Motion Analysis, Ray Tracing & Optical Flow

Overview

Motion Analysis in Computer Vision

- Understanding motion between frames
- Optical flow computation and analysis
- Object tracking systems and applications

Introduction to Ray Tracing

- Fundamentals of light transport
- Mathematical foundations
- Modern rendering techniques

Understanding Motion in Computer Vision

- Computer vision helps machines understand movement and objects in videos, just like how humans track moving objects with their eyes.

- When you watch a video, your brain naturally tracks moving objects. However, computers need special algorithms to understand this movement. We analyze how pixels change position between video frames

- Basic Concept of Motion:
    a. Motion is the change in position of objects between video frames
    b. Think of a video as a series of photos taken very quickly
    c. We need to understand where objects move between these photos

Why estimate motion?

- We live in a 4-D world (x,y,z,t)!

- Wide applications:
  - Motion detection and object tracking (surveillance etc.)
  - Correct for camera jitter (stabilization)
  - Align images (panoramic mosaics)
  - 3D shape reconstruction (shape from motion)
  - Video compression (MPEG)
  - Robotics (navigation etc.)
  - Entertainment: Special Effects, Sportscasting, Video Games

Types of Motion in Computer Vision

- Basic Types of Motion:
  - Translation: Objects moving in straight lines without changing orientation
    1. Example: A car moving straight down a highway
    2. Can be measured with (dx, dy) displacement vectors
    3. Simplest type of motion to track and analyze
  - Rotation: Objects spinning or changing orientation
    1. Example: A satellite rotating in space
    2. Measured by angle of rotation (θ)
    3. Can occur around different axes (X, Y, Z)
  - Scaling: Objects getting larger or smaller
    1. Example: A person walking towards or away from camera
    2. Changes in apparent size due to distance
    3. Measured by scale factors in X and Y directions

Types of Motion in Computer Vision

- Complex Motion Types:
  - Articulated Motion: Connected parts moving relatively
    1. Example: Human body movements, robot arms
    2. Multiple rigid parts connected at joints
    3. Each part can have independent rotation and translation
  - Non-Rigid Motion: Objects changing shape while moving
    1. Example: Clothing moving in wind, facial expressions
    2. Shape deformation along with position changes
    3. More challenging to track and analyze
  - Periodic Motion: Regular repeating patterns
    1. Example: Walking cycles, rotating fan blades
    2. Motion that repeats at fixed intervals
    3. Can be analyzed using frequency-based methods

Types of Motion in Computer Vision

- Camera-Based Motion:
  - Camera Translation: Camera moving through space
    1. Creates apparent motion of entire scene
    2. Example: Drone flying forward
  - Camera Rotation: Camera changing viewing direction
    1. Creates complex apparent motion patterns
    2. Example: Pan and tilt movements in surveillance
  - Combined Camera Motion: Mix of rotation and translation
    1. Most common in real-world scenarios
    2. Example: Hand-held camera movements

Introduction to Optical Flow

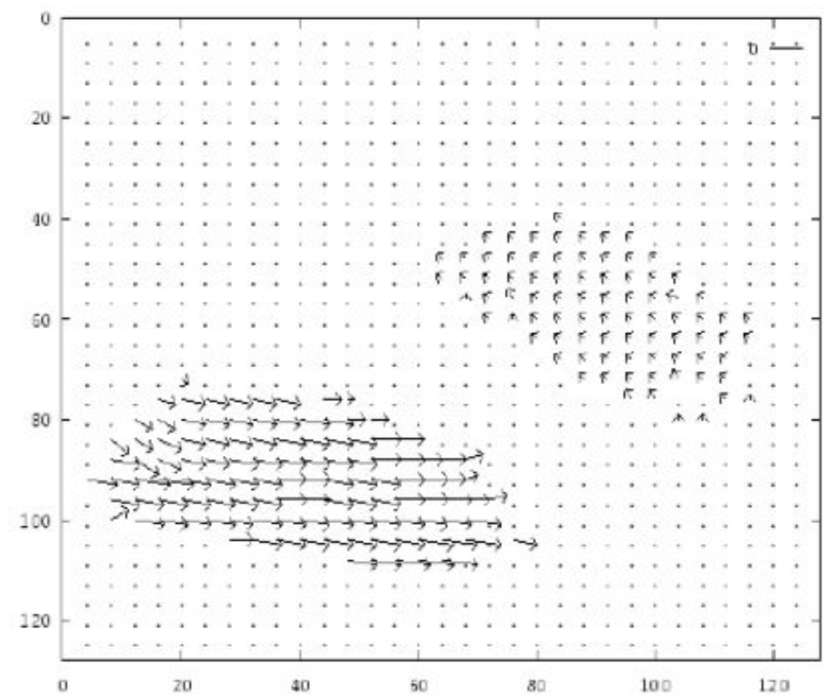
- One of the fundamental techniques used to analyze this motion - **Optical Flow**.

