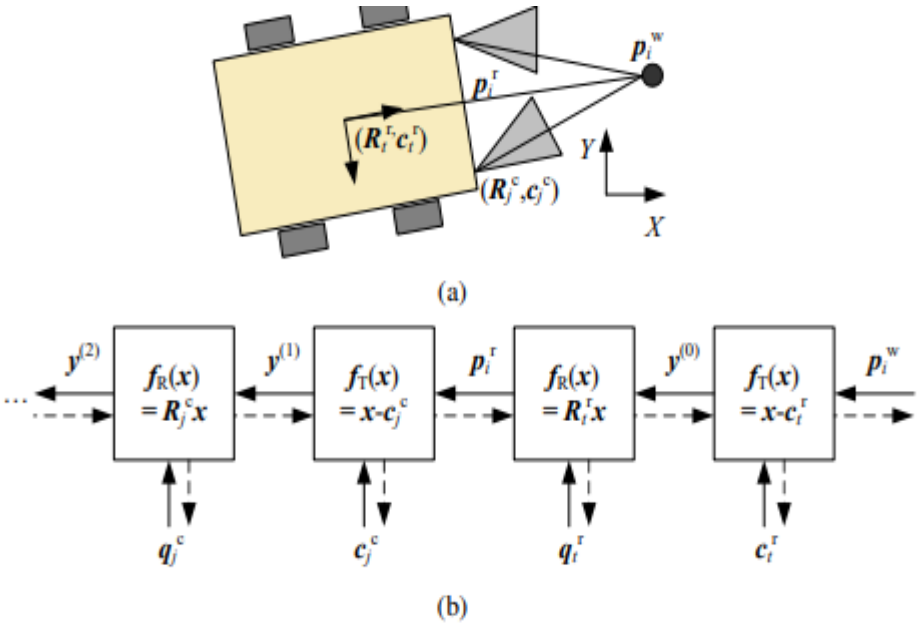

Fig 3.7 2D Bundle Adjustment

The advantage of the chained representation introduced above is that it not only makes the computations of the partial derivatives and Jacobians simpler but it can also be adapted to any camera configuration. Consider for example a pair of cameras mounted on a robot that is moving around in the world. By replacing the rightmost two transformations in with the transformations, we can simultaneously recover the position of the robot at each time and the calibration of each camera with respect to the rig, in addition to the 3D structure of the world.

3.7.1 Exploiting sparsity

Large bundle adjustment problems, such as those involving reconstructing 3D scenes from thousands of internet photographs can require solving non-linear least squares problems with millions of measurements (feature matches) and tens of thousands of unknown parameters (3D point positions and camera poses). Unless some care is taken, these kinds of problem can become intractable, because the (direct) solution of dense least squares problems is cubic in the number of unknowns.

Fortunately, structure from motion is a bipartite problem in structure and motion. Each feature point xij in a given image depends on one 3D point position pi and one 3D camera pose (Rj , cj). This is illustrated where each circle (1–9) indicates a 3D point, each square (A–D) indicates a camera, and lines (edges) indicate which points are visible in which cameras (2D features). If the values for all the points are known or fixed, the equations for all the cameras become independent, and vice versa.

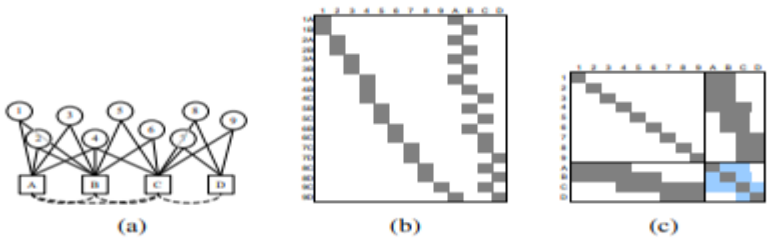

Fig 3.8 3D Bundle Adjustment

If we order the structure variables before the motion variables in the Hessian matrix A (and hence also the right-hand side vector b), we obtain a structure for the Hessian. When such a system is solved using sparse Cholesky factorization, the fill-in occurs in the smaller motion explore the use of iterative (conjugate gradient) techniques for the solution of bundle adjustment problems.

In more detail, the reduced motion Hessian is computed using the Schur complement

$$\mathbf{A}'_{\mathrm{CC}} = \mathbf{A}_{\mathrm{CC}} - \mathbf{A}_{\mathrm{PC}}^{T}\mathbf{A}_{\mathrm{PP}}^{-1}\mathbf{A}_{\mathrm{pc}},$$

where APP is the point (structure) Hessian (the top left block), APC is the point- camera Hessian (the top right block), and ACC and A0 CC are the motion Hessians before and after the point variable elimination (the bottom right block).

Notice that A0 CC has a non-zero entry between two cameras if they see any 3D point in common.

3.7.2 Application: Match move

One of the neatest applications of structure from motion is to estimate the 3D motion of a video or film camera, along with the geometry of a 3D scene, in order to superimpose 3D graphics or computer-generated images (CGI) on the scene. In the visual effects industry, this is known as the match move problem (Roble 1999), as the

motion of the synthetic 3D camera used to render the graphics must be matched to that of the real-world camera. For very small motions, or motions involving pure camera rotations, one or two tracked points can suffice to compute the necessary visual motion.

For planar surfaces moving in 3D, four points are needed to compute the homography, which can then be used to insert planar overlays, e.g., to replace the contents of advertising billboards during sporting events. The general version of this problem requires the estimation of the full 3D camera pose along with the focal length (zoom) of the lens and potentially its radial distortion parameters

When the 3D structure of the scene is known ahead of time, pose estimation techniques such as view correlation or through-the-lens camera control (Gleicher and Witkin 1992) can be used, as described in For more complex scenes, it is usually preferable to recover the 3D structure simultaneously with the camera motion using structure-from-motion techniques.

The trick with using such techniques is that to prevent any visible jitter between the synthetic graphics and the actual scene, features must be tracked to very high accuracy and ample feature tracks must be available in the vicinity of the insertion location

3.8 CONSTRAINED STRUCTURE AND MOTION

The most general algorithms for structure from motion make no prior assumptions about the objects or scenes that they are reconstructing. In many cases, however, the scene contains higher-level geometric primitives, such as lines and planes. These can provide information complementary to interest points and also serve as useful building blocks for 3D modeling and visualization.

Furthermore, these primitives are often arranged in particular relationships, i.e., many lines and planes are either parallel or orthogonal to each other. Sometimes, instead of exploiting regularity in the scene structure, it is possible to take advantage of a constrained motion model.

For example, if the object of interest is rotating on a turntable, i.e., around a fixed but unknown axis, specialized techniques can be used to recover this motion. In other situations, the camera itself may be moving in a fixed arc around some center of rotation Specialized capture setups, such as mobile stereo camera rigs or moving vehicles equipped with multiple fixed cameras, can also take advantage of the knowledge that individual cameras are (mostly) fixed with respect to the capture rig, as shown in Figu

Sometimes, instead of exploiting regularity in the scene structure, it is possible to take advantage of a constrained motion model. For example, if the object of interest is rotating on a turntable), i.e., around a fixed but unknown axis, specialized techniques can be used to recover this motion. In other situations, the camera itself may be moving in a fixed arc around some center of rotation (Shum and He 1999). Specialized capture

setups, such as mobile stereo camera rigs or moving vehicles equipped with multiple fixed cameras, can also take advantage of the knowledge that individual cameras are (mostly) fixed with respect to the capture.

3.8.1 Line-based techniques

It is well known that pairwise epipolar geometry cannot be recovered from line matches alone, even if the cameras are calibrated. To see this, think of projecting the set of lines in each image into a set of 3D planes in space.

You can move the two cameras around into any configuration you like and still obtain a valid reconstruction for 3D lines. Line-based techniques. When lines are visible in three or more views, the trifocal tensor can be used to transfer lines from one pair of images to another.

Author describe a widely used technique for matching 2D lines based on the average of 15 × 15 pixel correlation scores evaluated at all pixels along their common line segment intersection.35 In their system, the epipolar geometry is assumed to be known, e.g., computed from point matches. For wide baselines, all possible homographies corresponding to planes passing through the 3D line are used to warp pixels and the maximum correlation score is used.

For triplets of images, the trifocal tensor is used to verify that the lines are in geometric correspondence before evaluating the correlations between line segments. The results of using their system.

Instead of reconstructing 3D lines, use RANSAC to group lines into likely coplanar subsets. Four lines are chosen at random to compute a homography, which is then verified for these and other plausible line segment matches by evaluating color histogram-based correlation scores.

The 2D intersection points of lines belonging to the same plane are then used as virtual measurements to estimate the epipolar geometry, which is more accurate than using the homographies directly.

It describe a 3D modeling system that constructs calibrated panoramas from multiple images and then has the user draw vertical and horizontal lines in the image to demarcate the boundaries of planar regions.

The lines are used to establish an absolute rotation for each panorama and are then used (along with the inferred vertices and planes) to build a 3D structure, which can be recovered up to scale from one or more images

3.8.2 Plane-based techniques

In scenes that are rich in planar structures, e.g., in architecture, it is possible to directly estimate homographies between different planes, using either feature-based or intensity-based methods. In principle, this information can be used to simultaneously infer the camera poses and the plane equations, i.e., to compute plane-based structure

from motion.

It show how a fundamental matrix can be directly computed from two or more homographies using algebraic manipulations and least squares. Unfortunately, this approach often performs poorly, because the algebraic errors do not correspond to meaningful reprojection errors.

A better approach is to hallucinate virtual point correspondences within the areas from which each homography was computed and to feed them into a standard structure from motion algorithm.

An even better approach is to use full bundle adjustment with explicit plane equations, as well as additional constraints to force reconstructed co-planar features to lie exactly on their corresponding planes. (A principled way to do this is to establish a coordinate frame for each plane, e.g., at one of the feature points, and to use 2D in-plane parameterizations for the other points.)

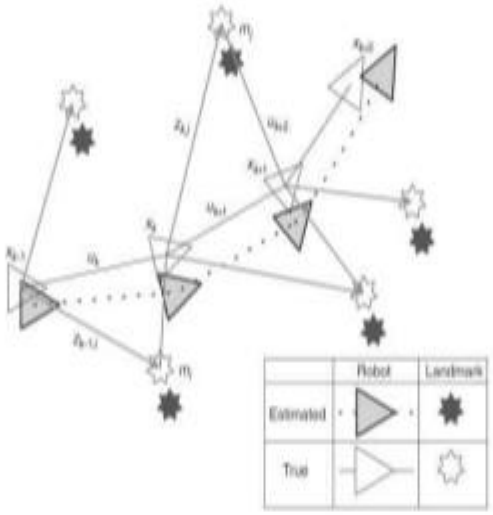

Fig 3.9 SLAM

3.8.3 Simultaneous localization and mapping (SLAM)

While the computer vision community has been studying structure from motion, i.e., the reconstruction of sparse 3D models from multiple images and videos, since the early 1980s), the mobile robotics community has in parallel been studying the automatic construction of 3D maps from moving robots.

36 In robotics, the problem was formulated as the simultaneous estimation of 3D robot and landmark poses and was known as probabilistic mapping and simultaneous localization and mapping (SLAM).

In the computer vision community, the problem was originally called visual odometry although that term is now usually reserved for shorter-range motion estimation that does not involve building a global map with loop closing.

Early versions of such algorithms used range-sensing techniques, such as ultrasound, laser range finders, or stereo matching, to estimate local 3D geometry, which could then be fused into a 3D model..

SLAM differs from bundle adjustment in two fundamental aspects. First, it allows for a variety of sensing devices, instead of just being restricted to tracked or matched feature points.

Second, it solves the localization problem online, i.e., with no or very little lag in providing the current sensor pose. This makes it the method of choice for both time-critical robotics applications such as autonomous navigation.

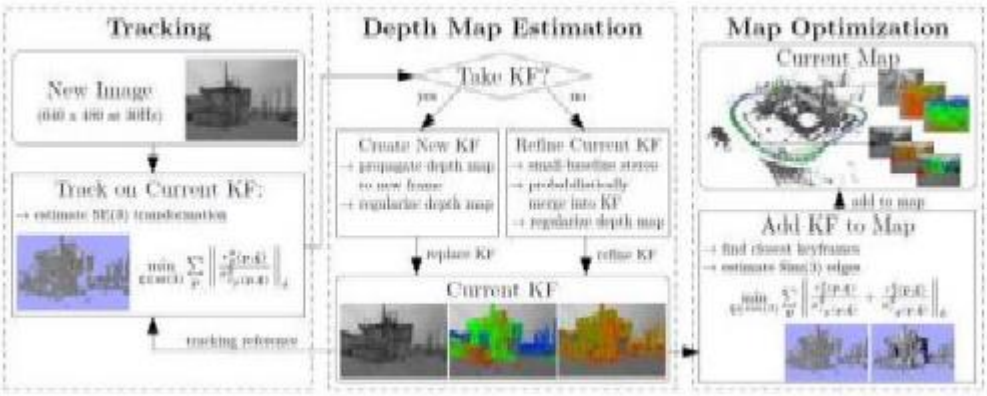

Fig 3.10 Two separate threads running at different rates

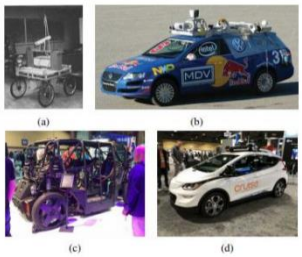

Fig 3.11 Autonomous Vehincles

As you can tell from this very brief overview, SLAM is an incredibly rich and rapidly evolving field of research, full of challenging robust optimization and real-time performance problems. A good source for finding a list of the most recent papers and algorithms is the KITTI Visual Odometry/SLAM Evaluation.

Application: Autonomous navigation Since the early days of artificial intelligence and robotics, computer vision has been used to enable manipulation for dextrous robots and navigation for autonomous robots (Janai, Guney ¨ et al. 2020; Kubota 2019). Some of the earliest vision-based navigation systems include the Stanford Cart and CMU Rover, the Terregator and the CMU Nablab.

Originally could only advance 4m every 10 sec (< 1 mph), and which was also the first system to use a neural network for driving .The early algorithms and technologies advanced rapidly, with the VaMoRs system of operating a 25Hz Kalman filter loop and driving with good lane markings at 100 km/h.

By the mid 2000s, when DARPA introduced their Grand Challenge and Urban Challenge, vehicles equipped with both range-finding lidar cameras and stereo cameras were able to traverse rough outdoor terrain and navigate city streets at regular human driving speeds. Systems led to the formation of industrial research projects at companies such as Google and Tesla, as well numerous startups, many of which exhibit their vehicles at computer vision conferences

3.9 TRANSLATIONAL ALIGNMENT

$$E_{\mathrm{SSD}}(\mathbf{u}) = \sum_i [I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i)]^2 = \sum_i e_i^2,$$

The simplest way to establish an alignment between two images or image patches is to shift one image relative to the other. Given a template image I0(x) sampled at discrete pixel locations {xi = (xi , yi)}, we wish to find where it is located in image I1(x). A least squares solution to this problem is to find the minimum of the sum of squared differences (SSD) function

where u = (u, v) is the displacement and ei = I1(xi + u) − I0(xi) is called the residual error (or the displaced frame difference in the video coding literature).1 (We ignore for the moment the possibility that parts of I0 may lie outside the boundaries of I1 or be otherwise not visible.) The assumption that corresponding pixel values remain the same in the two images is often called the brightness constancy constraint.

In general, the displacement u can be fractional, so a suitable interpolation function must be applied to image I1(x). In practice, a bilinear interpolant is often used, but bicubic interpolation can yield slightly better results.

Color images can be processed by summing differences across all three-color channels, although it is also possible to first transform the images into a different color space or to only use the luminance (which is often done in video encoders). Robust error metrics. We can make the above error metric more robust to outliers by replacing the squared error terms with a robust function ρ(ei)

$$E_{\mathrm{SRD}}(\mathbf{u}) = \sum_i \rho(I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i)) = \sum_i \rho(e_i).$$

The robust norm ρ(e) is a function that grows less quickly than the quadratic penalty associated with least squares. One such function, sometimes used in motion estimation for video coding because of its speed, is the sum of absolute differences (SAD) metric3 or L1 norm, i.e., However, because this function is not differentiable at the origin, it is not well suited to gradient-descent approaches such as the ones presented.

Instead, a smoothly varying function that is quadratic for small values but grows more slowly away from the origin is often used

$$\rho_{\mathrm{GM}}(x) = \frac{x^2}{1 + x^2/a^2},$$

where a is a constant that can be thought of as an outlier threshold. An appropriate value for the threshold can itself be derived using robust statistics. e.g., by computing the median absolute deviation, MAD = medi |ei |, and multiplying it by to obtain a robust estimate of the standard deviation of the inlier noise process proposes a generalized robust loss function that can model various outlier distributions and thresholds, as discussed in more detail in, and also has a Bayesian method for estimating the loss function parameters.

Spatially varying weights. The error metrics above ignore that fact that for a given alignment, some of the pixels being compared may lie outside the original image boundaries. Furthermore, we may want to partially or completely downweight the contributions of certain pixels.

For example, we may want to selectively “erase” some parts of an image from consideration when stitching a mosaic where unwanted foreground objects have been cut out. For applications such as background stabilization, we may want to downweight the middle part of the image, which often contains independently moving objects being tracked by the camera

All of these tasks can be accomplished by associating a spatially varying per-pixel weight with each of the two images being matched. The error metric then becomes the weighted (or windowed) SSD function

$$E_{\mathrm{WSSD}}(\mathbf{u}) = \sum_i w_0(\mathbf{x}_i) w_1(\mathbf{x}_i + \mathbf{u}) [I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i)]^2,$$

3.9.1 Hierarchical motion estimation

The simplest solution is to do a full search over some range of shifts, using either integer or sub-pixel steps. This is often the approach used for block matching in motion compensated video compression, where a range of possible motions (say, ±16 pixels) is explored. To accelerate this search process, hierarchical motion estimation is often used: an image pyramid is constructed and a search over a smaller number of discrete pixels (corresponding to the same range of motion) is first performed at coarser levels.

