Summer of Science

Computer Vision

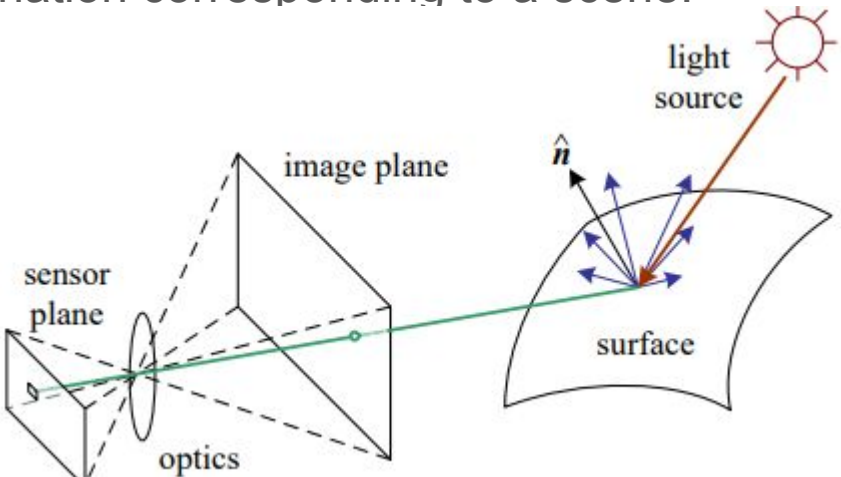
Midterm report 2025

By prem kumar lodhi(24B1847)
Under Snigdha Reddy

Content

1.Photometric Image Formation

When light hits an object's surface, it is scattered and reflected. Many different models have been developed to describe this interaction. In this section, we first describe the most general form, the bidirectional reflectance distribution function, and then look at some more specialized models, including the diffuse, specular, and Phong shading models. We also discuss how these models can be used to compute the global illumination corresponding to a scene.



1.1The Bidirectional Reflectance Distribution Function (BRDF)

Imagine a flashlight shining on a surface and a camera observing the reflected light. The BRDF tells you **how bright the surface appears** to the camera **from that angle**, given the incoming light direction.the BRDF is a fourdimensional function that describes how much of each wavelength arriving at an incident direction vˆi is emitted in a reflected direction vˆr. The function can be written in terms of the angles of the incident and reflected directions relative to the surface frame as

$$f_r(\omega_i, \omega_o) = \frac{dL_r(\omega_o)}{dE_i(\omega_i)}$$

units of **inverse steradians**

- fr is the BRDF
- dLr is the reflected radiance in direction ωo
- dEi is the incoming irradiance from direction ωi