- Understanding Optical Flow-
  - Imagine looking out of a moving train - everything seems to flow past you
  - Optical flow measures this apparent motion for every pixel
  - It creates a "motion map" showing where everything is moving

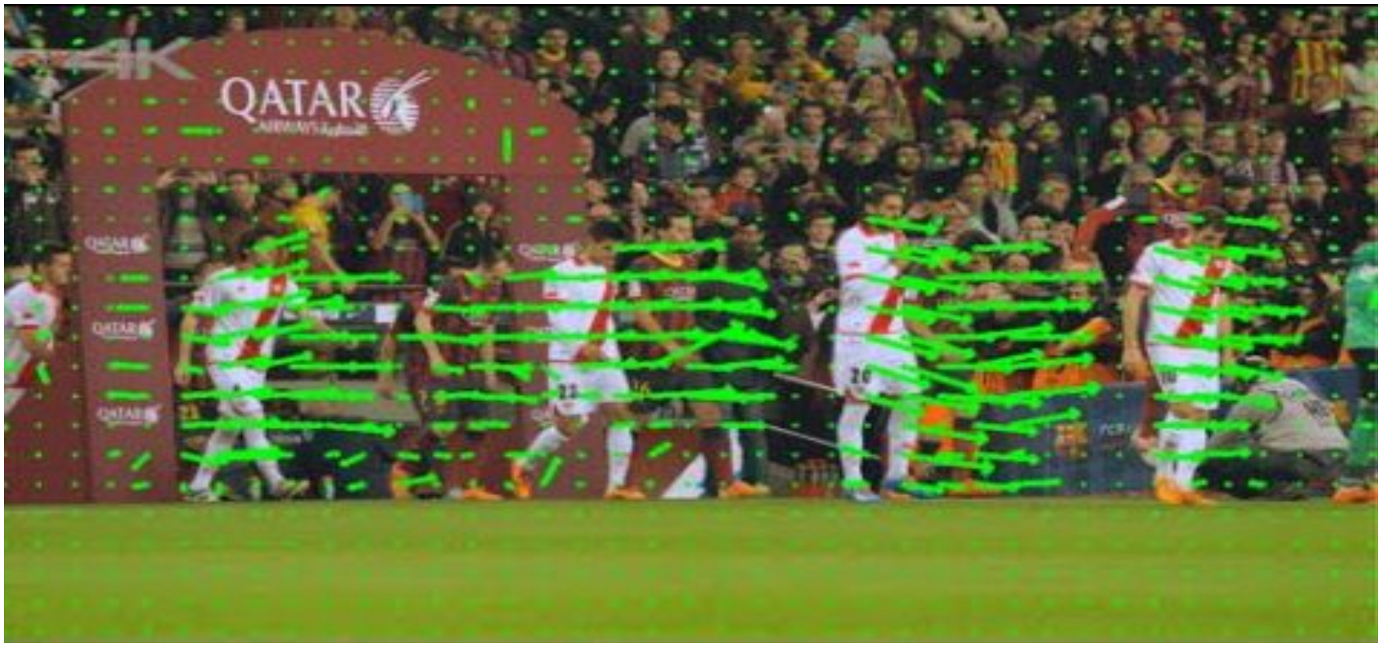
Introduction to Optical Flow

- How Optical Flow Works (simplified):
  a. Takes two consecutive video frames
  b. For each pixel, finds where it moved to in the next frame
  c. Creates arrows (vectors) showing direction and speed of motion