The motion estimate from one level of the pyramid is then used to initialize a smaller local search at the next finer level. Alternatively, several seeds (good solutions) from the coarse level can be used to initialize the fine-level search. While this is not guaranteed to produce the same result as a full search, it usually works almost as well and is much faster.

3.9.2 Fourier-based alignment

When the search range corresponds to a significant fraction of the larger image (as is the case in image stitching, the hierarchical approach may not work that well, as it is often not possible to coarsen the representation too much before significant features are blurred away. In this case, a Fourier-based approach may be preferable.

3.9.3 Windowed correlation.

Unfortunately, the Fourier convolution theorem only applies when the summation over xi is performed over all the pixels in both images, using a circular shift of the image when accessing pixels outside the original boundaries. While this is acceptable for small shifts and comparably sized images, it makes no sense when the images overlap by a small amount or one image is a small subset of the other.

3.9.4 Phase correlation.

A variant of regular correlation that is sometimes used for motion estimation is phase correlation.Here, the spectrum of the two signals being matched is whitened by dividing each per- frequency product in the magnitudes of the Fourier transforms.

3.10 PARAMETRIC MOTION

Parametric motion estimation In this section, we consider motion estimation approaches which estimate the parameters of the motion models. As previously discussed, these models can be applied to a coherently moving region of support. An important special case is when a single region of support corresponding to the whole image is selected.

In this case, referred to as global motion estimation, the dominant motion is estimated. This dominant motion is resulting from camera motion, such as dolly, track, boom, pan, tilt and roll, which is a widely used cinematic technique in filmmaking and video production. Hereafter, we describe two classes of techniques for parametric motion estimation. We also discuss difficulties arising due to outliers, and related robust estimators.

Indirect parametric motion estimation A first class of approaches indirectly computes the motion parameters from a dense motion field rather than from the image pixels. More specifically, a dense motion field is first estimated, and then the parametric motion model is fitted on the obtained motion vectors. A Least Mean Square (LMS) technique is commonly used for this model fitting. More specifically, the motion parameters are derived from the expressions

3.11 SPLINE-BASED MOTION

3.11.1 Concept

Motion fields are represented not per-pixel, but via a spline—typically a 2D grid of control points whose influence is defined by basis functions (like B-splines or thin-plate splines). This reduces complexity and imposes smoothness

3.11.2 Use cases:

Smooth interpolation of camera or object paths in graphics/animation

Medical image registration and video motion modeling with cubic/B-spline intensity representations .

3.11.3 Strengths & limitations:

- **Pros**: Fewer parameters, local control, smoother, and less noise.
- **Cons**: May over-smooth edges, requires careful tuning of knot/control point placement

3.12 OPTICAL FLOW

3.12.1 Definition:

Estimates a dense pixel-wise motion field that preserves brightness constancy and spatial smoothness over time.

3.12.2 Key algorithms:

- **Local**: Lucas-Kanade (small-window least squares).
- **Global**: Horn–Schunck, Farnebäck.
- **Spline-enhanced**: Embedding spline models improves accuracy and convergence in dense optical flow
- **Challenges**: Occlusions, brightness changes, large displacements, motion discontinuities, and blur.

3.13 LAYERED MOTION

3.13.1 Core idea:

Scene is decomposed into distinct layers—each with coherent motion—separate background and foreground movements

3.13.2 Why Layers?

They handle occlusions and motion boundaries naturally. Enables warping/blurring/deblurring separately per layer. Incorporates temporal and semantic info, boosting robustness

3.13.3 Techniques:

- Traditional: EM algorithms (e.g., Black & Jepson 1992)
- Modern: Graph-cuts, CRFs to segment layers and enforce temporal consistency
- Deep learning: Soft-mask layers within networks (e.g. FlowNet variants)

PART A

1. What is bundle adjustment?
2. Define reprojection error.
3. What is geometric intrinsic calibration?
4. Briefly explain triangulation in structure from motion.
5. What does the essential matrix encode?
6. What is factorization in SfM?
7. Define optical flow.
8. What is layered motion?
9. Describe translational alignment.
10. What is pose estimation?

PART B

1. Compare and contrast the factorization method (e.g., Tomasi–Kanade) and bundle adjustment for SfM
2. Describe the triangulation process using a linear SVD-based method:
3. Explain the formulation and optimization in bundle adjustment:
4. Explain how geometric constraints (e.g., parallelism, coplanarity) can be integrated into SfM:
5. Explain translational alignment methods for image registration or motion estimation:
6. Compare parametric (e.g., affine, homography) motion models with spline-based motion:
7. Explain the optical flow formulation and two classical computation methods:
8. Describe the concept of layered motion models in image sequences:
9. Explain the interplay between intrinsic calibration and pose estimation:
10. Explain the full pipeline of two-frame SfM

PART C

1. What is the goal of feature-based alignment?
2. Define the task and its applications
3. Explain the pipeline: feature detection → description → matching → geometric model estimation
4. How does RANSAC improve robustness in feature alignment?
5. What are intrinsic camera parameters and how are they estimated?
6. Explain triangulation: solving for 3D point from multiple views via linear least squares or SVD, then refine with bundle adjustment
7. Describe how a checkerboard pattern is used: detect corners, use DLT, then refine via non-linear minimization
8. What are intrinsic camera parameters and how are they estimated?
9. Compare structure-from-motion vs. optical flow: what each uses and reconstructs.

10. What challenges arise in pose estimation and motion analysis (e.g. noise, occlusions)?
11. Why use spline models for camera trajectories/video sequences? Discuss temporal smoothness.
12. Parametric motion estimation: how to estimate motion parameters (translation, rotation) via reprojection error minimization.

.

UNIT – IV

3D RECONSTRUCTION

Shape from X - Active range finding - Surface representations - Point-based representations- Volumetric representations - Model-based reconstruction - Recovering texture maps and albedosos

4.1 SHAPE FORM X

In addition to binocular disparity, shading, texture, and focus all play a role in how we perceive shape. The study of how shape can be inferred from such cues is sometimes called shape from X, because the individual instances are called shape from shading, shape from texture, and shape from focus.

4.1.1 Shape from shading and photometric stereo

When you look at images of smooth shaded objects, such as the ones, you can clearly see the shape of the object from just the shading variation.

The problem of recovering the shape of a surface from this intensity variation is known as shape from shading and is one of the classic problems in computer vision.

Most shape from shading algorithms assume that the surface under consideration is of a uniform albedo and reflectance, and that the light source directions are either known or can be calibrated by the use of a reference object. Under the assumptions of distant light sources and observer, the variation in intensity (irradiance equation) becomes purely a function of the local surface orientation.

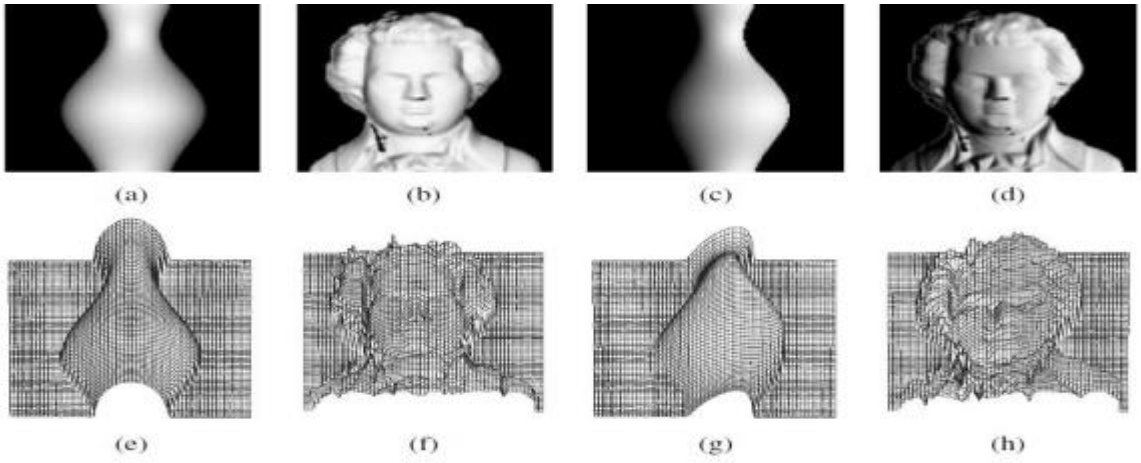

Fig 4.1 Shape From Shading And Photometric Stereo

Instead of first recovering the orientation fields (p,q) and integrating them to obtain surface, it is also possible to directly minimize the discrepancy in the image formation equation (13.1) while finding the optimal depth map z(x,y).

Unfortunately, shape from shading is susceptible to local minima in the search space and, like other variational problems that involve the simultaneous estimation of many variables, can also suffer from slow convergence. Using multi-resolution techniques can help accelerate the convergence, while using more sophisticated optimization

techniques can help avoid local minima.

Photometric stereo another way to make shape from shading more reliable is to use multiple light sources that can be selectively turned on and off. This technique is called photometric stereo, as the light sources play a role analogous to the cameras located at different locations in traditional stereo.For each light source, we have a different reflectance map, R1(p,q), R2(p,q), etc. Given the corresponding intensities I1, I2, etc. at a pixel, we can in principle recover both an unknown albedo _ and a surface orientation estimate (p,q).

For diffuse surfaces (13.2), if we parameterize the local orientation by ^n, we get (for non-shadowed pixels) a set of linear equations of the form from which we can recover _^n using linear least squares. These equations are well conditioned as long as the (three or more) vectors vk are linearly independent, i.e., they are not along the same azimuth (direction away from the viewer).

Once the surface normals or gradients have been recovered at each pixel, they can beintegrated into a depth map using a variant of regularized surface fitting .The combination of multi-view stereo for coarse shape and photometric stereo for fine detail continues to be an active area of research describe such a system that can produce very high-quality scans, although it requires a sophisticated laboratory setup.

It is also possible to apply photometric stereo to outdoor web camera sequences , using the trajectory of the Sun as a variable direction illuminator. When surfaces are specular, more than three light directions may be required. In fact, the irradiance equation given in (13.1) not only requires that the light sources and camera be distant from the surface, it also neglects inter-reflections, which can be a significant source of the shading observed on object surfaces, e.g., the darkening seen inside concave structures such as grooves and crevasses.

4.1.2 Shape From Texture

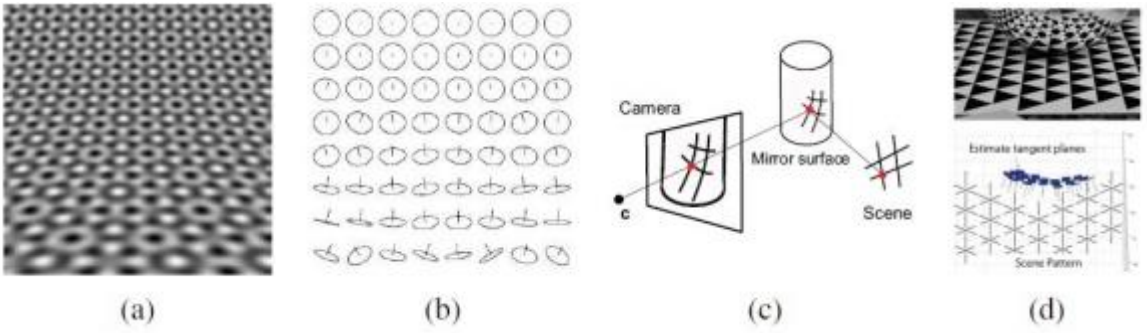

Fig 4.2 Shape From Texture

The variation in foreshortening observed in regular textures can also provide useful information about local surface orientation. An example of such a pattern, along with the estimated local surface orientations. Shape from texture algorithms require a number of processing steps, including the extraction of repeated patterns or the

measurement of local frequencies to compute local affine deformations, and a subsequent stage to infer local surface orientation.

When the original pattern is regular, it is possible to fit a regular but slightly deformed grid to the image and use this grid for a variety of image replacement or analysis tasks.

The deformations induced in a regular pattern when it is viewed in the reflection of a curved mirror. It can be used to recover the shape of the surface. It is also possible to infer local shape information from specular flow, i.e., the motion of specularities when viewed from a moving camera.

Shape from focus a number of techniques have been developed to estimate depth from the amount of defocus (depth from defocus). To make such a technique practical, a number of issues need to be addressed:

The amount of blur increase in both directions as you move away from the focus plane. Therefore, it is necessary to use two or more images captured with different focus distance settings or to translate the object in depth and look for the point of maximum sharpness.

The magnification of the object can vary as the focus distance is changed or the object is moved. This can be modeled either explicitly (making correspondence more difficult) or using telecentric optics, which approximate an orthographic camera and require an aperture in front of the lens.

The amount of defocus must be reliably estimated. A simple approach is to average the squared gradient in a region, but this suffers from several problems, including the image magnification problem mentioned above. A better solution is to use carefully designed rational filters.

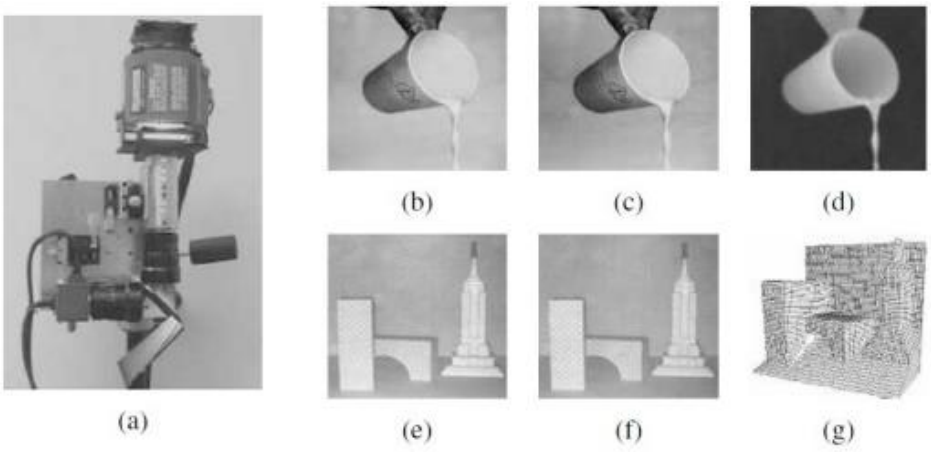

Fig 4.3 Real-Time Depth From Defocus

4.2 ACTIVE RANGE FINDING

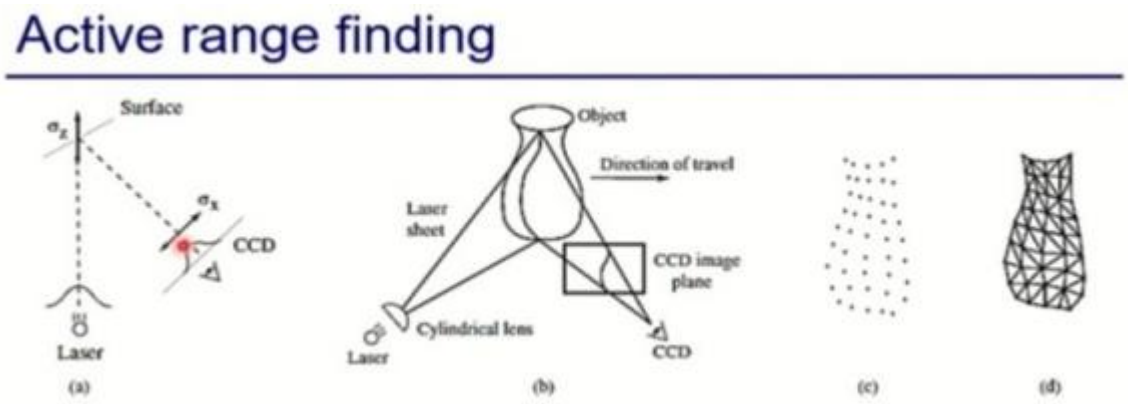

Fig 4.4 Range Data Scanning

Active Range Finding is a technique used to directly measure the distance from a sensor (camera or scanner) to points on the surface of an object or scene. It is "active" because it involves emitting some form of signal (e.g., light, laser, infrared) and measuring the time or characteristics of its return to compute depth.

4.2.1 Role in 3D Reconstruction

3D Reconstruction involves capturing the shape and appearance of a physical object or environment to create a digital 3D model.

Active range finding provides depth information by giving precise distance measurements to the object's surfaces, which can be combined with color or intensity data to reconstruct 3D geometry.

4.2.2 Common Active Range Finding Methods

4.2.2.1 Time-of-Flight (ToF) Cameras

- Emit a pulse of light (usually infrared).
- Measure the time it takes for the light to reflect off surfaces and return.
- Depth = speed of light × time delay.
- Example: Microsoft Kinect v2.

4.2.2.2 Structured Light Scanning

- Project a known light pattern (e.g., stripes, grids) onto the object.
- Capture the distortion of the pattern with a camera.
- Triangulate to find depth.
- Example: Early Microsoft Kinect, Artec scanners.

4.2.2.3 Laser Scanners (LIDAR)

- Sweep a laser beam across the scene.
- Measure either the time delay or phase shift of the reflected laser.
- Often used for large-scale outdoor environments.

4.2.2.4 Advantages of Active Range Finding in 3D Reconstruction

- Direct Depth Measurement: More robust to texture-less surfaces and lighting conditions
- High Accuracy: Can produce millimeter to sub-millimeter accuracy depending on technology.
- Real-Time Capability: Some ToF cameras provide real-time depth maps.

4.2.3 Challenges

- Noise and Artifacts: Reflective, transparent, or very dark surfaces can distort depth data.
- Limited Range: Effective range depends on the sensor; ToF cameras are typically limited indoors.
- Cost and Complexity: High-end laser scanners can be expensive.