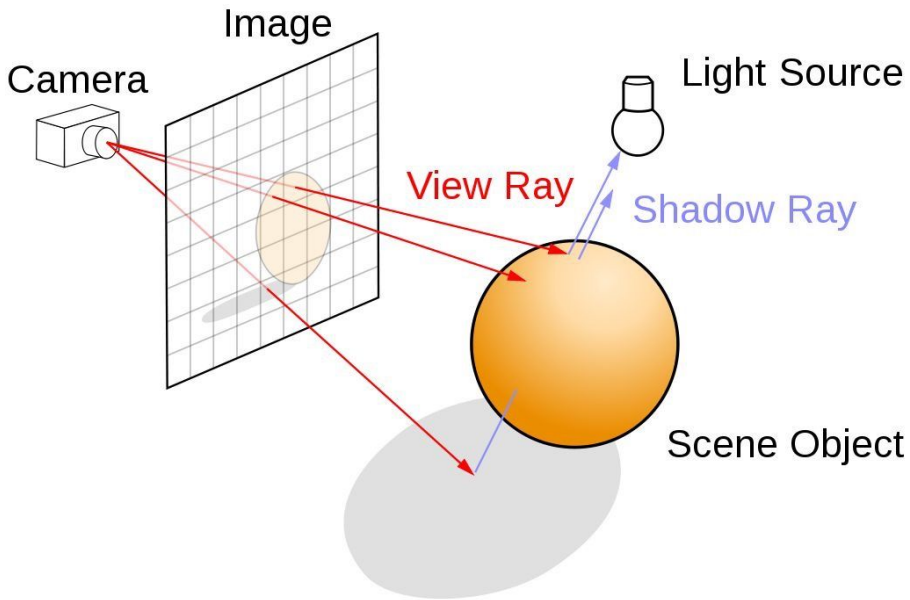
1.2.Matte Reflection

scatters light uniformly in all directions and is the phenomenon we most normally associate with shading, e.g., the smooth (non-shiny) variation of intensity with surface normal that is seen when observing a statue Diffuse reflection also often imparts a strong body color to the light since it is caused by selective absorption and re-emission of light inside the object's material. the amount of light depends on the angle between the incident light direction and the surface normal θi . This is because the surface area exposed to a given amount of light becomes larger at oblique angles, becoming completely self-shadowed as the outgoing surface normal points away from the light

1.3.Ray traycing

Ray tracing is a rendering technique that can realistically simulate the lighting of a scene and its objects by rendering physically accurate reflections, refractions, shadows, and indirect lighting. Ray tracing generates computer graphics images by tracing the path of light from the view camera (which determines your view into the scene), through the 2D viewing plane (pixel plane), out into the 3D scene, and back to the light sources. As it traverses the scene, the light may reflect from one object to another (causing reflections), be blocked by objects (causing shadows), or pass through transparent or semi-transparent objects (causing refractions). All of these interactions are combined to produce the final color and illumination of a pixel that is then displayed on the screen. This reverse tracing process of eye/camera to light source is chosen because it is far more efficient than tracing all light rays emitted from light sources in multiple directions.

Another way to think of ray tracing is to look around you, right now. The objects you're seeing are illuminated by beams of light. Now turn that around and follow the path of those beams backwards from your eye to the objects that light interacts with. That's ray tracing.The primary application of ray tracing is in computer graphics, both non-real-time (film and television) and real-time (video games). Other applications include those in architecture, engineering, and lighting design.



2.Feature Detection

Feature detection and matching are an essential component of many computer vision applications.The first kind of feature that you may notice are specific locations in the images, such as mountain peaks, building corners, doorways, or interestingly shaped patches of snow. These kinds of localized feature are often called keypoint features or interest points (or even corners) and are often described by the appearance of patches of pixels surrounding the point location.

Another class of important features are edges, e.g., the profile of mountains against the sky. These kinds of features can be matched based on their orientation and local appearance (edge profiles) and can also be good indicators of object boundaries and occlusion events in image sequences. Edges can be grouped into longer curves and straight line segments, which can be directly matched or analyzed to find vanishing points and hence internal and external camera parameters.