Example- Hamburg Taxi Sequence

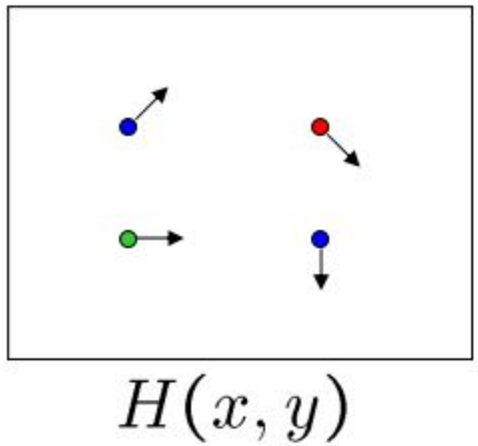


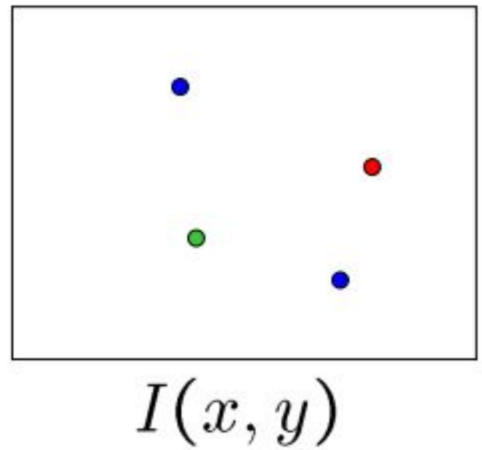
Visualizing Optical Flow in Sports Analysis



- The green arrows show the direction and magnitude of motion
- Each arrow represents the movement of players and objects on the field
- Motion vectors help track player movements and analyze game patterns

Estimating optical flow





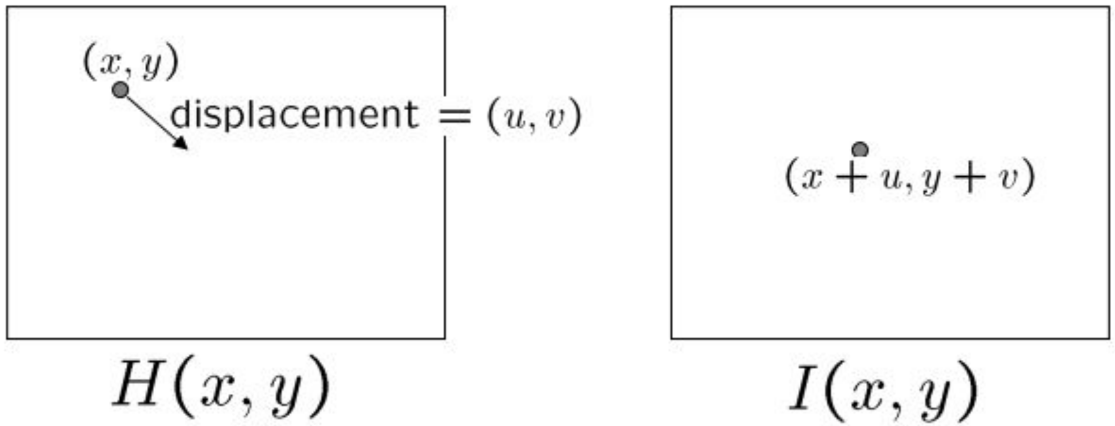
How to estimate pixel motion from image H to image I?