4.2.4 Applications

- Robotics (navigation and manipulation)
- Augmented Reality (AR) and Virtual Reality (VR)
- Industrial inspection
- Medical imaging

4.3 SURFACE REPRESENTATIONS

Surface representations in computer vision refer to the ways in which the geometry or shape of surfaces in a three-dimensional (3D) scene is represented. These representations are crucial for tasks such as 3D reconstruction, computer graphics, and virtual reality. Different methods exist for representing surfaces, and the choice often depends on the application's requirements and the characteristics of the data.

4.3.1 Surface interpolation

One of the most common operations on surfaces is their reconstruction from a set of sparse data constraints, i.e., scattered data interpolation, When formulating such problems, surfaces may be parameterized as height fields f(x), as 3D parametric surfaces f (x), or as non-parametric models such as collections of triangles.

Two-dimensional function interpolation and approximation problem $\{d_i\} \longrightarrow f(x)$ could be cast as energy minimization problems using regularization . Such problems can also specify the locations of discontinuities in the surface as well as local orientation constraints

One approach to solving such problems is to discretize both the surface and the energy on a discrete grid or mesh using finite element analysis. Such problems can then be solved using sparse system solving techniques, such as multigrid or hierarchically preconditioned conjugate gradient.

The surface can also be represented using a hierarchical combination of multilevel B- splines. An alternative approach is to use radial basis (or kernel) functions Unfortunately, because the dense system solving is cubic in the number of data points, basis function approaches can only be used for small problems such as feature based image morphing .

When a three-dimensional parametric surface is being modeled, the vector-valued function f in encodes 3D coordinates (x, y, z) on the surface and the domain x = (s; t) encodes the surface parameterization. One example of such surfaces are symmetry seeking parametric models, which are elastically deformable versions of generalized cylinders. In these models, s is the parameter along the spine of the deformable tube and t is the parameter around the tube.

A variety of smoothness and radial symmetry forces are used to constrain the model while it is fitted to image-based silhouette curves. It is also possible to define non-parametric surface models, such as general triangulated meshes, and to equip such meshes (using finite element analysis) with both internal smoothness metrics and external data fitting metrics.

Both parametric and non-parametric surface models assume that the topology of the surface is known and fixed ahead of time. For more flexible surface modeling, we can either represent the surface as a collection of oriented points or use 3D implicit functions, which can also be combined with elastic 3D surface models. The field of surface reconstruction from unorganized point samples continues to advance rapidly, with more recent work addressing issues with data imperfections.

4.3.1.1 Surface interpolation Simplified:

Surface interpolation involves reconstructing surfaces from sparse data constraints. Surfaces can be parameterized as height fields, 3D parametric surfaces, or non-parametric models like triangle collections.

Interpolation problems are cast as energy minimization tasks, allowing for discontinuity and orientation constraint specification. Finite element analysis discretizes surfaces and energies onto grids or meshes for solution using techniques like multigrid or conjugate gradient.

Hierarchical B-splines or radial basis functions offer alternative approaches but are limited by computational complexity.

4.3.2 Surface simplification

Once a triangle mesh has been created from 3D data, it is often desirable to create a hierarchy of mesh models, for example, to control the displayed level of detail (LOD) in a computer graphics application.One approach to doing this is to approximate a given mesh with one that has subdivision connectivity, over which a set of triangular

wavelet coefficients can then be computed.

A more continuous approach is to use sequential edge collapse operations to go from the original fine-resolution mesh to a coarse base-level mesh. The resulting progressive mesh (PM) representation can be used to render the 3D model at arbitrary levels of detail.

4.3.2.1 Surface simplification Simplified

Hierarchy of Mesh Models: After creating a triangle mesh from 3D data, it's often useful to establish a hierarchy of mesh models to control the level of detail (LOD) displayed in computer graphics applications.

Subdivision Connectivity: One method involves approximating the given mesh with one having subdivision connectivity. This allows computation of triangular wavelet coefficients over the mesh, facilitating LOD control.

Sequential Edge Collapse: Another approach employs sequential edge collapse operations to transition from the original fine-resolution mesh to a coarse base-level mesh. This results in a progressive mesh (PM) representation.

Progressive Mesh (PM): The PM representation enables rendering the 3D model at arbitrary levels of detail. It provides flexibility in displaying different levels of detail based on requirements, enhancing rendering efficiency and optimizing performance in graphics applications.

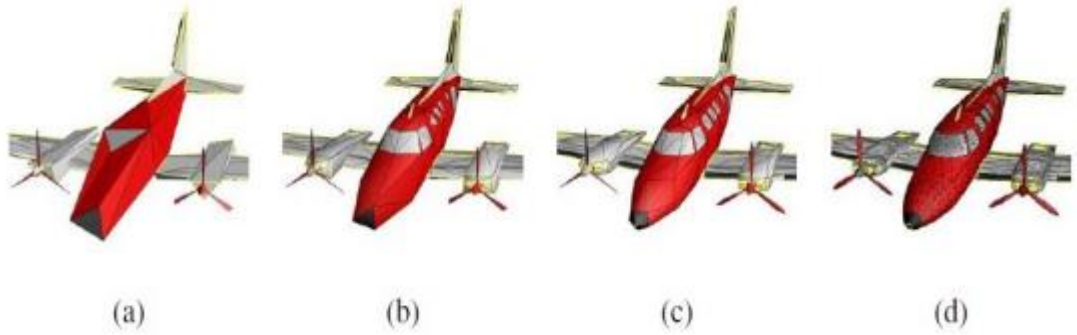

Fig 4.5 Surface Representations

4.3.3 Geometry Images

While multi-resolution surface representations support level of detail operations, they still consist of an irregular collection of triangles, which makes them more difficult to compress and store in a cache-efficient manner.

To make the triangulation completely regular (uniform and gridded), describe how to create geometry images by cutting surface meshes along well chosen line shows the resulting (x; y; z) values of the surface mesh mapped over the unit square, while Figu shows the associated (nx; ny; nz) normal map, i.e., the surface normals associated with each mesh vertex, which can be used to compensate for loss in visual fidelity if the original geometry image is heavily compressed.

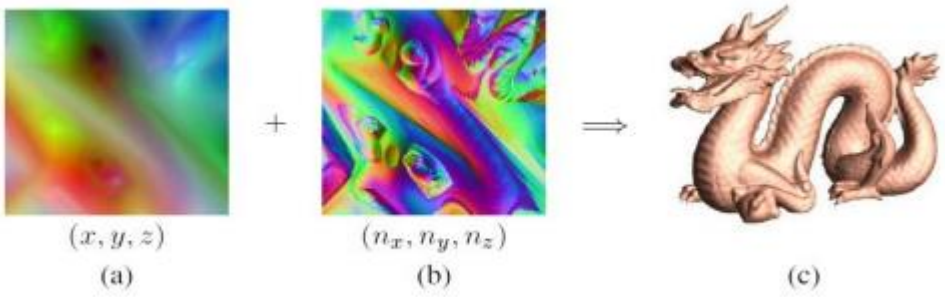

Fig 4.6 Geometry Images

4.3.3.1 Geometry Image Simplified

Creating Geometry Images: To transform multi-resolution surface representations into a completely regular and gridded form, geometry images are constructed. This process involves cutting surface meshes along well-chosen lines and then "flattening" the resulting representation into a square.

Mapping Surface Mesh onto Unit Square: , the resulting (x, y, z) values of the surface mesh are mapped over the unit square.

Normal Map Generation: Depicts the associated (nx, ny, nz) normal map. This map represents the surface normals associated with each mesh vertex. These normals can compensate for any loss in visual fidelity that may occur due to heavy compression of the original geometry image.

4.4 POINT-BASED REPRESENTATIONS

As we mentioned previously, triangle-based surface models assume that the topology (and often the rough shape) of the 3D model is known ahead of time. While it is possible to re-mesh a model as it is being deformed or fitted, a simpler solution is to dispense with an explicit triangle mesh altogether and to have triangle vertices behave as oriented points, or particles, or surface elements (surfels).

To endow the resulting particle system with internal smoothness constraints, pairwise interaction potentials can be defined that approximate the equivalent elastic bending energies that would be obtained using local finite-element analysis.Instead of defining the finite element neighborhood for each particle (vertex) ahead of time, a soft influence function is used to couple nearby particles.

Another alternative is to first convert the point cloud into an implicit signed distance or inside–outside function, using either minimum signed distances to the oriented points or by interpolating a characteristic (inside–outside) function using radial basis functions.

Even greater precision over the implicit function fitting, including the ability to handle irregular point densities, can be obtained by computing a moving least squares (MLS) estimate of the signed distance function.

Point-based representations in computer vision and computer graphics refer to methods that represent surfaces or objects using a set of individual points in three-dimensional (3D) space. Instead of explicitly defining the connectivity between points as in polygonal meshes, point-based representations focus on the spatial distribution of points to describe the surface geometry.

4.4.1 Some Common Point-based Representations:

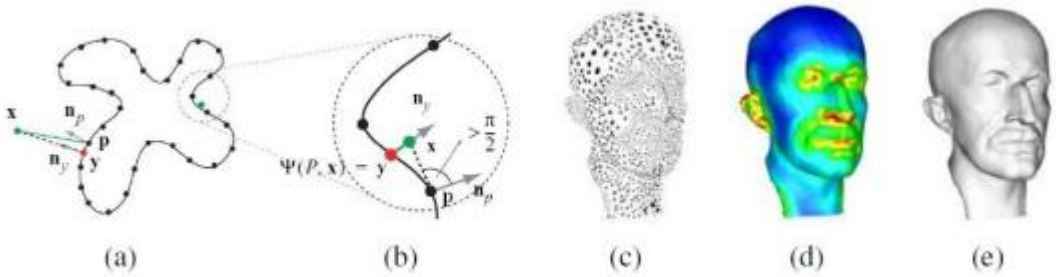

Fig 4.7 Point-Based Representation Least Squares

4.4.1.1 Point Clouds:

- Description: A collection of 3D points in space, each representing a sample on the surface of an object or a scene.
- Application: Point clouds are generated by 3D scanners, LiDAR, depth
- sensors, or photogrammetry. They find applications in robotics, autonomous vehicles, environmental mapping, and 3D modeling.

4.4.1.2 Dense Point Clouds:

- Description: Similar to point clouds but with a high density of points, providing more detailed surface information.
- Application: Used in applications requiring detailed 3D reconstructions, such as cultural heritage preservation, archaeological studies, and industrial inspections.

4.4.1.3 Sparse Point Sets:

- Description: Representations where only a subset of points is used to describe the surface, resulting in a sparser dataset compared to a dense point cloud.
- Application: Sparse point sets are useful in scenarios where computational efficiency is crucial, such as real-time applications and large-scale environments.

4.4.1.4 Point Splats:

- Description: Represent each point as a disc or a splat in 3D space. The size and orientation of the splats can convey additional information.
- Application: Commonly used in point-based rendering and visualization to represent dense point clouds efficiently.

4.4.1.5 Point Features:

- Description: Represent surfaces using distinctive points or key points, each associated with local features such as normals, color, or texture information.
- Application: Widely used in feature based registration , object recognition and 3D reconstruction.

4.4.1.6 Point Set Surfaces:

- Description: Represent surfaces as a set of unorganized points without connectivity information. Surface properties can be interpolated from neighboring points.
- Application: Used in surface reconstruction from point clouds and point- based rendering.

4.4.1.7 Radial Basis Function (RBF) Representations:

- Description: Use radial basis functions to interpolate surface properties between points. These functions define a smooth surface that passes through the given points.
- Application: Commonly used in shape modeling, surface reconstruction, and computer-aided design.

4.5 VOLUMETRIC REPRESENTATIONS

Volumetric representations in computer vision and computer graphics are methods used to describe and model three-dimensional (3D) space in a volumetric manner. Unlike surface representations, which focus on defining the surface geometry explicitly, volumetric representations capture information about the entire volume, including the interior of objects

A third alternative for modeling 3D surfaces is to construct 3D volumetric inside–outside functions, where we looked at voxel coloring , space carving , and level set techniques for stereo matching,where we discussed using binary silhouette images to reconstruct volumes.

4.5.1 Implicit Surfaces And Level Sets

While polyhedral and voxel-based representations can represent three-dimensional shapes to an arbitrary precision, they lack some of the intrinsic smoothness properties available with continuous implicit surfaces, which use an indicator function (or characteristic function) F(x; y; z) to indicate which 3D points are inside F(x; y; z) < 0 or outside F(x; y; z) > 0 the object.

An early example of using implicit functions to model 3D objects in computer vision were superquadrics, To model a wider variety of shapes, superquadrics are usually combined with either rigid or non-rigid deformations. Superquadric models can either be fitted to range data or used directly for stereo matching.

A different kind of implicit shape model can be constructed by defining a signed distance function over a regular three-dimensional grid, optionally using an octree spline to represent this function more coarsely away from its surface (zero-set) .

We have already seen examples of signed distance functions being used to represent distance transforms, level sets for 2D contour fitting and tracking , volumetric stereo , range data merging , and point-based modelling . The advantage of representing such functions directly on a grid is that it is quick and easy to look up distance function values for any (x, y,z) location and also easy to extract the isosurface using the marching cubes algorithm.

The function itself is represented using a quadratic tensorproduct B-spline over an octree, which provides a compact representation with larger cells away from the surface or in regions of lower point density, and also admits the efficient solution of the related Poisson equations (4.24–4.27).

It is also possible to replace the quadratic penalties used in the Poisson equations with L1 (total variation) constraints and still obtain a convex optimization problem, which can be solved using either continuous or discrete graph cut techniques.

Signed distance functions also play an integral role in level-set evolution equations, where the values of distance transforms on the mesh are updated as the surface evolves to fit multi-view stereo photo consistency measures.

As with many other areas of computer vision, deep neural networks have started being applied to the construction and modeling of volumetric object representations. Some neural networks construct 3D surface or volumetric occupancy grid models from single images , although more recent experiments suggest that these networks may just be recognizing the general object category and doing a small amount of fitting .

All of these networks use latent codes to represent individual instances from a generic class (e.g., car or chair) from the ShapeNet dataset (Chang, Funkhouser et al. 2015), although they use the codes in a different part of the network (either in the input or through conditional batch normalization). This allows them to reconstruct 3D models from just a single image.

They are trained specifically on clothed humans and can hallucinate full 3D models from just a single color image. Neural Radiance Fields (NeRF) extend this to also use pixel ray directions as inputs and also output continuous valued opacities and radiance values, enabling ray-traced rendering of shiny 3D models constructed from multiple input images. This representation is related to Lumigraphs and surface LightFields.

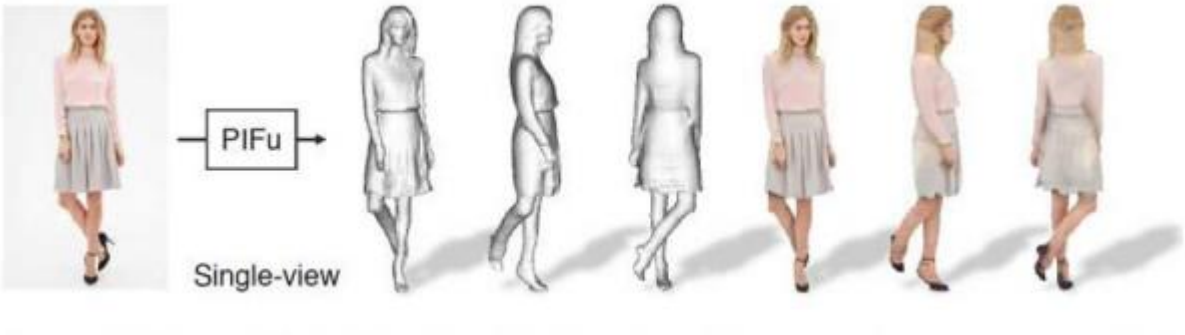

Fig 4.8 Pixels-Alinged Implicit Function

4.5.2 Voxel Grids:

- Description: A regular grid of small volume elements, called voxels, where each voxel represents a small unit of 3D space.
- Application: Used in medical imaging , Computer -aided design (CAD) ,computational fluid dynamics, and robotics. Voxel grids are effective for representing both the exterior and interior of objects.

4.5.3 Octrees:

- Description: A hierarchical data structure that recursively divides 3D space into octants. Each leaf node in the octree contains information about the occupied or unoccupied status of the corresponding volume.
- Application: Octrees are employed for efficient storage and representation of volumetric data, particularly in real-time rendering, collision detection, and adaptive resolution.

4.5.4 Signed Distance Fields (SDF):

- Description: Represent the distance from each point in space to the nearest surface of an object, with positive values inside the object and negative values outside.

- Application: Used in shape modeling, surface reconstruction, and physics-based simulations. SDFs provide a compact representation of geometry and are often used in conjunction with implicit surfaces.

4.5.5 3D Texture Maps:

- Description: Extend the concept of 2D texture mapping to 3D space, associating color or other properties with voxels in a volumetric grid.
- Application: Employed in computer graphics, simulations, and visualization to represent complex volumetric details such as smoke, clouds, or other phenomena.

4.5.6 Point Clouds with Occupancy Information:

- Description: Combine the idea of point clouds with additional information about the occupancy of each point in space.
- Application: Useful in scenarios where capturing both the surface and interior details of objects is necessary, such as in robotics and 3D reconstruction.

4.5.7 Tensor Fields:

- Description: Represent the local structure of a volumetric region using tensors. Tensor fields capture directional information, making them suitable for anisotropic materials and shapes.
- Application: Commonly used in materials science, biomechanics, and simulations where capturing anisotropic properties is important.

4.5.8 Shell Maps:

- Description: Represent the surfaces of objects as a collection of shells or layers, each encapsulating the object's geometry .
- Application: Used in computer graphics and simulation to efficiently represent complex objects and enable dynamic level-of-detail rendering.

4.6 MODEL-BASED RECONSTRUCTION

When we know something ahead of time about the objects we are trying to model, We can construct more detailed and reliable 3D models using specialized techniques and representations.

For example, architecture is usually made up of large planar regions other parametric forms (such as surfaces of revolution), usually oriented perpendicular to gravity and to each other. Heads and faces can be represented using low-dimensional, nonrigid shape models, because the variability in shape and appearance of human faces, while extremely large, is still bounded. Human bodies or parts, such as hands, form highly articulated structures, which can be represented using kinematic chains of piecewise rigid skeletal elements linked by joints.