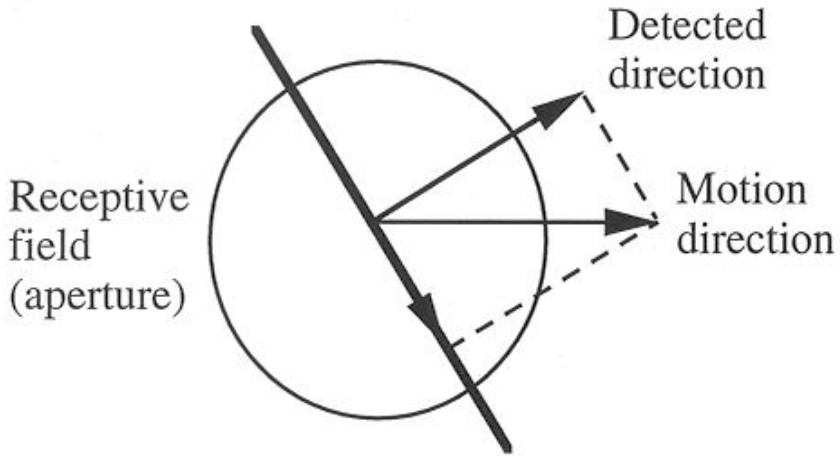
2.1.Point Feature Detection

There are two main approaches to finding feature points and their correspondences. The first is to find features in one image that can be accurately tracked using a local search technique, such as correlation or least squares. The second is to independently detect features in all the images under consideration and then match features based on their local appearance. The former approach is more suitable when images are taken from nearby viewpoints or in rapid succession (e.g., video sequences), while the latter is more suitable when a large amount of motion or appearance change is expected, e.g., in stitching together panoramas, establishing correspondences in wide baseline stereo, or performing object recognition.

we split the keypoint detection and matching pipeline into three separate stages. During the feature detection (extraction) stage (Section 2.1.1), each image is searched for locations that are likely to match well in other images. At the feature description stage (Section 2.1.2), each region around detected keypoint locations is converted into a more compact and stable (invariant) descriptor that can be matched against other descriptors.Feature matching stage (Section 2.1.3) efficiently searches for likely matching candidates in other images.

2.1.1 The Feature Detection (extraction) stage

textureless patches are nearly impossible to localize. Patches with large contrast changes (gradients) are easier to localize, although straight line segments at a single orientation suffer from the aperture problem, as shown schematically in Figure, i.e., it is only possible to align the patches along the direction normal to the edge direction.Gradients in at least two (significantly) different orientations are the easiest to localize.



Steps to detect feature points location

1. Compute the horizontal and vertical derivatives of the image Ix and Iy by convolving the original image with derivatives of Gaussians.

2. Compute the three images corresponding to the outer products of these gradients. (The matrix A is symmetric, so only three entries are needed.)

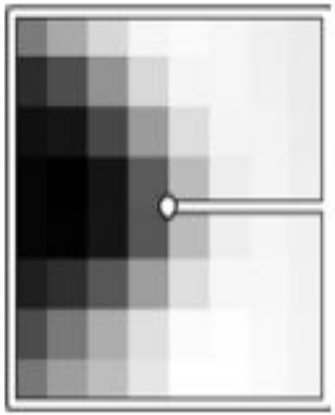
3. Convolve each of these images with a larger Gaussian.

4. Compute a scalar interest measure using one of the formulas discussed above.

5. Find local maxima above a certain threshold and report them as detected feature point locations.

2.1.2.The Feature Description Stage

After detecting features (keypoints), we must match them, i.e., we must determine which features come from corresponding locations in different images. In some situations, e.g., for video sequences (Shi and Tomasi 1994) or for stereo pairs that have been rectified, the local motion around each feature point may be mostly translational. In this case, simple error metrics, such as the sum of squared differences or normalized cross-correlation, can be used to directly compare the intensities in small patches around each feature point.Because feature points may not be exactly located, In most cases, however, the local appearance of features will change in orientation and scale, and sometimes even undergo affine deformations. Extracting a local scale, orientation, or affine frame estimate and then using this to resample the patch before forming the feature descriptor is thus usually preferable (as shown in Figure). Even after compensating for these changes, the local appearance of image patches will usually still vary from image to image.



MOPS descriptors are formed using an 8 × 8 sampling of bias and gain normalized intensity values, with a sample spacing of five pixels relative to the detection scale. This low frequency sampling gives the features some robustness to interest point location error and is achieved by sampling at a higher pyramid level than the detection scale.

2.1.3.Feature matching stage

Once we have extracted features and their descriptors from two or more images, the next step is to establish some preliminary feature matches between these images.The first is to select a matching strategy, which determines which correspondences are passed on to the next stage for further processing. The second is to devise efficient data structures and algorithms to perform this matching as quickly as possible.Once we have decided on a matching strategy, we still need to search efficiently for potential candidates. The simplest way to find all corresponding feature points is to compare all features against all other features in each pair of potentially matching images. Unfortunately, this is quadratic in the number of extracted features, which makes it impractical for most applications. A better approach is to devise an indexing structure, such as a multi-dimensional search tree or a hash table, to rapidly search for features near a given feature.