- Solve pixel correspondence problem
  - given a pixel in H, look for nearby pixels of the same color in I

Key assumptions-

- **Color constancy**: a point in H looks the same in I – For grayscale images, this is brightness constancy
- **Small motion**: points do not move very far
- **Spatial coherence**: points move like their neighbors

The brightness constancy constraint



Let's look at these constraints more closely

- Brightness constancy: $I(x, y, t-1) = I(x + u(x, y), y + v(x, y), t)$

- Linearizing the right side using Taylor expansion:

$$I(x, y, t-1) \approx I(x, y, t) + I_x\, u(x, y) + I_y\, v(x, y)$$

Hence, $I_x\, u + I_y\, v + I_t \approx 0$

The brightness constancy constraint

$$I_x u + I_y v + I_t = 0$$

How many equations and unknowns per pixel?

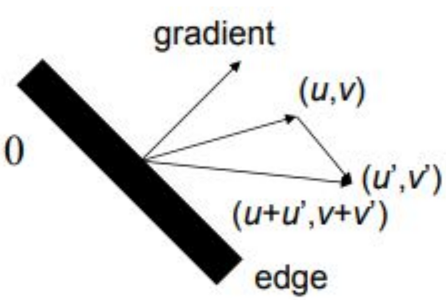
– **One equation, two unknowns (makes the problem under-constrained)**

What does this constra

$$\nabla I \cdot (u, v) + I_t = 0$$

The component of the flow perpendicular to the gradient (i.e., parallel to the edge) is

If (u, v) satisfies the equation, so does (u+u', v+v') if $\nabla I \cdot (u', v') = 0$

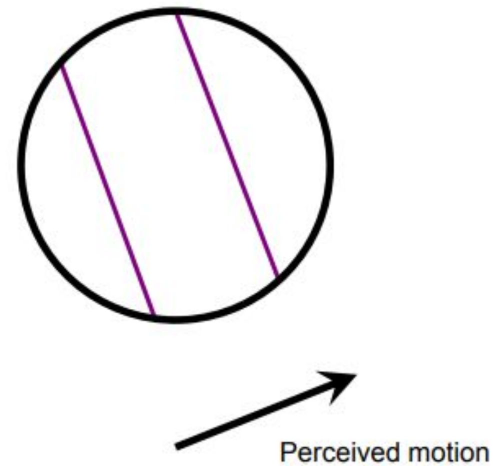


Optical Flow Equation Challenges

- The optical flow equation: $I_x u + I_y v + I_t = 0$
  - Gives us one equation per pixel
  - But we have two unknowns (u,v) per pixel
  - This makes the problem under-constrained
- Why is this a problem?
  - Cannot solve for unique flow vector
  - Multiple possible solutions exist
  - Need additional constraints
- This leads us to two key issues:
  - The Aperture Problem (coming up next)
  - Need for additional assumptions and constraints
- Solutions involve:
  - Looking at neighboring pixels
  - Making assumptions about motion
  - Using different mathematical approaches

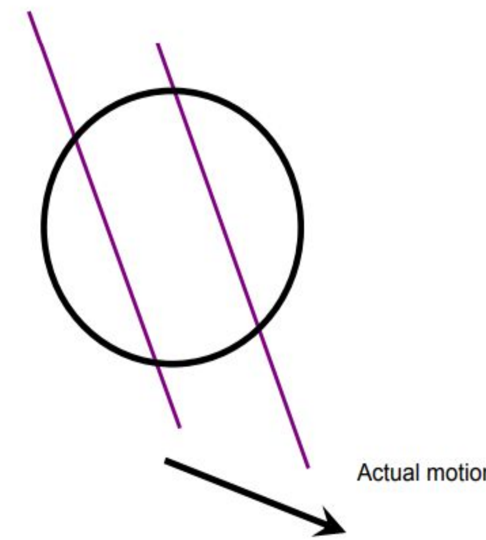
The aperture problem

- The aperture problem occurs when viewing motion through a limited window (aperture) like looking at a moving object through a small circular opening
- Circle represents our limited viewing window (aperture), purple lines represent an edge or feature we're tracking, black arrows show motion vectors