4.6.1 Architecture

Architectural modeling, especially from aerial photography, has been one of longest studied problems in both photo grammetry and computer vision. In the last two decades, the development of reliable image-based modeling techniques, as well as the prevalence of digital cameras and 3D computer games, has led to widespread deployment of such systems.

Fig. 4.9 Interactive Modeling Using Facade System

The work by Debevec, Taylor, and Malik (1996) was one of the earliest geometry and image-based modeling and rendering systems. Their Fac¸ade system combines an interactive image-guided geometric modeling tool with model-based (local plane plus parallax) stereo matching and view-dependent texture mapping. During the interactive photogrammetric modeling phase, the user selects block elements and aligns their edges with visible edges in the input images.

The system then automatically computes the dimensions and locations of blocks along with the camera positions using constrained optimization. This approach is intrinsically more reliable than general feature-based structure from motion, because it exploits the strong geometry available in the block primitives.

Once the rough geometry has been estimated, more detailed offset maps can computed for each planar face using a local plane sweep, call model-based stereo. Finally, during rendering, images from different viewpoints are warped and blended together as the camera moves around the scene, using a process called view-dependent texture mapping.

For interior modeling, instead of working with single pictures, it is more useful to work with panoramas, as you can see larger extents of walls and other structures. The lines are initially used to establish an absolute rotation for each panorama and are later used (along with the inferred vertices and planes) to optimize the 3D structure, which can be recovered up to scale from one or more images. Recent advances in deep networks now make it possible to both automatically infer the lines and their junctions high dynamic range panoramas can also be used for outdoor modeling,

4.6.2 Facial Modeling and Tracking

As you can See, it is then possible to fit morphable 3d models to single images and to Use such models for a variety of animation and visual effects. It is also possible to design stereo matching algorithms that optimize Directly for the head model parameters Or to use the output of real-time stereo with active illumination.

Once a 3d head model has been constructed, it can be used in a variety applications, Such as head tracking and face transfer, i.e., replacing one person's face With another in a video . Additional Applications include face beautification by warping face images toward a more attractive "standard" , face de-identification for privacy protection , and face swapping

Fig 4.10 Photorealistic Avatars

More recent applications of 3d head models include photorealistic avatars for video conferencing , 3d unwarping for better selfies, and single image portrait relighting.

4.6.3 Application: Facial Animation

Blanz And vetter (1999) describe a system where they first capture a set of 2 colored range scans of faces , which can be represented as a large collection of (X,Y,Z,R,G,B) samples (vertices).16 for 3d morphing to be meaningful, corresponding vertices in different people's scans must first be put into correspondence.

After computing a subspace representation, different directions in this space can associated with different characteristics such as gender, facial expressions, or facial features . 3D morphable models can be fitted to a single image using gradient descent on the error between the input image and the re-synthesized model image, after an initial manual placement of the model in an approximately correct pose, scale, and location.

The resulting texture-mapped 3d model can then be modified to produce a variety visual effects, including changing a person's weight or expression, or three- dimensional effects such as re-lighting or 3d video-based animation. Such models can also be used for video compression, e.g., by only transmitting a small number of facial expression and pose parameters to drive a synthetic avatar or to bring a still portrait image to life. The survey paper on 3d morphable face models by discusses additional research and

applications in this area.

3d facial animation is often matched to the performance of an actor, in what is know as performance-driven animation. Traditional performance driven animation systems use marker-based motion capture, while some newer systems use depth cameras or regular video to control the animation

4.6.4 Human Body Modeling and Tracking

The topics of tracking humans, modeling their shape and appearance, and recognizing their activities, are some of the most actively studied areas of computer vision. The HumanEva database of articulated human motions contains multi-view video sequences of human actions along with corresponding motion capture data, evaluation code, and a reference 3D tracker based on particle filtering.

We refer the reader to the previously mentioned surveys for other topics and more details. Background subtraction : One of the first steps in many human tracking systems is to model the background to extract the moving foreground objects (silhouettes) corresponding to people.Tracking such silhouettes over time supports the analysis of multiple people moving around a scene, including building shape and appearance models and detecting if they are carrying objects.

Initialization and detection : To track people in a fully automated manner, it is necessary to first detect (or re-acquire) their presence in individual video frames.Single-frame human detection and pose estimation algorithms can be used by themselves to perform tracking. They are often combined, however, with frame-to-frame tracking techniques to provide better reliability.

Tracking with flow: The tracking of people and their pose from frame to frame can be enhanced by computing optical flow or matching the appearance of their limbs from one frame to another. For example, the cardboard people models the appearance of each leg portion (upper and lower) as a moving rectangle, and uses optical flow to estimate their location in each subsequent frame. It is also possible to match the estimated motion field itself to some prototypes in order to identify the particular phase of a running motion or to match two low-resolution video portions to perform video replacement.

3D kinematic models : The effectiveness of human modeling and tracking can be greatly enhanced using a more accurate 3D model of a person’s shape and motion. Underlying such representations, which are ubiquitous in 3D computer animation in games and special effects, is a kinematic model or kinematic chain, which specifies the length of each limb in a skeleton as well as the 2D or 3D rotation angles between the limbs or segments Inferring the values of the joint angles from the locations of the visible surface points is called inverse kinematics (IK) and is widely studied in computer graphics.

Shows the kinematic model for a human hand used to track hand motion in a video. As you can see, the attachment points between the fingers and the thumb have two degrees of freedom, while the finger joints themselves have only one. Using this kind of model can greatly enhance the ability of an edge-based tracker to cope with rapid motion, ambiguities in 3D pose, and partial occlusions.One popular approach is to associate an ellipsoid or superquadric with each rigid limb in the kinematic model.

Probabilistic models : Because tracking can be such a difficult task, sophisticated probabilistic inference techniques are often used to estimate the likely states of the person being tracked. An example of a sophisticated spatio-temporal probabilistic graphical model called loose-limbed people, which models not only the geometric relationship between various limbs, but also their likely temporal dynamics. The conditional probabilities relating various limbs and time instances are learned from training data, and particle filtering is used to perform the final pose inference.

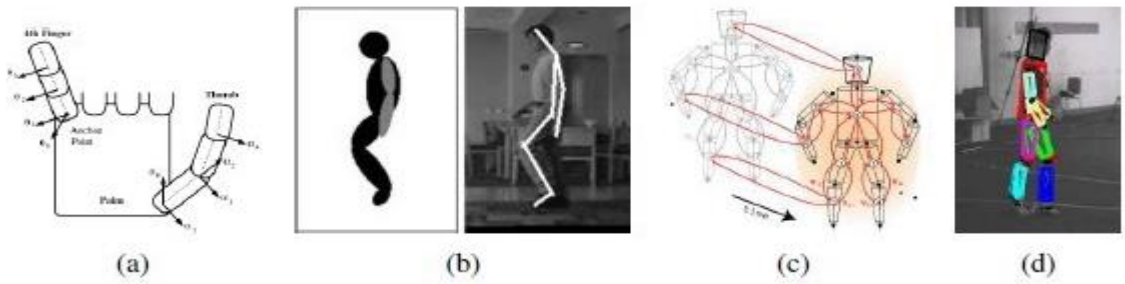

Fig 4.11 Tracking 3d Motion

Adaptive shape modeling : Adaptive shape modeling refers to the use of mathematical and computational techniques to represent and analyze the shapes of objects in images or videos. These techniques allow for the automatic detection, recognition, and tracking of objects by adapting to variations in shape, size, and appearance. The registered datasets are used to model the variation in shape as a function of personal characteristics and skeletal pose,

The resulting system can then be used for shape completion, i.e., the recovery of a full 3D mesh model from a small number of captured markers, by finding the best model parameters in both shape and pose space that fit the measured data. Because it is constructed completely from scans of people in close-fitting clothing and uses a parametric shape model, the SCAPE system cannot cope with people wearing loose fitting clothing.

While the preceding body fitting and pose estimation systems use multiple views to estimate body shape, fit a human shape and pose model to a single image of a person on a natural background. Manual initialization is used to estimate a rough pose (skeleton) and height model, and this is then used to segment the person’s outline using

the Grab Cut segmentation algorithm. The shape and pose estimate are then refined using a combination of silhouette edge cues and shading information. The resulting 3D model can be used to create novel animations.

Activity recognition : The final widely studied topic in human modeling is motion, activity, and action recognition. Examples of actions that are commonly recognized include walking and running, jumping, dancing, picking up objects, sitting down and standing up, and waving.

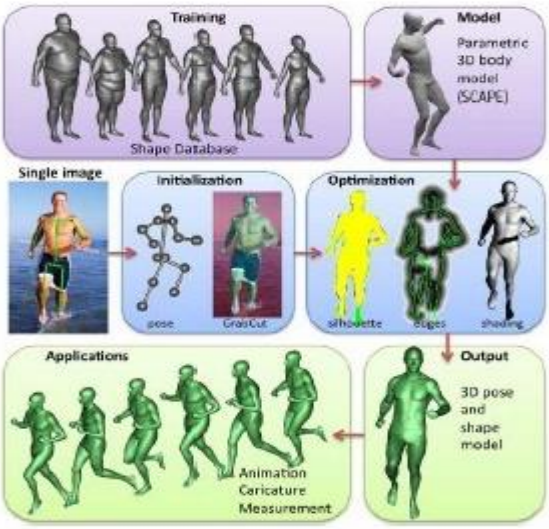

Fig 4.12 Estimating Human And Post

4.7 RECOVERING TEXTURE MAPS AND ALBEDOSOS

After a 3D model of an object or person has been acquired, the final step in model is usually to recover a texture map to describe the object's surface appearance. This first requires establishing a parameterization for the (u,v) texture coordinates as a function of 3D surface position.One simple way to do this is to associate a separate texture map with each triangle (or pair of triangles). More space-efficient techniques involve unwrapping the surface onto one or more maps,

Once the (u; v) coordinates for each triangle have been fixed, the projection equations mapping from texture (u,v) to an image j's pixel (uj,vj) coordinates can be obtained by concatenating the affine (u,v)->(X,Y,Z) mapping with the perspective homography (X,Y,Z) -> (uj ; vj) . The color values for the (u; v) texture map can then be re-sampled and stored, or the original image can itself be used as the texture source using projective texture mapping . The situation becomes more involved when more than one source image is available for appearance recovery, which is the usual case.

One possibility is to use a view-dependent texture map (Section 14.1.1), in which different source image (or combination of source images) is used for each polygonal face based on the angles between the virtual camera, the surface normals, and the source images.

In some situations, e.g., when used models in traditional 3D games, it is preferable merge all of the source images into a single coherent texture map during pre-processing. Ideally, each surface triangle should select the source image where it is seen most directly (perpendicular to its normal) and at the resolution best matching the texture map resolution. This can be posed as a graph cut optimization problem, where the smoothness term encourages adjacent triangles to use similar source images, followed by blending to compensate for exposure differences.

Neural" texture map representations can also be used as an alternative to RGB fields techniques in more detail. These kinds of approaches produce good results when the lighting stays fixed with respect to the object, i.e., when the camera moves around the object or space. When the lighting is strongly directional, however, and the object is being moved relative to this lighting, strong shading effects or specularities may be present, which will interfere with the reliable recovery of a texture (albedo) map. In this case, it is preferable to explicitly undo the shading effects by modeling the light source directions and estimating the surface reflectance properties while recovering the texture map.

The results of one such approach, where the specularities are first removed while estimating the matte reflectance component (albedo) and then later re- introduced by estimating the specular component ks in a Torrance–Sparrow reflection model .

4.7.1 Estimating BRDFs

A more ambitious approach to the problem of view-dependent appearance modeling is to estimate a general bidirectional reflectance distribution function (BRDF) for each point on an object's surface.The BRDF can be written as where the e subscript now represents the emitted rather than the reflected light directions.

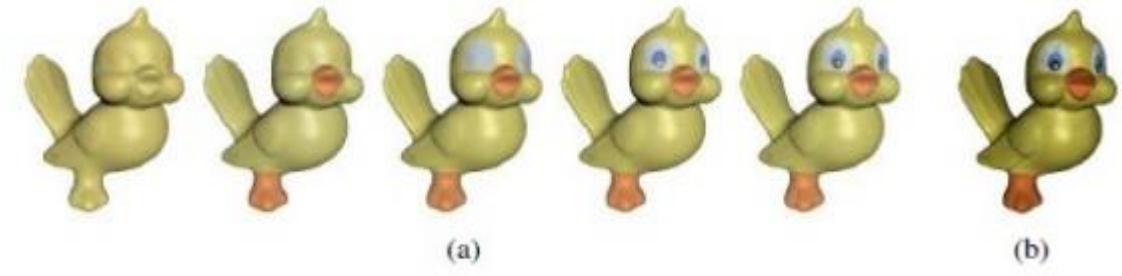

Fig 4.13 Image-Based Reconstruction

To build up their models, they first associate a lumitexel, which contains a 3D position, a surface normal, and a set of sparse radiance samples, with each surface point. Next, they cluster such lumitexels into materials that share common properties, using a Lafortune reflectance model and a divisive clustering approach.

Finally, to model detailed spatially varying appearance, each lumitexel (surface point) is projected onto the basis of clustered appearance models. More recent approaches to recovering spatially varying BRDFs (SVBRDFs) either start with RGB-D scanners, flash/noflash image pairs , or use deep learning approaches to simultaneously estimate surface normals and appearance models . Even more sophisticated systems can also estimate shape and environmental lighting from range scanner sequences or single monocular images .

While most of the techniques discussed in this section require large numbers of views to estimate surface properties, an interesting challenge is to take these techniques out of the lab and into the real world, and to combine them with regular and internet photo image- based modeling approaches.

4.7.2 Application: 3D Model Capture

The techniques described in this chapter for building complete 3D models from multiple images and then recovering their surface appearance have opened up a whole new range of applications that often go under the name 3D photography.

An example of a more recent commercial photogrammetric modeling system that can be used for both object and scene capture is Pix4D, whose website shows a wonderful example of a 3D texture-mapped castle reconstructed from both regular and aerial drone photographs. An alternative to such fully automated systems is to put the user in the loop in what is sometimes called interactive computer vision.

PART A

1. What does "Shape from X" refer to in computer vision?
2. State one key difference between passive and active range finding.
3. What is a surface representation in 3D reconstruction?
4. Define a point-based representation.
5. What are volumetric representations used for in 3D modeling?
6. Mention one example of a model-based reconstruction technique.
7. What is the primary purpose of recovering albedo in texture mapping?
8. How is texture mapping different from surface reconstruction?
9. State one advantage of point clouds over meshes.
10. Name one sensor commonly used for active range finding.

PART B

1. Explain the concept of "Shape from X" and give two examples of such techniques.
2. Describe the working of an active range finding system and its role in 3D data acquisition.
3. Compare point-based and surface-based representations in terms of memory, accuracy, and application.
4. Discuss the structure and purpose of volumetric representations in 3D reconstruction.
5. Explain how model-based reconstruction can assist in handling occlusions or missing data.
6. Describe the steps involved in recovering surface normals and albedos from multiple images.
7. What challenges arise when converting point clouds into surface representations?
8. Illustrate and explain the voxel-based method in volumetric reconstruction.
9. Write a short note on the differences between depth maps and albedo maps.
10. How does recovering texture maps enhance realism in 3D models?

PART C

1. Discuss various "Shape from X" techniques such as shape from shading, stereo, texture, and silhouette. Explain their principles, advantages, and limitations.
2. Describe in detail the complete process of active range finding, including triangulation, time-of-flight, and structured light methods. Include use cases.
3. Explain surface representations in 3D reconstruction, comparing meshes, splines, and implicit surfaces. Use examples and diagrams.
4. Analyze point-based representations in 3D graphics. Discuss how they are generated, stored, and used. Compare with mesh-based methods.

5. Evaluate volumetric representation methods such as voxels and signed distance functions (SDFs). Include applications and efficiency trade-offs.
6. Explain the principles, advantages, and challenges of model-based reconstruction. Use examples from face or body modeling.
7. Describe in detail how albedo and texture maps are recovered from image data. Discuss challenges such as lighting variation and shadowing.
8. Provide a comparative study between point-based, surface-based, and volumetric representations in terms of scalability, precision, and rendering.
9. Discuss a full pipeline of 3D reconstruction starting from image acquisition to textured model generation. Mention all intermediate steps.
10. Illustrate how learned priors (deep learning-based models) are used in model-based 3D reconstruction and texture recovery. Discuss current trends.

UNIT-V

IMAGE-BASED RENDERING AND RECOGNITION

View interpolation - Layered depth images - Light fields and Lumigraphs - Environment mattes - Video-based rendering -Object detection - Face recognition - Instance recognition - Category recognition - Context and scene understanding - Recognition databases and test sets.

5.1 VIEW INTERPOLATION :

In view interpolation, pairs of rendered images are combined with their precomputed depth maps to generate interpolated views that mimic what a virtual camera would see in between the two reference views.

View interpolation combines two ideas that were previously used in computer vision and computer graphics. The first is the idea of pairing a recovered depth map with the reference image used in its computation and then using the resulting texture-mapped 3D model to generate novel views .

The second is the idea of morphing, where correspondences between pairs of images are used to warp each reference image to an in-between location while simultaneously cross-dissolving between the two warped images.

First, both source images are warped to the novel view, using both the knowledge of the reference and virtual 3D camera pose along with each image's depth map. The depth maps are represented as quad trees for both space and rendering time efficiency

During the forward warping process, multiple pixels (which occlude one another) may land on the same destination pixel. To resolve this conflict, either a z-buffer depth value can be associated with each destination pixel or the images can be warped in back-to-front order, which can be computed based on the knowledge of epipolar geometry.

Once the two reference images have been warped to the novel view , they can be merged to create a coherent composite . Whenever one of the images has a hole (illustrated as a cyan pixel), the other image is used as the final value.

When both images have pixels to contribute, these can be blended as in usual morphing, i.e., according to the relative distances between the virtual and source cameras.