Such indexing structures can either be built for each image independently (which is useful if we want to only consider certain potential matches, e.g., searching for a particular object) or globally for all the images in a given database, which can potentially be faster, since it removes the need to iterate over each image. For extremely large databases (millions of images or more), even more efficient structures based on ideas from document retrieval can be used.

Recognizing objects in a cluttered scene.Two of the training images in the database are shown on the left. These are matched to the cluttered scene in the middle using SIFT features, shown as small squares in the right image. The affine warp of each recognized database image onto the scene is shown as a larger parallelogram in the right image.

2.2 Edge Detection

While interest points are useful for finding image locations that can be accurately matched in 2D, edge points are far more plentiful and often carry important semantic associations. For example, the boundaries of objects, which also correspond to occlusion events in 3D, are usually delineated by visible contours. Other kinds of edges correspond to shadow boundaries or crease edges, where surface orientation changes rapidly. Isolated edge points can also be grouped into longer curves or contours, as well as straight line segments.

Think of an image as a height field. On such a surface, edges occur at locations of steep slopes, or equivalently, in regions of closely packed contour lines (on a topographic map). A mathematical way to define the slope and direction of a surface is through its gradient,

$$\boldsymbol{J}(\boldsymbol{x}) = \nabla I(\boldsymbol{x}) = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})(\boldsymbol{x}).$$

the local gradient vector J points in the direction of steepest ascent in the intensity function. Its magnitude is an indication of the slope or strength of the variation, while its orientation points in a direction perpendicular to the local contour. Unfortunately, taking image derivatives accentuates high frequencies and hence amplifies noise, since the proportion of noise to signal is larger at high frequencies. It is therefore prudent to smooth the image with a low-pass filter prior to computing the gradient. Because we would like the response of our edge detector to be independent of orientation, a circularly symmetric smoothing filter is desirable.the Gaussian is the only separable circularly symmetric filter and so it is used in most edge detection algorithms.

Coloured Edge Detection

While most edge detection techniques have been developed for grayscale images, color images can provide additional information. For example, noticeable edges between iso-luminant colors (colors that have the same luminance) are useful cues but fail to be detected by grayscale edge operators. One simple approach is to combine the outputs of grayscale detectors run on each color band separately.7 However, some care must be taken. For example, if we simply sum up the gradients in each of the color bands, the signed gradients may actually cancel each other! (Consider, for example a pure red-to-green edge.) We could also detect edges independently in each band and then take the union of these, but this might lead to thickened or doubled edges that are hard to link. A better approach is to compute the oriented energy in each band, e.g., using a second-order steerable filter, and then sum up the orientation-weighted energies and find their joint best orientation. Unfortunately, the directional derivative of this energy may not have a closed form solution (as in the case of signed first-order steerable filters), so a simple zero crossing-based strategy cannot be used. However, the technique described by Elder and Zucker (1998) can be used to compute these zero crossings numerically instead.

Combined brightness, color, texture boundary detector. Successive rows show the outputs of the brightness gradient (BG), color gradient (CG), texture gradient (TG), and combined (BG+CG+TG) detectors. The final row shows human-labeled boundaries derived from a database of hand-segmented images

An alternative approach is to estimate local color statistics in regions around each pixel. This has the advantage that more sophisticated techniques (e.g., 3D color histograms) can be used to compare regional statistics and that additional measures, such as texture, can also be considered. Above Figure shows the output of such detectors.

Resources used till now

https://developer.nvidia.com/discover/ray-tracing

https://www.youtube.com/@Eigensteve

https://www.youtube.com/@DigitalSreeni

Book of CV by Richard Szeliski

seas.ucla.edu

google.com