What We See vs Reality:

- **Actual Motion**: The true movement is diagonal/downward
- **Perceived Motion**: We only detect horizontal movement
  - Can only measure motion perpendicular to the edge
  - Component parallel to the edge is invisible



This is a fundamental challenge in optical flow computation

Solving the aperture problem

- How to get more equations for a pixel?
- Spatial coherence constraint:
  - pretend the pixel's neighbors have the same (u,v)
  - E.g., if we use a 5x5 window, that gives us 25 equations per pixel

$$\begin{bmatrix} I_x(\mathbf{x}_1) & I_y(\mathbf{x}_1) \\ I_x(\mathbf{x}_2) & I_y(\mathbf{x}_2) \\ \vdots & \vdots \\ I_x(\mathbf{x}_n) & I_y(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{x}_1) \\ I_t(\mathbf{x}_2) \\ \vdots \\ I_t(\mathbf{x}_n) \end{bmatrix}$$

Lucas-Kanade flow

- Linear least squares problem-

$$\begin{bmatrix} I_x(\mathbf{x}_1) & I_y(\mathbf{x}_1) \\ I_x(\mathbf{x}_2) & I_y(\mathbf{x}_2) \\ \vdots & \vdots \\ I_x(\mathbf{x}_n) & I_y(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{x}_1) \\ I_t(\mathbf{x}_2) \\ \vdots \\ I_t(\mathbf{x}_n) \end{bmatrix}$$

$$\underset{n\times 2}{\mathbf{A}} \; \underset{2\times 1}{\mathbf{d}} = \underset{n\times 1}{\mathbf{b}}$$

- Solution given by $(\mathbf{A}^T\mathbf{A})\mathbf{d} = \mathbf{A}^T\mathbf{b}$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

Methods of Optical Flow

- **Lucas-Kanade Method**:
  - Assumes flow is constant in a local neighborhood
  - Solves aperture problem by combining constraints from multiple pixels
  - Works well for small motions
  - Advantages: Robust to noise, computationally efficient
  - Limitations: Fails with large motions

- **Horn-Schunck Method**:
  - Global smoothness assumption
  - Minimizes distortion in flow and deviations from data
  - Creates denser flow fields
  - Advantages: Dense flow field, works well for transparent motion
  - Limitations: More sensitive to noise

Sparse vs. Dense Optical Flow

- **Sparse Optical Flow:**
  - Tracks specific key points (e.g., corners, edges).
  - Less computationally intensive.
  - Example: Monitoring selected points on a moving car.

- **Dense Optical Flow:**
  - Estimates motion for every pixel.
  - More detailed but computationally expensive.
  - Example: Visualizing entire flow fields in weather simulations.

Optical Flow Algorithm Steps

1. Pre-processing:
   - Image smoothing to reduce noise
   - Convert to grayscale if needed
2. Gradient Computation:
   - Calculate spatial gradients **(Ix, Iy)**
   - Calculate temporal gradient **(It)**
3. Flow Computation: For **Lucas-Kanade**:
   - Define window size
   - Compute local motion vectors
   - Solve least squares equations
4. For **Horn-Schunck**:
   - Initialize flow field
   - Iteratively update estimates
   - Apply smoothness constraint
5. Post-processing:
   - Filter outliers
   - Smooth flow field
   - **Visualize** results