The final step in view interpolation is to fill any remaining holes or cracks due to the forward warping process or lack of source data (scene visibility).

In the case where the two reference images are views of a non-rigid scene, e.g., a person smiling in one image and frowning in the other, view morphing, which combines ideas from view interpolation with regular morphing, can be used. A depth map fitted to a face can also be used to synthesize a view from a longer distance, removing the enlarged nose and other facial features common to "selfie" photography.

While the original view interpolation paper describes how to generate novel views based on similar precomputed (linear perspective) images, argues that cylindrical images should be used to store the precomputed rendering or real-world images. It also proposes using environment maps (cylindrical, cubic, or spherical) as source images for view interpolation.

Fig 5.1 View Interpolation

5.1.1 View-Dependent Texture Maps

View-dependent texture maps are closely related to view interpolation. Instead of associating a separate depth map with each input image, a single 3D model is created for the scene, but different images are used as texture map sources depending on the virtual camera's current position.

In more detail, given a new virtual camera position, the similarity of this camera's view of each polygon (or pixel) is compared to that of potential source images. The images are then blended using a weighting that is inversely proportional to the angles_i between the virtual view and the source views. Even though the geometric model can be fairly coarse , blending different views gives a strong sense of more detailed geometry because of the visual motion between corresponding pixels.

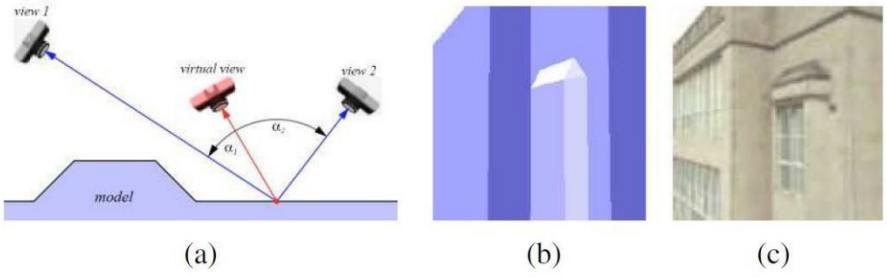

Fig 5.2 View-Dependent Texture Mapping

While the original paper performs the weighted blend computation separately at each pixel or coarsened polygon face, follow-on work by presents a more efficient implementation based on precomputing contributions for various portions of viewing space and then using projective texture mapping.

The idea of view-dependent texture mapping has been used in a large number of subsequent image-based rendering systems, including facial modeling and animation and 3D scanning and visualization.

To provide even more realism in their Facade system, also include a model-based stereo component, which computes an offset (parallax) map for each coarse planar facet of their model. They call the resulting analysis and rendering system a hybrid geometry- and image-based approach, as it uses traditional 3D geometric modelling to create the global 3D model, but then uses local depth offsets, along with view interpolation, to add visual realism.

Instead of warping per-pixel depth maps or coarser triangulated geometry (as in unstructured Lumigraphs, Section 14.3.1), it is also possible to use super pixels as the basic primitives being warped.

5.1.2 Application: Photo Tourism

While view interpolation was originally developed to accelerate the rendering of 3D scenes on low-powered processors and systems without graphics acceleration, it turns out that it can be applied directly to large collections of casually acquired photographs.

To perform an image-based exploration of the resulting sea of images, Photo Tourism first associates a 3D proxy with each image. While a triangulated mesh obtained from the point cloud can sometimes form a suitable proxy, e.g., for outdoor terrain models, a simple dominant plane fit to the 3D points visible in each image often performs better, because it does not contain any erroneous segments or connections that pop out as artifacts.

As automated 3D modeling techniques continue to improve, however, the pendulum may swing back to more detailed 3D geometry. One example is the hybrid rendering system , who use dense per-image depth maps for the well-reconstructed portions of each image and 3D colored point clouds for the less confident regions.

(a) (b) (c)

Fig 5.3 Photo Tourism

The resulting image-based navigation system lets users move from photo to photo, either by selecting cameras from a top-down view of the scene or by selecting regions of interest in an image, navigating to nearby views, or selecting related thumbnails. The system can also be used to annotate regions of images and to automatically propagate such annotations to other photographs.

The 3D planar proxies used in Photo Tourism and the related Photosynth system from Microsoft result in non- photo realistic transitions reminiscent of visual effects such as “page flips”. Selecting a stable 3D axis for all the planes can reduce the amount of swimming and enhance the perception of 3D

It is also possible to automatically detect objects in the scene that are seen from multiple views and create “orbits” of viewpoints around such objects. Furthermore, nearby images in both 3D position and viewing direction can be linked to create “virtual paths”, which can then be used to navigate between arbitrary pairs of images, such as those you might take yourself while walking around a popular tourist site (Snavely, Gargetal. 2008).

It produce higher-resolution quality assessments of image-based modeling and rendering system using what they call virtual rephotography. The spatial matching of image features and regions performed by Photo Tourism can also be used to infer more information from large image collections. For example, it show how the match graph between images of popular tourist sites can be used to find the most iconic (commonly photographed) objects in the collection, along with their related tags.

5.2 LAYERED DEPTH IMAGES

Traditional view interpolation techniques associate a single depth map with each source or reference image. Unfortunately, when such a depth map is warped to a novel view, holes and cracks inevitably appear behind the foreground objects.

One way to alleviate this problem is to keep several depth and color values (depth pixels) at every pixel in a reference image (or, at least for pixels near foreground–background transitions). The resulting data structure, which is called a layered depth image (LDI), can be used to render new views using a back-to-front forward warping (splatting) algorithm .

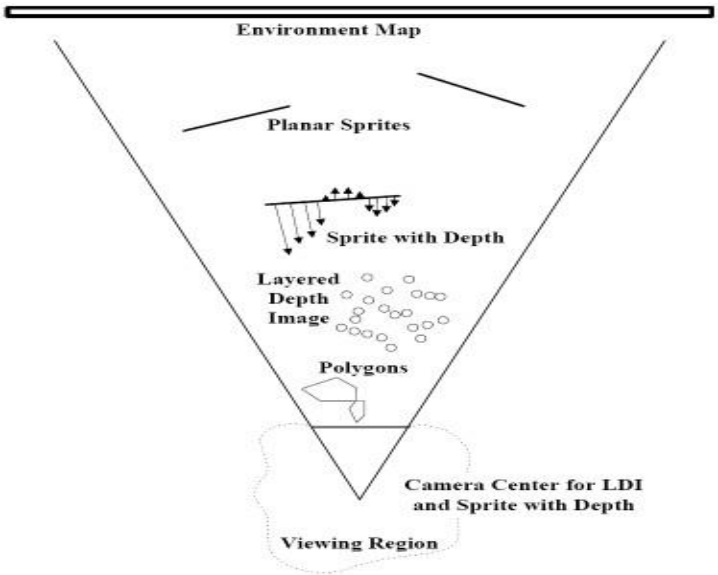

Fig 5.4 Layered Depth Images

5.2.1 Impostors, Sprites, And Layers

An alternative to keeping lists of color-depth values at each pixel, as is done in the LDI, is to organize objects into different layers or sprites. The term sprite originates in the computer game industry, where it is used to designate flat animated characters in games such as Pac- Man or Mario Bros. When put into a 3D setting, such objects are often called impostors, because they use a piece of flat, alpha-matted geometry to represent simplified versions of 3D objects that are far away from the camera . In computer vision, such representations are usually called layers.

While flat layers can often serve as an adequate representation of geometry and appearance for far-away objects, better geometric fidelity can be achieved by also modeling the per-pixel offsets relative to a base plane and 14.6a–b. Such representations are called plane plus parallax it is also possible to paint in depth layers.

Results of applying such a two-pass rendering algorithm. From this still image, you can appreciate that the foreground sprites look more rounded; however, to fully appreciate the improvement in realism, you would have to look at the actual animated sequence. Sprites with depth can also be rendered using conventional graphics hardware, describe GPU implementations of both real-time stereo matching and real-time forward and inverse rendering algorithms.

An alternative to constructing a small number of layers is to discretize the viewing frustum subtending a layered depth image into a large number of fronto-parallel planes, each of which contains RGBA values an MPI representation derived from a stereo image pair along with a novel synthesized view.

MPIs are easier to derive from pairs or collections of stereo images than true (minimal) layered representations because there is a 1:1 correspondence between pixels (actually, voxels) in a plane sweep cost volume and an MPI. However, they are not as compact and can lead to tearing artifacts once the viewpoint exceeds a certain range. Since their initial development for novel view extrapolation, i.e., “stereo magnification”,.

MPIs have found a wide range of applications in image- based rendering, including extension to multiple input images and faster inference , MPIs, and large view extrapolations. Another important application of layers is in the modeling of reflections. When the reflector (e.g., a glass pane) is planar, the reflection forms a virtual image, which can be modelled as a separate layer Figures 9.16–9.17, so long as additive (instead of over) compositing is used to combine the reflected and transmitted images.

An example of a two-layer decomposition reconstructed from a short video clip, which can be re-rendered from novel views by adding warped versions of the two layers (each of which has its own depth map). When the reflective surface is curved, a quasi-stable virtual image may still be available, although this depends on the local variations in

principal curvatures.

The modeling of reflections is one of the advantages attributed to layered representations such as MPIs, although in these papers over compositing is still used, which results in plausible but not physically correct renderings.

5.2.3 Application: 3D photography

Fig 5.5 Image-Based Rendering Of Scenes Using Multiple Additive Layers

It has also underpinned much of the research in 3D shape and appearance capture and modeling we studied in the previous chapter and more specifically. Until recently, however, while the required multiple images could be captured with hand-held cameras, desktop or laptop computers were required to process and interactively view the images.

The ability to capture, construct, and widely share such 3D models has dramatically increased in the last few years and now goes under the name of 3D photography.

It describe their Casual 3D Photography system, which takes a sequence of overlapping images taken from a moving camera and then uses a combination of structure from motion, multi-view stereo, and 3D image warping and stitching to construct two-layer partial panoramas that can be viewed on a computer.

The Instant 3D system builds a similar system, but starts with the depth images available from newer dual-camera smartphones to significantly speed up the process. Note, however, that the individual depth images are not metric, i.e., related to true depth with a single global scalar transformation, so must be deformably warped before being stitched together.

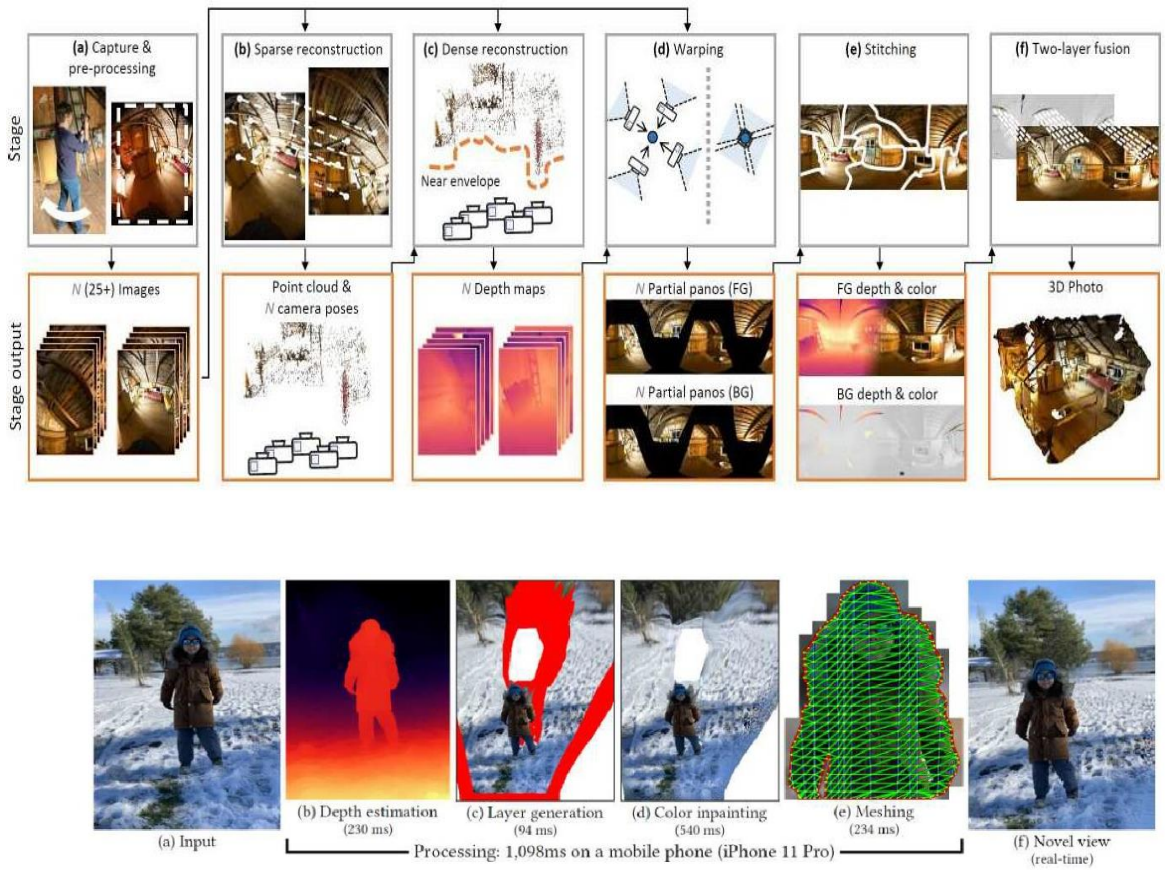

Fig 5.6 System For Capturing And Modeling 3D Scenes From Handheld Photographs

A texture atlas is then constructed to compactly store the pixel color values while also supporting multiple layers. While these systems produce beautiful wide 3D images that can create a true sense of immersion (“being there”), much more practical and fast solutions can be constructed using a single depth image.

5.3 LIGHT FIELDS AND LUMIGRAPHS

While image-based rendering approaches can synthesize scene renderings from novel viewpoints, they raise the following more general question: Is is possible to capture and render the appearance of a scene from all Possible viewpoints and, if so, what is the complexity of the resulting structure?

Let us assume that we are looking at a static scene, i.e., one where the objects and illuminants are fixed, and only the observer is moving around. Under these conditions, we can describe each image by the location and orientation of the virtual camera (6 dof) as well as its intrinsics (e.g., its focal length). However, if we capture a two-dimensional spherical image around each possible camera location, we can re-render any view from this information.

Thus, taking the cross-product of the three-dimensional space of camera positions with the 2D space of spherical images, which forms the basis of the image-based rendering system.

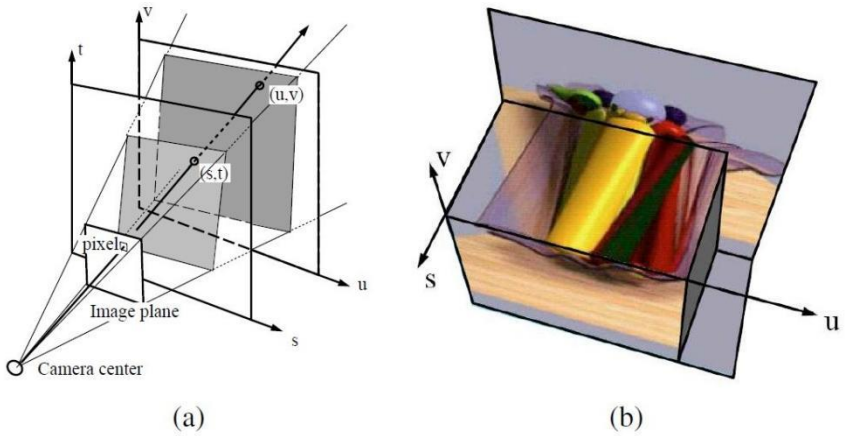

Fig 5.7 The Lumigraph

Notice, however, that when there is no light dispersion in the scene, i.e., no smoke or fog, all the coincident rays along a portion of free space (between solid or refractive objects) have the same color value. Under these conditions, we can reduce the 5D plenoptic function to the 4D light field of all possible rays. To make the parameterization of this 4D function simpler, let us put two planes in the 3D scene roughly bounding the area of interest

Consider the ray (s; u) corresponding to the dashed line in which intersects the object's surface at a distance z from the uv plane.

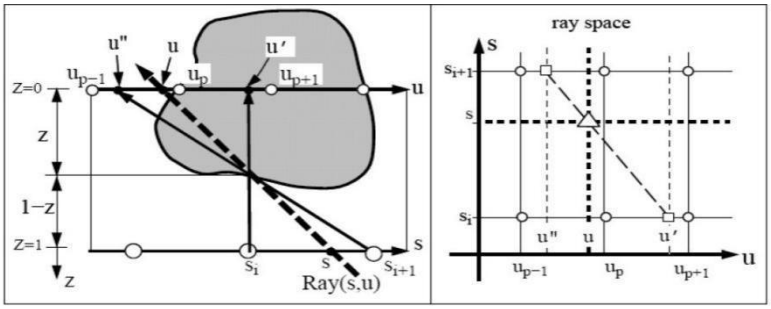

Fig 5.8 Depth Compensation In The Lumigraph

When we look up the pixel's color in camera si (assuming that the light field is discretely sampled on a regular 4D (s; t; u; v) grid), the actual pixel coordinate is u0, instead of the original u value specified by the (s; u) ray.

Similarly, for camera si+1 (where si _ s _ si+1), pixel address u00 is used. Thus, instead of using quadri-linear interpolation of the nearest sampled (s; t; u; v) values around a given ray to determine its color, the (u; v) values are modified for each discrete (si; ti) camera. Same reasoning in ray space. Here, the original continuous valued (s; u) ray is represented by a triangle and the nearby sampled discrete values are shown as circles. Instead of just blending the four nearest samples, as would be

indicated by the vertical and horizontal dashed lines, the modified (si; u0) and (si+1; u00) values are sampled instead and their values are then blended.

The resulting rendering system produces images of much better quality than a proxy- free light field and is the method of choice whenever 3D geometry can be inferred. In subsequent work, Isaksen, McMillan, and Gortler (2000) show how a planar proxy for the scene, which is a simpler 3D model, can be used to simplify the resampling equations.