Applications of Optical Flow

1. Traffic Analysis: Detecting vehicle speeds and trajectories.

2. Sports Analytics: Tracking player movements during a game.

3. Video Stabilization: Correcting shaky camera footage.

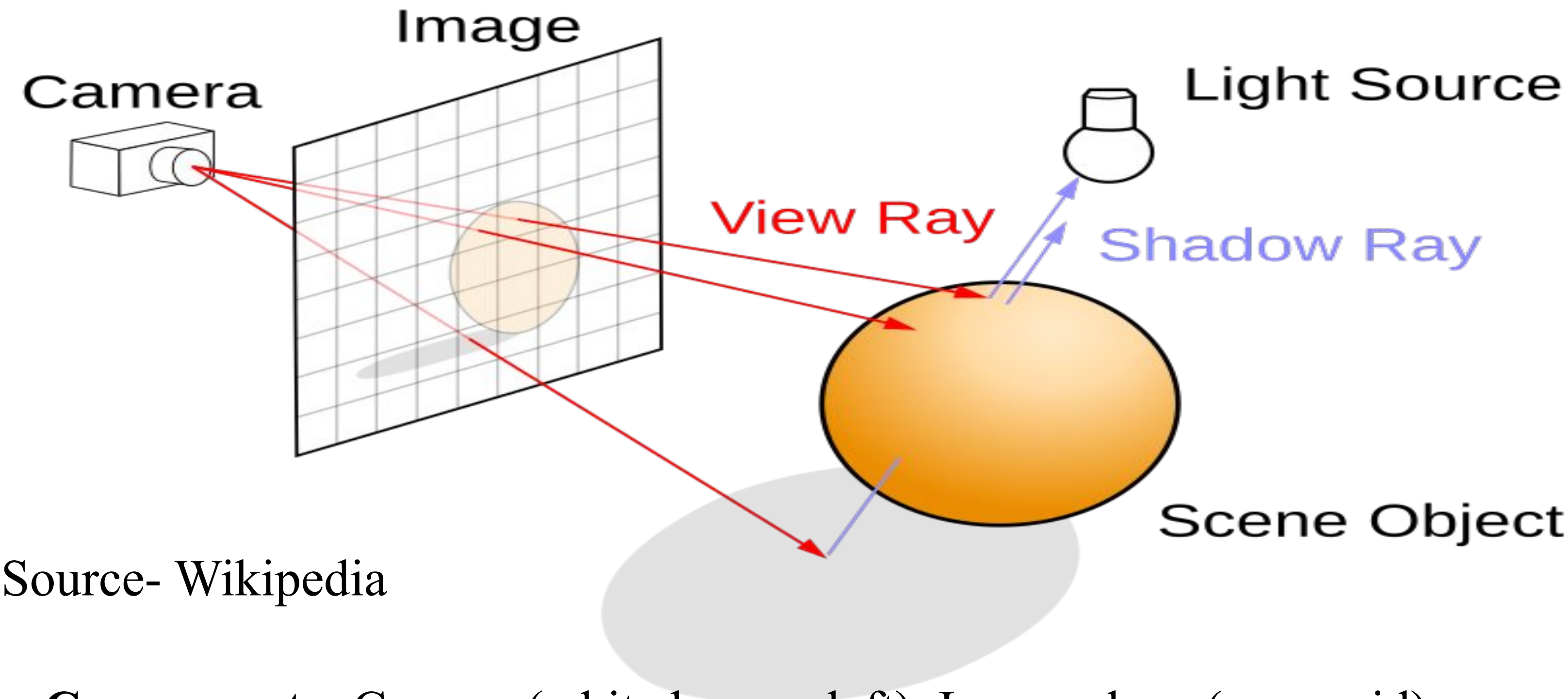
4. Action Recognition: Understanding human activities in video sequences.

Introduction to Ray Tracing

- Ray tracing is a technique for rendering three-dimensional graphics with very complex light interactions. This means you can create pictures full of mirrors, transparent surfaces, and shadows, with stunning results.
- A very simple method to both understand and implement.
- It is based on the idea that you can model reflection and refraction by recursively following the path that light takes as it bounces through an environment

Raytraced Images

Ray Tracing Model



Source- Wikipedia

Key Components: Camera (white box on left), Image plane (gray grid), Scene object (orange sphere), Light source (white bulb)