They also describe how to create synthetic aperture photos, which mimic what might be seen by a wide-aperture lens, by blending more nearby samples . Because of the large number of images involved, light fields and Lumigraphs can be quite voluminous to store and transmit. Fortunately, as you can tel, there is a tremendous amount of redundancy (coherence) in a light field, which can be made even more explicit by first computing a 3D model, as in the Lumigraph.

5.3.1 Unstructured Lumigraph

When the images in a Lumigraph are acquired in an unstructured (irregular) manner, it can be counterproductive to resample the resulting light rays into a regularly binned (s; t; u; v) data structure. This is both because resampling always introduces a certain amount of aliasing and because the resulting gridded light field can be populated very sparsely or irregularly.

The alternative is to render directly from the acquired images, by finding for each light ray in a virtual camera the closest pixels in the original images. The unstructured Lumigraph rendering (ULR) system describes how to select such pixels by combining a number of fidelity criteria, including epipole consistency (distance of rays to a source camera's center), angular deviation (similar incidence direction on the surface), resolution (similar sampling density along the surface), continuity (to nearby pixels), and consistency (along the ray).

To make the algorithm more efficient, the computations are performed by discretizing the virtual camera's image plane using a regular grid overlaid with the polyhedral object mesh model and the input camera centers of projection and interpolating the weighting functions between vertices.

The unstructured Lumigraph generalizes previous work in both image-based rendering and light field rendering. When the input cameras are gridded, the ULR behaves the same way as regular Lumigraph rendering. When fewer cameras are available but the geometry is accurate, the algorithm behaves similarly to view-dependent texture mapping. If RGB-D depth images are available, these can be fused into lower-resolution proxies that can be combined with higher-resolution source images at rendering time. And while the original ULR paper uses manually constructed rules for determining pixel weights.

5.3.2 Surface Light Fields

Of course, using a two-plane parameterization for a light field is not the only possible choice. As we mentioned on the topic of light field compression, if we know the 3D shape of the object or scene whose light field is being modeled, we can effectively compress the field because nearby rays emanating from nearby surface elements have similar color values.

In fact, if the object is totally diffuse, ignoring occlusions, which can be handled using 3D graphics algorithms or z-buffering, all rays passing through a given surface point will have the same color value. Hence, the light field "collapses" to the usual 2D texture-map defined over an object's surface. Conversely, if the surface is totally specular (e.g., mirrored), each surface point reflects a miniature copy of the environment surrounding that point. In the absence of inter-reflections (e.g., a convex object in a large open space), each surface point simply reflects the far-field environment map, which again is two-dimensional.

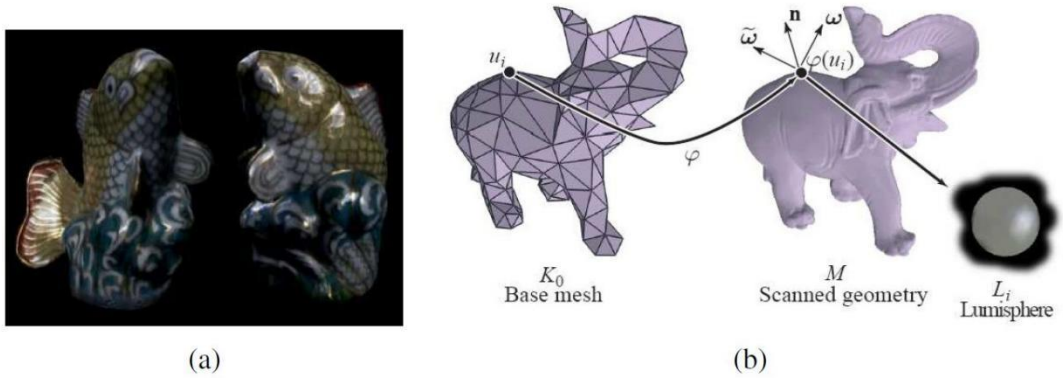

Fig 5.9 Surface Light Fields

Then the Lumisphere of all rays emanating from each surface point is estimated or captured . Nearby Lumispheres will be highly correlated and hence amenable to both compression and manipulation.

To estimate the diffuse component of each Lumisphere, a median filtering over all visible exiting directions is first performed for each channel. Once this has been subtracted from the Lumisphere, the remaining values, which should consist mostly of the specular components, are reflected around the local surface normal, which turns each Lumisphere into a copy of the local environment around that point. Nearby Lumispheres can then be compressed using predictive coding, vector quantization, or principal component analysis.

The decomposition into a diffuse and specular component can also be used to perform editing or manipulation operations, such as re-painting the surface, changing the specular component of the reflection (e.g., by blurring or sharpening the specular Lumispheres), or even geometrically deforming the object while preserving detailed surface appearance.

They then estimate a simple piecewise-constant BRDF model to account for the specular components. It also estimate the specular reflectance map, which is a convolution of the environment map with the object's specular BRDF. Additional techniques to estimate spatially varying BRDFs.

5.3.3 Applications:

5.3.3.1 Concentric Mosaics

A useful and simple version of light field rendering is a panoramic image with parallax, i.e., a video or series of photographs taken from a camera swinging in front of some rotation point. Such panoramas can be captured by placing a camera on a boom on a tripod, or even more simply, by holding a camera at arm's length while rotating your body around a fixed axis.

The resulting set of images can be thought of as a concentric mosaic or a layered depth panorama. The term "concentric mosaic" comes from a particular structure that can be used to re-bin all of the sampled rays, essentially associating each column of pixels with the "radius" of the concentric circle to which it is tangent.

While the virtual camera's motion is constrained to lie in the plane of the original cameras and within the radius of the original capture ring, the resulting experience can exhibit complex rendering phenomena, such as reflections and translucencies, which cannot be captured using a texture-mapped 3D model of the world.

5.3.3.2 Synthetic Re-Focusing

In addition to the interactive viewing of captured scenes and objects, light field rendering can be used to add synthetic depth of field effects to photographs.

In the computational photography, we mentioned how the depth estimates produced by modern dual-lens and/or dual-pixel smartphones can be used to synthetically blur photographs.

5.4 ENVIRONMENT MATTES

So far in this chapter, we have deal with view interpolation and light fields, which are techniques for modeling and rendering complex static scenes seen from different viewpoints.

What if, instead of moving around a virtual camera, we take a complex, refractive object, such as the water goblet, and place it in front of a new background? Instead of modeling the 4D space of rays emanating from a scene, we now need to model how each pixel in our view of this object refracts incident light coming from its environment.

Let us assume that if we trace a light ray from the camera at pixel (x,y) toward the object, it is reflected or refracted back out toward its environment at an angle (ϕ, θ).

If we assume that other objects and illuminants are sufficiently distant (the same assumption we made for surface light fields , this 4D mapping $(x,y) \rightarrow (\phi,\theta)$ captures all the information between a refractive object and its environment.

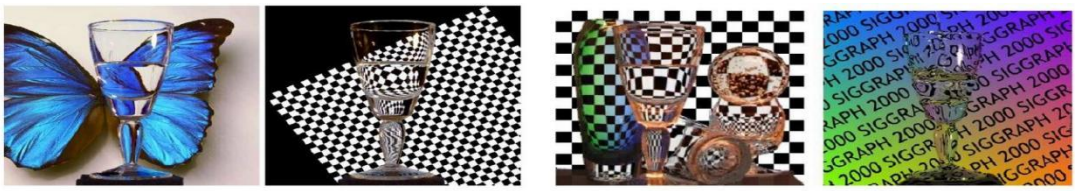

Fig 5.10 Environment Mattes

Premultiplied colors and opacities $(\alpha F, \alpha)$ Such a matte can then be composited onto a new background B using

$$C_i = \alpha_i F_i + (1 - \alpha_i) B_i,$$

where i is the pixel under consideration. In environment matting, we augment this equation with a reflective or refractive term to model indirect light paths between the environment and the camera. This indirect component Ii is modeled as

$$I_i = R_i \int A_i(\mathbf{x}) B(\mathbf{x}) d\mathbf{x},$$

where Ai is the rectangular area of support for that pixel, Ri is the colored reflectance or transmittance (for colored glossy surfaces or glass), and B(x) is the background (environment)image, which is integrated over the area Ai(x). It uses use a superposition of oriented Gaussians,is modelled by its centre c_{ij} , unrotated widths .

$$I_i = \sum_j R_{ij} \int G_{ij}(\mathbf{x}) B(\mathbf{x}) d\mathbf{x},$$

where each 2D Gaussian

$$G_{ij}(\mathbf{x}) = G_{2\mathrm{D}}(\mathbf{x}; \mathbf{c}_{ij}, \sigma_{ij}, \theta_{ij})$$

Given a representation for an environment matte, how can we go about estimating it for a particular object? The trick is to place the object in front of a monitor (or surrounded by a set of monitors), where we can change the illumination patterns B(x) and observe the value of each composite pixel C_i. We can use a variety of solid colored backgrounds to estimate each pixel's foreground color $\alpha_i F_i$ and partial coverage (opacity) α_i.

To estimate the area of support A_i in use a series of periodic horizontal and vertical solid stripes at different frequencies and phases, which is reminiscent of the structured light patterns used in active rangefinding .

Once an environment matte has been “pulled”, it is then a simple matter to replace the background with a new image B(x) to obtain a novel composite of the object placed in a different environment. The use of multiple backgrounds during the matting process, however, precludes the use of this technique with dynamic scenes, e.g., water pouring into a glass. In this case, a single graded color background can be used to estimate a single 2D monochromatic displacement for each pixel.

5.4.1 Higher-Dimensional Light Fields

As you can tell from the preceding discussion, an environment matte in principle maps every pixel (x; y) into a 4D distribution over light rays and is, hence, a six-dimensional representation.

In this case, we need a mapping from every incoming 4D light ray to every potential exiting 4D light ray, which is an 8D representation. If we use the same trick as with surface light fields, we can parameterize each surface point by its 4D BRDF to reduce this mapping back down to 6D, but this loses the ability to handle multiple refractive paths. If we want to handle dynamic light fields, we need to add another temporal dimension. Similarly, if we want a continuous distribution over wavelengths, this becomes another dimension. These examples illustrate how modeling the full complexity of a visual scene through sampling can be extremely expensive.

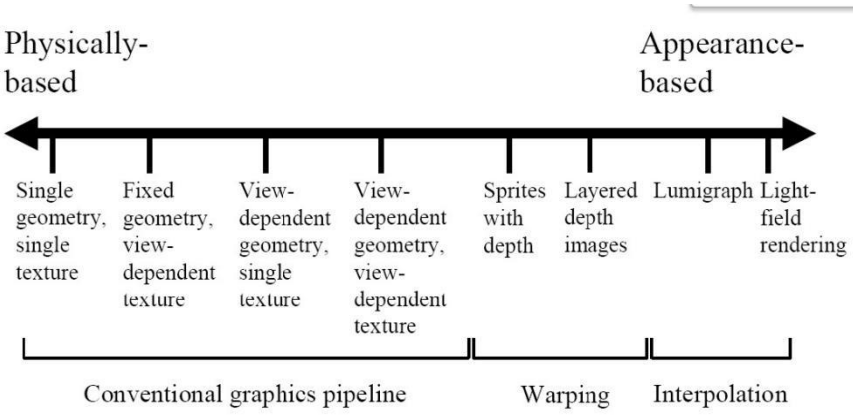

Fig 5.11 The Geometry-Image Continuum In Image-Based Rendering

5.4.2 The Modeling To Rendering Continuum

The image-based rendering representations and algorithms we have studied in this chapter span a continuum ranging from classic 3D texture-mapped models all the way to pure sampled ray-based representations such as light fields. Representations such as view-dependent texture maps and Lumigraphs still use a single global geometric model, but select the colors to map onto these surfaces from nearby images.

View-dependent geometry, e.g., multiple depth maps, sidestep the need for coherent 3D geometry, and can sometimes better model local non-rigid effects such as specular motion. Sprites with depth and layered depth images use image-based representations of both color and geometry and can be efficiently rendered using warping operations rather than 3D geometric rasterization.

The best choice of representation and rendering algorithm depends on both the quantity and quality of the input imagery as well as the intended application. When nearby views are being rendered, image-based representations capture more of the visual fidelity of the real world because they directly sample its appearance.

On the other hand, if only a few input images are available or the image-based models need to be manipulated, e.g., to change their shape or appearance, more abstract 3D representations such as geometric and local reflection models are a better fit. As we continue to capture and manipulate increasingly larger quantities of visual data, research into these aspects of image-based modeling and rendering will continue to evolve.

5.5 VIDEO-BASED RENDERING

In fact , a fair amount of work has been done in the area of video-based rendering and video-based animation, two terms introduced to denote the process of generating new video sequences from captured video footage.

An early example of such work is Video Rewrite , in which archival video footage is “re-animated” by having actors say new utterances. More recently, the term video-based rendering has been used by some researchers to denote the creation of virtual camera moves from a set of synchronized video cameras placed in a studio.

In this section, we present a number of video-based rendering systems and applications.We start with video-based animation, in which video footage is re-arranged or modified, e.g., in the capture and re-rendering of facial expressions.

Fig 5.12 Video Rewrite

5.5.1 Video-Based Animation

As we mentioned above, an early example of video-based animation is Video Rewrite, in which frames from original video footage are rearranged to match them to novel spoken utterances, e.g., for movie dubbing. This is similar in spirit to the way that

concatenative speech synthesis systems work (Taylor 2009).

In their system, Bregler, Covell, and Slaney (1997) first use speech recognition to extract phonemes from both the source video material and the novel audio stream. Phonemes are grouped into triphones (triplets of phonemes), as these better model the coarticulation effect present when people speak. Matching triphones are then found in the source footage and audio track.

The mouth images corresponding to the selected video frames are then cut and pasted into the desired video footage being re-animated or dubbed, with appropriate geometric transformations to account for head motion. During the analysis phase, features corresponding to the lips, chin, and head are tracked using computer vision techniques.

During synthesis, image morphing techniques are used to blend and stitch adjacent mouth shapes into a more coherent whole. In subsequent work, describe how to use a multidimensional morphable model combined with regularized trajectory synthesis to improve these results.

A more sophisticated version of this system, called face transfer, uses a novel source video, instead of just an audio track, to drive the animation of a previously captured video, i.e., to re-render a video of a talking head with the appropriate visual speech, expression, and head pose elements .

5.5.2 Video Textures

Video-based animation is a powerful means of creating photo-realistic videos by re- purposing existing video footage to match some other desired activity or script. For example, many websites use images or videos to highlight their destinations, e.g., to portray attractive beaches with surf and palm trees waving in the wind.

We can view these transition probabilities as encoding the hidden Markov model (HMM) that underlies a stochastic video generation process. Sometimes, it is not possible to find exactly matching subsequences in the original video.In this case, morphing, i.e., warping and blending frames during transitions can be used to hide the visual differences . If the motion is chaotic enough, as in a bonfire or a waterfall, simple blending (extended cross- dissolves) may be sufficient. Improved transitions can also be obtained by performing 3D graph cuts on the spatio-temporal volume around a transition.

Video textures need not be restricted to chaotic random phenomena such as fire, wind, and water. Pleasing video textures can be created of people, e.g., a smiling face or someone running on a treadmill. When multiple people or objects are moving independently, we must first segment the video into independently moving regions and animate each region separately. It is also possible to create large panoramic video textures from a slowly panning camera.

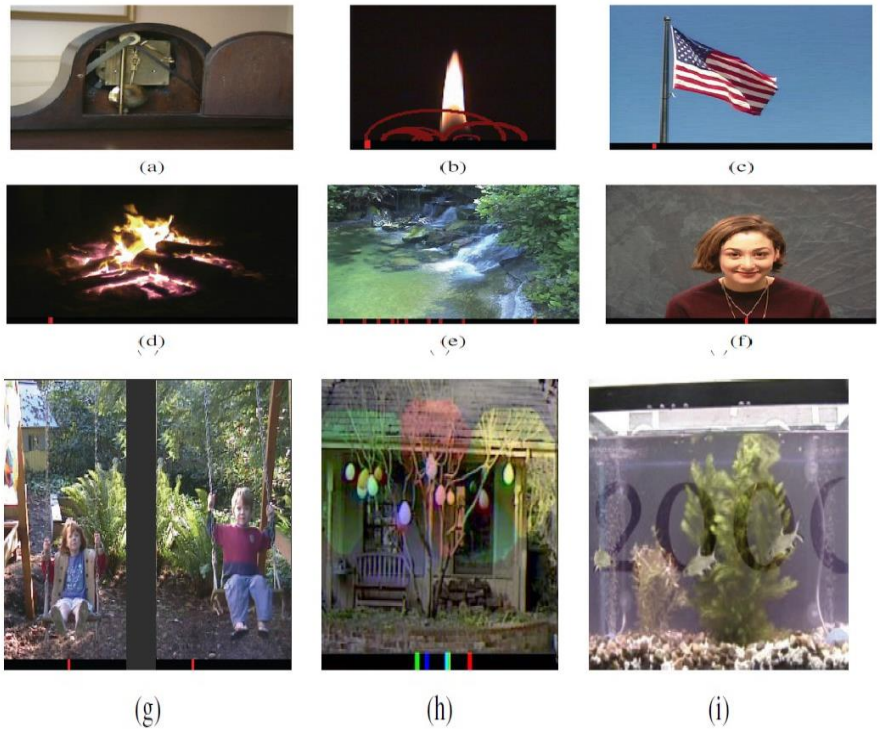

Fig 5.13 Video Textures

Instead of just playing back the original frames in a stochastic (random) manner, video textures can also be used to create scripted or interactive animations. If we extract individual elements, such as fish in a fishtank into separate video sprites, we can animate them along prespecified paths (by matching the path direction with the original sprite motion) to make our video elements move in a desired fashion.

A more recent example of controlling video sprites is the Vid2Player system, which models the movements and shots of tennis players to create synthetic video-realistic games. In fact, work on video textures inspired research on systems that re-synthesize new motion sequences from motion capture data, which some people refer to as “mocap soup”.

While video textures primarily analyze the video as a sequence of frames (or regions) that can be re-arranged in time, temporal textures and dynamic textures treat the video as a 3D spatio-temporal volume with textural

5.5.3 3D And Free-Viewpoint Video

In the last decade, the 3D movies have become an established medium. Currently, such releases are filmed using stereoscopic camera rigs and displayed in theaters (or at home) to viewers wearing polarized glasses. In the future, however, home audiences may wish to view such movies with multi-zone auto-stereoscopic displays, where each person gets his or her own customized stereo stream and can move around a scene to see it from different perspectives.

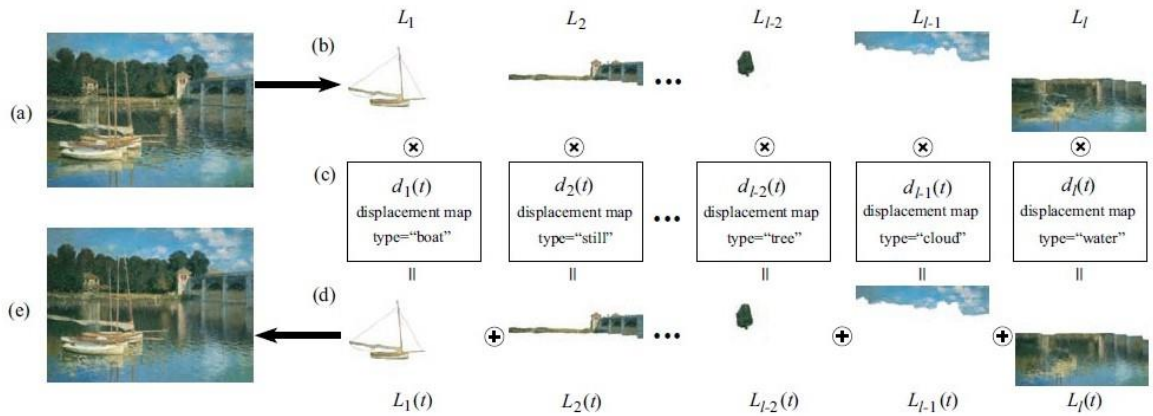

Fig 5.14 Animating still pictures

The stereo matching techniques developed in the computer vision community along with image-based rendering (view interpolation) techniques from graphics are both essential components in such scenarios, which are sometimes called free-viewpoint video or virtual viewpoint video.

The Virtual Viewpoint Video system, on the other hand, associates a two-layer depth map with each input image, which allows them to accurately model occlusion effects such as the mixed pixels that occur at object boundaries. Their system, which consists of eight synchronized video cameras connected to a disk array, first uses segmentation-based stereo to extract a depth map for each input image

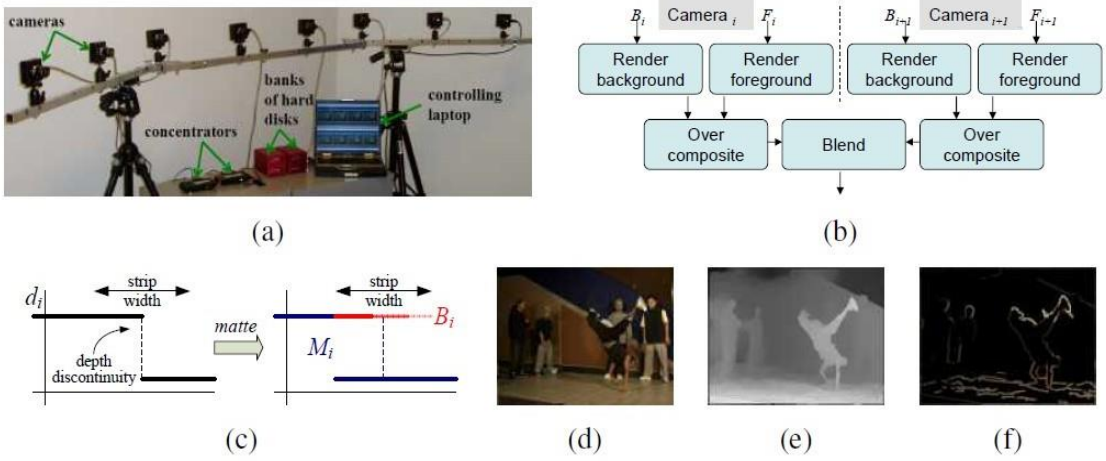

Fig 5.15 Video View Interpolation

Near object boundaries (depth discontinuities), the background layer is extended along a strip behind the foreground object and its color is estimated from the neighboring images where it is not occluded

Automated matting techniques are then used to estimate the fractional opacity and color of boundary pixels in the foreground layer. At render time, given a new virtual camera that lies between two of the original cameras, the layers in the neighboring cameras are rendered as texture-mapped triangles and the foreground

layer (which may have fractional opacities) is then composited over the background layer. The resulting two images are merged and blended by comparing their respective z-buffer values. (Whenever the two z-values are sufficiently close, a linear blend of the two colors is computed.)

5.5.4 Application: Video-Based Walkthroughs

However, extending this idea to larger settings, e.g., beyond a single room, can become tedious and data-intensive. Instead, a natural way to explore a space is often to just walk through it along some prespecified paths, just as museums or home tours guide users along a particular path, say down the middle of each room. Similarly, city-level exploration can be achieved by driving down the middle of each street and allowing the user to branch at each intersection.

In their system, the synchronized video streams from the six cameras are stitched together into 360° panoramas using a variety of techniques develop specifically for this project.

Because the cameras do not share the same center of projection, parallax between the cameras can lead to ghosting in the overlapping fields of view. To remove this, a multi-perspective plane sweep stereo algorithm is used to estimate per- pixel depths at each column in the overlap area. To calibrate the cameras relative to each other, the camera is spun in place and a constrained structure from motion algorithm is used to estimate the relative camera poses and intrinsics.

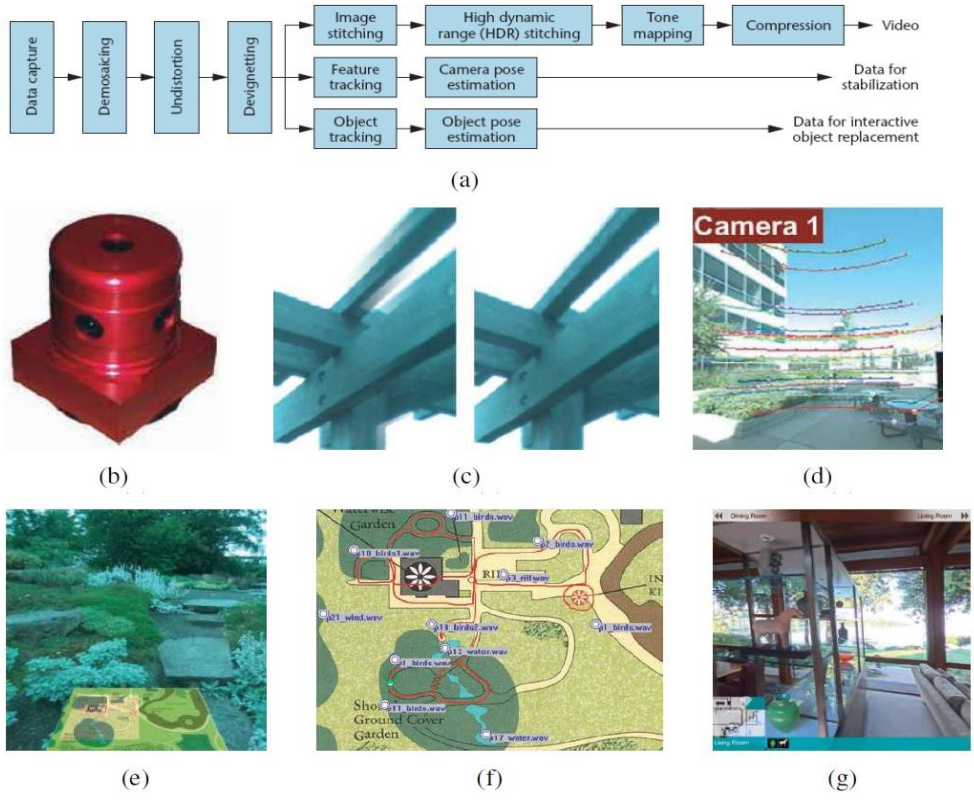

Fig 5.16 Video-Based Walkthroughs

5.6 OBJECT DETECTION

5.6.1 What Is Object Detection

Object Detection is a computer vision technique to locate objects in an image or in a video. Organizations and researchers are spending huge time and resources to uncover this capability. When we humans look at a picture, we can quickly identify the objects and their respective position in an image.

We can quickly categorize if it is an apple or a car or a human being. We can also determine from any angle. The reason is that our minds have been trained in such a way that it can identify various objects. Even if the size of an object gets smaller or bigger, we are able to locate them and detect them. The goal is to replicate this decision-making intelligence using Machine Learning and Deep Learning.

5.6.2 What Is Object classification , Object Localization And Object Detection

Look at the images of a vacuum cleaner. The image classification solutions to classify such images into “a Vacuum Cleaner” or “not.” So we could have easily labeled the first image as a vacuum cleaner.

On the other hand, localization refers to finding the position of the object in an image. So when we do Image Localization, it means that the algorithm is having a dual responsibility of classifying an image as well as drawing a bounding box around it, which is depicted in the second image. In the first image, we have a vacuum cleaner, and in the second image, we have localized it.

Fig 5.17 Object Detection Means Identifying, Localizaton Of The Object

To scale the solution, we can have multiple objects in the same image and even multiple objects of different categories in the same image, and we have to identify all of them. And draw the bounding boxes around them. An example can be of a solution trained to detect cars. On a busy road, there will be many cars, and hence

the solution should be able to detect each of them and draw bounding boxes around them. Object detection is surely a fantastic solution. We will now discuss the major object detection use cases in the next section.

5.6.3 Defining The Terms Object Classification And Localisation

Object classification determines which specific objects are within an image or video actuallyare. It labels these objects.

Object localization specifically tracks where objects are located in an image or video. This determines the position of any object within a piece of visual content.

5.6.4 Applications of Object Classification and Detection :

- Facial Recogntion
- Cancer Detection
- Other Vehicles
- Pedestrians
- Cyclists
- Traffic signals
- Lane markings
- Construction

5.6.5 Use Cases Of Object Detection

Deep Learning has expanded many capabilities across domains and organizations. Object detection is a key one and is a very powerful solution which is making huge ripples in our business and personal world. The major use cases of object detection are Object Detection is the key intelligence behind autonomous driving technology. It allows the users to detect the cars, pedestrians, the background, motorbikes, and so on to improve road safety.

We can detect objects in the hands of people, and the solution can be used for security and monitoring purposes. Surveillance systems can be made much more intelligent and accurate. Crowd control systems can be made more sophisticated, and the reaction time will be reduced.

A solution might be used for detecting objects in a shopping basket, and it can be used by the retailers for the automated transactions. This will speed up the overall process with less manual intervention.

Object Detection is also used in testing of mechanical systems and on manufacturing lines. We can detect objects present on the products which might be contaminating the product quality.

In the medical world, the identification of diseases by analyzing the images of a body part will help in faster treatment of the diseases.

There are very less areas where the usage is not envisioned. It is one of the areas which are highly researched, and every day new progress is being made in this domain. Organizations and researchers across the globe are making huge ripples in this area and creating path- breaking solutions.

5.7 FACE RECOGNITION

Among the various recognition tasks that computers are asked to perform, face recognition is the one where they have arguably had the most success. Face recognition can be used in a variety of additional applications, including human–computer interaction (HCI), identity verification, desktop login, parental controls, and patient monitoring, but it also has the potential for misuse .

Some of the earliest approaches to face recognition involved finding the locations of distinctive image features, such as the eyes, nose, and mouth, and measuring the distances between these feature locations . Other approaches relied on comparing gray-level images projected onto lower dimensional subspaces called eigen faces and jointly modeling shape and appearance variations (while discounting pose variations) using active appearance models .

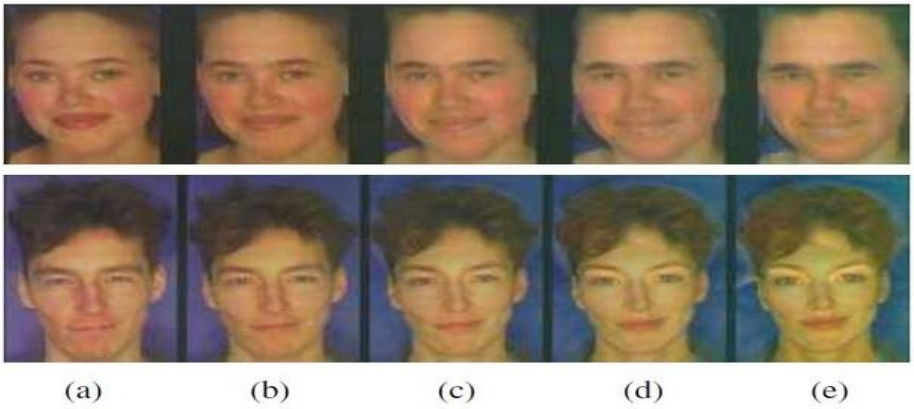

Fig 5.18 Manipulating Facial Appearance Through Shape And Color

Active Appearance and 3D Shape Models

The need to use modular or view-based eigenspaces for face recognition , is symptomatic of a more general observation, i.e., that facial appearance and identifiability depend as much on shape as they do on color or texture (which is what eigenfaces capture).

Later techniques such as local feature analysis and elastic bunch graph matching combined local filter responses (jets) at distinctive feature locations together with shape models to perform recognition.

A visually compelling example of why both shape and texture are important is the work of Rowland and Perrett (1995), who manually traced the contours of facial features and then used these contours to normalize (warp) each image to a canonical shape. After analyzing both the shape and color images for deviations from the mean,

they were able to associate certain shape and color deformations with personal characteristics such as age and gender.

Their work demonstrates that both shape and color have an important influence on the perception of such characteristics. Around the same time, researchers in computer vision were beginning to use simultaneous shape deformations and texture interpolation to model the variability in facial appearance caused by identity or expression, developing techniques such as Active Shape Models, 3D Morphable Models,and Elastic Bunch Graph Matching

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{U}_s\mathbf{a}$$
$$\mathbf{t} = \bar{\mathbf{t}} + \mathbf{U}_t\mathbf{a},$$

The active appearance models (AAMs) both the variation in the shape of an image s, which is normally encoded by the location of key feature points on the image, as well as the variation in texture t, which is normalized to a canonical shape before being analyzed. Both shape and texture are represented as deviations from a mean shape s and texture t, data.where the eigenvectors in Us and Ut have been pre-scaled (whitened) so that unit vectors in a represent one standard deviation of variation observed in the training

As you can see, the same appearance parameters a in simultaneously control both the shape and texture deformations from the mean, which makes sense if we believe them to be correlated. how moving three standard deviations along each of the first four principal directions ends up changing several correlated factors in a person's appearance, including expression, gender, age, and identity.

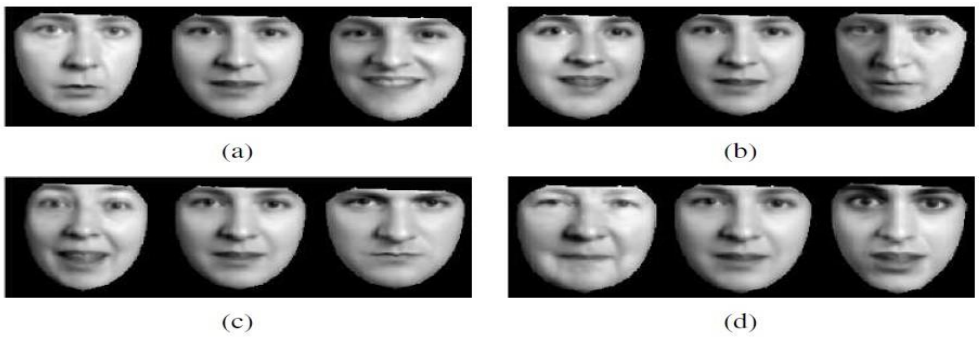

Fig 5.19 Principal Modes Of Variation In Active Appearance Models

5.7.1 Facial Recognition Using Deep Learning

Prompted by the dramatic success of deep networks in image categorization,

face recognition researchers started using deep neural network backbones as part of their systems. two stages, which was one of the first systems to realize large gains using deep networks.

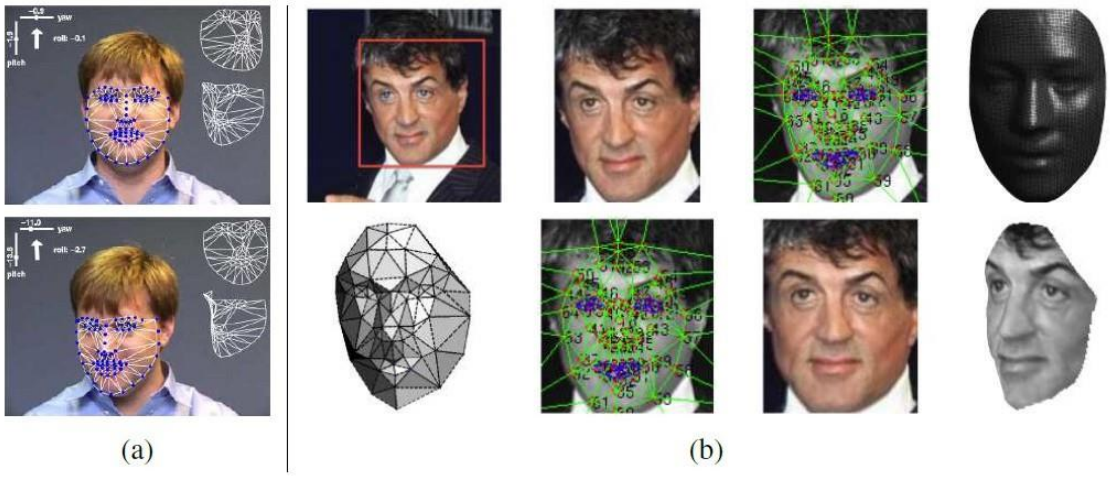

Fig 5.20 Head Tracking And Frontalization

In their system, a landmark-based pre-processing frontalization step is used to convert the original color image into a well-cropped front-looking face. Then, a deep locally connected network (where the convolution kernels can vary spatially) is fed into two final fully connected layers before classification.

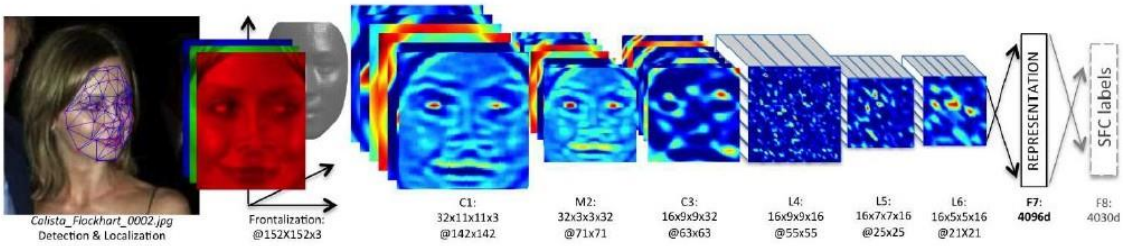

Fig 5.21 DeepFace Architecture

Some of the more recent deep face recognizers omit frontalization stage instead use data augmentation (Section 5.3.3) to create synthetic inputs with a larger variety of poses.It provide an excellent tutorial and survey on deep face recognition, including a list of widely used training and testing datasets, a discussion of frontalization and dataset augmentation, and a section on training losses

5.7.2 Personal Photo Collections

In addition to digital cameras automatically finding faces to aid in auto-focusing and video cameras finding faces in video conferencing to center on the speaker (either mechanically or digitally), face detection has found its way into most consumer-level photo organization packages and photo sharing sites. Finding faces and allowing users to tag them makes it easier to find photos of selected people at a later date or to automatically share them with friends.

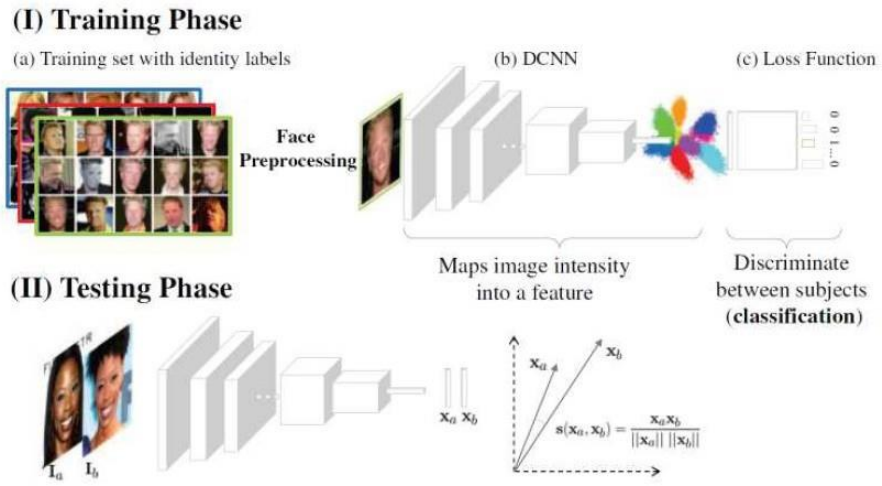

Fig 5.22 A Typical Modern Deep Face Recognition Architecture

In fact, the ability to tag friends in photos is one of the more popular features on Facebook. Sometimes, however, faces can be hard to find and recognize, especially if they are small, turned away from the camera, or otherwise occluded. In such cases, combining face recognition with person detection and clothes recognition can be very effective, as illustrated .

5.8 INSTANCE RECOGNITION

General object recognition falls into two broad categories, namely instance recognition and class recognition. The former involves re-recognizing a known 2D or 3D rigid object, potentially being viewed from a novel viewpoint, against a cluttered background, and with partial occlusions.

Fig 5.23 Clutter Background With Pratial Occlusions

The latter, which is also known as category-level or generic object recognition , is the much more challenging problem of recognizing any instance of a particular general class, such as “cat”, “car”, or “bicycle”.

Over the years, many different algorithms have been developed for instance recognition. surveys earlier approaches, which focused on extracting lines, contours, or 3D surfaces from images and matching them to known 3D object models.

Another popular approach was to acquire images from a large set of viewpoints and illuminations and to represent them using an eigenspace decomposition. More recent approaches tend to use viewpoint-invariant 2D features.

After extracting informative sparse 2D features from both the new image and the images in the database, image features are matched against the object database, using one of the sparse feature matching strategie .

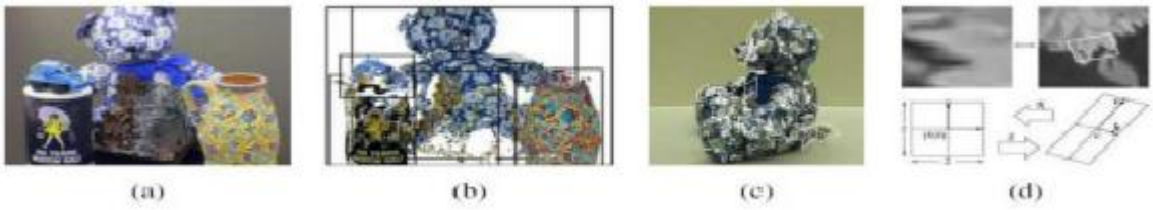

Fig 5.24 3D Object Recognition With Affine Regions

Whenever a sufficient number of matches have been found, they are verified by finding a geometric transformation that aligns the two sets of features.

5.8.1 Geometric Alignment

To recognize one or more instances of some known objects, such as those shown in the left column, the recognition system first extracts a set of interest points in each database image and stores the associated descriptors (and original positions) in an indexing structure such as a search tree.

At recognition time, features are extracted from the new image and compared against the stored object features.

Whenever a sufficient number of matching features (say, three or more) are found for a given object, the system then invokes a match verification stage, whose job is to determine whether the spatial arrangement of matching features is consistent with those in the database image.

5.9 CATEGORY RECOGNITION

Category Recognition, also known as object category recognition or image categorization, involves assigning a label or category to an entire image based on the objects or scenes it contains.

The goal is to identify the overall content or theme of an image without necessarily distinguishing individual instances or objects within it.

5.9.1 Scope:

Whole-Image Recognition: Category recognition focuses on recognizing and classifying the entire content of an image rather than identifying specific instances or details within the image.

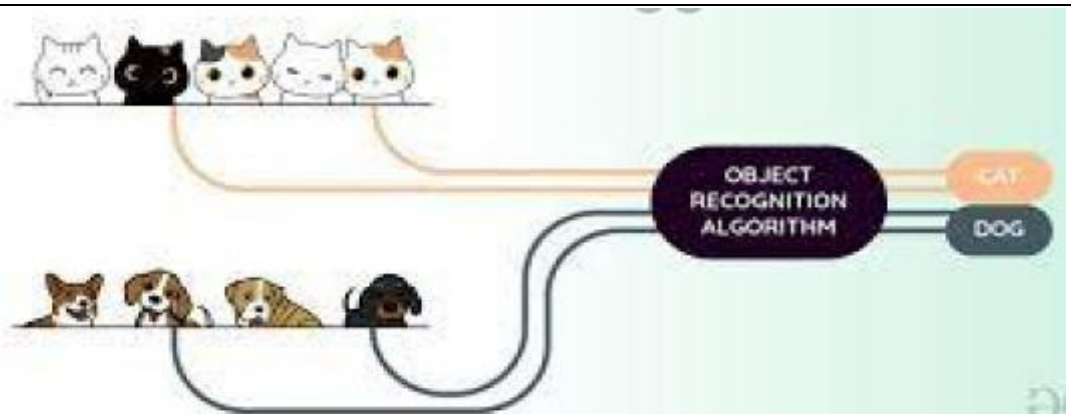

Fig 5.25 Object Recognition Algorithm

5.9.2 Methods:

Convolutional Neural Networks (CNNs): Deep learning methods, particularly CNNs, have shown significant success in image categorization tasks, learning hierarchical features.

Bag-of-Visual-Words: Traditional computer vision approaches that represent images as histograms of visual words based on local features.

Transfer Learning: Leveraging pre-trained models on large datasets and fine-tuning them for specific category recognition tasks.

5.9.3 Applications:

Image Tagging: Automatically assigning relevant tags or labels to images for organization and retrieval.

Content-Based Image Retrieval (CBIR): Enabling the retrieval of images based on their content rather than textual metadata.

Visual Search: Powering applications where users can search for similar images by providing a sample image.

5.9.4 Challenges:

Intra-class Variability: Dealing with variations within the same category, such as different poses, lighting conditions, or object appearances.

Fine-grained Categorization: Recognizing subtle differences between closely related categories.

Handling Clutter: Recognizing the main category in images with complex backgrounds or multiple objects.

5.9.5 Datasets:

ImageNet: A large-scale dataset commonly used for image classification tasks, consisting of a vast variety of object categories.

CIFAR-10 and CIFAR-100: Datasets with smaller images and multiple categories, often used for benchmarking image categorization models.

Open Images: A dataset with a large number of annotated images covering diverse categories.

5.9.10 Evaluation Metrics:

Top-k Accuracy : Measures the proportion of images for which the correct category is among the top-k predicted categories.

Confusion Matrix: Provides a detailed breakdown of correct and incorrect predictions across different categories.

5.9.11 Multi-Label Categorization:

Definition: Extends category recognition to handle cases where an image may belong to multiple categories simultaneously.

Applications: Useful in scenarios where images can have complex content that falls into multiple distinct categories.

5.9.12 Real-world Applications:

E-commerce: Categorizing product images for online shopping platforms.

Content Moderation: Identifying and categorizing content for moderation purposes, such as detecting inappropriate or unsafe content.

Automated Tagging: Automatically categorizing and tagging images in digital libraries or social media platforms.

5.9.13 Future Trends:

Weakly Supervised Learning: Exploring methods that require less annotated data for training, such as weakly supervised or self-supervised learning for category recognition.

Interpretable Models: Developing models that provide insights into the decision-making process for better interpretability and trustworthiness.

5.10 CONTEXT AND SCENE UNDERSTANDING

Context and Scene Understanding in computer vision involves comprehending the overall context of a scene, recognizing relationships between objects, and understanding the semantic meaning of the visual elements within an image or a sequence of images.

5.10.1 Scene Understanding vs. Object Recognition:

Object Recognition: Focuses on identifying and categorizing individual objects within an image.

Scene Understanding: Encompasses a broader understanding of the relationships, interactions, and contextual information that characterize the overall scene.

5.10.2 Elements of Context and Scene Understanding:

Spatial Relationships: Understanding the spatial arrangement and relative positions of objects within a scene.

Temporal Context: Incorporating information from a sequence of images or frames to understand changes and dynamics over time.

Semantic Context: Recognizing the semantic relationships and meanings associated with objects and their interactions.

5.10.3 Methods:

Graph-based Representations: Modeling scenes as graphs, where nodes represent objects and edges represent relationships, to capture contextual information.

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM): Utilizing recurrent architectures for processing sequences of images and capturing temporal context.

Graph Neural Networks (GNNs): Applying GNNs to model complex relationships and dependencies in scenes.

5.10.4 Applications:

Autonomous Vehicles: Scene understanding is critical for autonomous navigation, as it involves comprehending the road, traffic, and dynamic elements in the environment Robotics: Enabling robots to understand and navigate through indoor and outdoor environments.

Augmented Reality: Integrating virtual objects into the real world in a way that considers the context and relationships with the physical environment.

Surveillance and Security: Enhancing the analysis of surveillance footage by understanding activities and anomalies in scenes.

5.10.6 Challenges:

Ambiguity: Scenes can be ambiguous, and objects may have multiple interpretations depending on context.

Scale and Complexity: Handling large-scale scenes with numerous objects and complex interactions.

Dynamic Environments: Adapting to changes in scenes over time, especially in dynamic and unpredictable environments .

5.10.7 Semantic Segmentation and Scene Parsing:

Semantic Segmentation: Assigning semantic labels to individual pixels in an image, providing a detailed understanding of object boundaries.

Scene Parsing: Extending semantic segmentation to recognize and understand the overall scene layout and context.

5.10.8 Hierarchical Representations:

Multiscale Representations: Capturing information at multiple scales, from individual objects to the overall scene layout.

Hierarchical Models: Employing hierarchical structures to represent objects, sub-scenes, and the global context.

5.10.9 Context-Aware Object Recognition:

Definition: Enhancing object recognition by considering contextual information surrounding objects.

Example: Understanding that a "bat" in a scene with a ball and a glove is likely associated with the sport of baseball.

5.10.10 Future Directions:

Cross-Modal Understanding: Integrating information from different modalities, such as combining visual and textual information for a more comprehensive understanding.

Explainability and Interpretability: Developing models that can provide explanations for their decisions to enhance transparency and trust.

5.11 RECOGNITION DATABASES AND TEST SETS

Recognition databases and test sets play a crucial role in the development and evaluation of computer vision algorithms, providing standardized datasets for training, validating, and benchmarking various recognition tasks. These datasets often cover a wide range of domains, from object recognition to scene understanding. Here are some commonly used recognition databases and test sets:

5.11.1 ImageNet:

Task: Image Classification, Object Recognition

Description: ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is a widely used dataset for image classification and object detection. It includes millions of labeled images across thousands of categories.

5.11.2 COCO (Common Objects in Context):

Tasks: Object Detection, Instance Segmentation, Keypoint Detection

Description: COCO is a large-scale dataset that includes complex scenes with multiple objects and diverse annotations. It is commonly used for evaluating algorithms in object detection and segmentation tasks.

5.11.3 PASCAL VOC (Visual Object Classes):

Tasks: Object Detection, Image Segmentation, Object Recognition

Description: PASCAL VOC datasets provide annotated images with various object categories.

5.11.4 MOT (Multiple Object Tracking) Datasets:

Task: Multiple Object Tracking

Description: MOT datasets focus on tracking multiple objects in video sequences. They include challenges related to object occlusion, appearance changes, and interactions.

5.11.5 KITTI Vision Benchmark Suite:

Tasks: Object Detection, Stereo, Visual Odometry

Description: KITTI dataset is designed for autonomous driving research and includes tasks such as object detection, stereo estimation, and visual odometry using data collected from a car.

5.11.6 ADE20K:

Tasks: Scene Parsing, Semantic Segmentation

Description: ADE20K is a dataset for semantic segmentation and scene parsing.

It contains images with detailed annotations for pixel-level object categories and scene labels.

5.11.7 Cityscapes:

Tasks: Semantic Segmentation, Instance Segmentation

Description: Cityscapes dataset focuses on urban scenes and is commonly used for semantic segmentation and instance segmentation tasks in the context of autonomous driving and robotics.

5.11.8 CelebA:

Tasks: Face Recognition, Attribute Recognition

Description: CelebA is a dataset containing images of celebrities with annotations for face recognition and attribute recognition tasks.

5.11.9 LFW : (Labelled Faces in the world) Task: Face Verification

Description: LFW dataset is widely used for face verification tasks, consisting of images of faces collected from the internet with labeled pairs of matching and non-matching faces.

5.11.10 Open Images Dataset:

Tasks: Object Detection, Image Classification

Description: Open Images Dataset is a large-scale dataset that includes images with annotations for object detection, image classification, and visual relationship prediction.

PART A

1. What is view interpolation in computer graphics?
2. Define layered depth image (LDI).
3. What is a light field in the context of image-based rendering?
4. State one difference between light fields and Lumigraphs.
5. What is an environment matte used for?
6. Name one application of video-based rendering.
7. What is object detection in computer vision?
8. Differentiate between face recognition and face detection.
9. Define instance recognition.
10. What is the role of context in scene understanding?

PART B

1. Explain the process of view interpolation and its significance in novel view synthesis.
2. Describe the structure and function of layered depth images. How do they differ from simple depth maps?
3. Compare light fields and Lumigraphs in terms of data acquisition and image rendering.
4. Discuss the principle of environment mattes and how they are used in compositing.
5. Explain the key steps in a video-based rendering pipeline.
6. Describe the process of object detection using deep learning models.
7. How does face recognition work? Outline the main steps in a typical recognition pipeline.
8. Differentiate between instance recognition and category recognition with suitable examples.
9. What is scene context in image understanding, and how does it assist object recognition?
10. Discuss the role of benchmark datasets (like ImageNet or COCO) in training and evaluating recognition models.

PART C

1. Discuss the challenges and techniques involved in view interpolation. Explain its role in synthesizing novel views from sparse camera setups.
2. Analyze the working of Layered Depth Images (LDI) in capturing multiple depths per pixel. How do LDIs help in occlusion handling? Include diagrams.
3. Explain the concept of light fields and Lumigraphs. Compare their data representation, acquisition techniques, and rendering strategies
4. Describe in detail how environment mattes are generated and used for integrating real and virtual objects. Discuss their use in augmented reality and VFX.

5. Discuss a complete pipeline for video-based rendering. Include key stages like view synthesis, camera tracking, temporal coherence, and output generation.
6. Explain in depth the modern approaches to object detection, focusing on methods like R-CNN, YOLO, or SSD. Compare accuracy and speed.
7. Discuss how face recognition has evolved with the advent of deep learning. Include examples of architectures like FaceNet or DeepFace.
8. Compare and contrast instance recognition and category recognition. Provide use cases and explain how training data requirements differ.
9. Describe techniques used in context-aware scene understanding. How do spatial, semantic, and co-occurrence cues improve recognition performance?
10. Evaluate the importance of recognition databases and test sets. Discuss common datasets, annotation methods, evaluation metrics, and dataset bias.