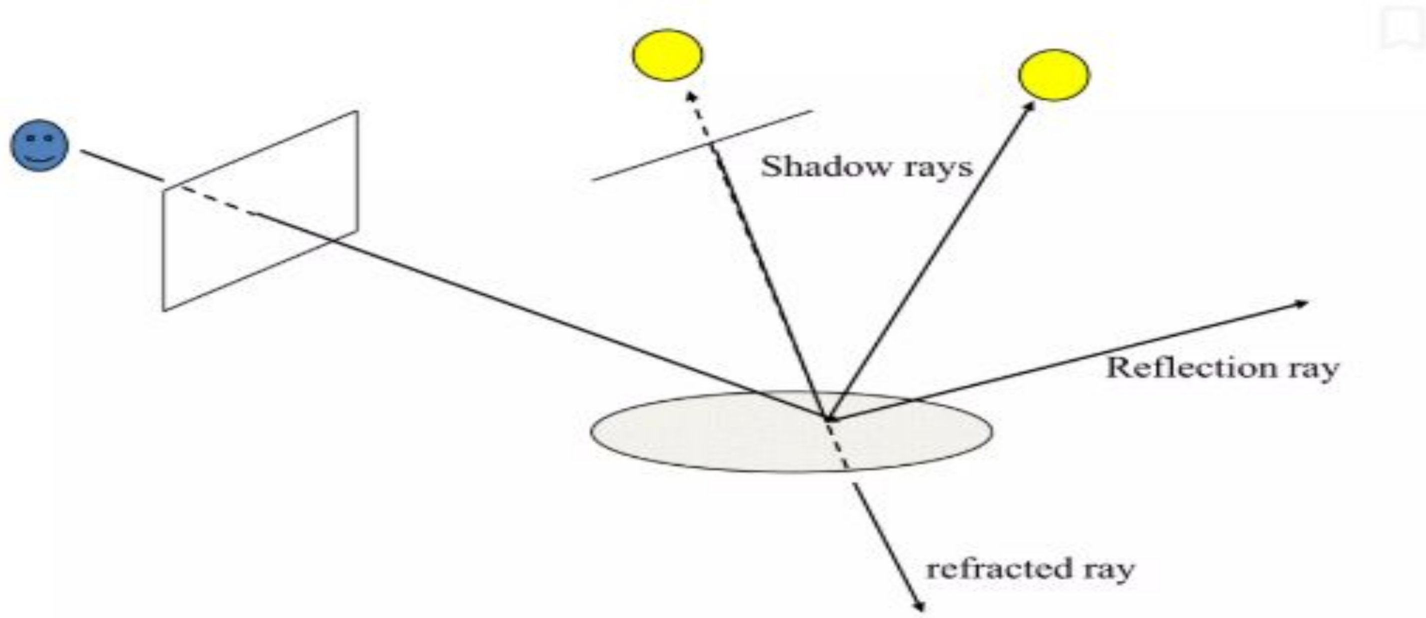
View Ray (Red):

- Originates from camera/eye
- Passes through image plane pixels
- Intersects with objects in scene

Shadow Ray (Blue):

- Cast from object intersection points
- Points toward light source
- Determines shadow calculations

Ray Tracing



Primary Ray (View Ray):

- Cast from camera through image plane
- Determines what camera sees
- First point of intersection with objects

Shadow Rays:

- Cast from intersection points to light sources
- Used to determine if point is illuminated
- Multiple rays for multiple light sources

Secondary Rays:

1. **Reflection Ray:**
   - Bounces off surface at equal angle
   - Creates mirror-like reflections
2. **Refraction Ray:**
   - Passes through transparent materials
   - Changes direction based on material properties
   - Creates glass/water effects

Ray Tracing Algorithm

- Builds the image pixel by pixel Cast additional rays from the hit point to determine the pixel color

- Shoot rays toward each light. If they hit something, the object is shadowed from that light, otherwise use "standard model" for the light

- Reflection rays for mirror surfaces, to see what should be reflected in the mirror

- Refraction rays to see what can be seen through transparent objects

- Sum all the contributions to get the pixel color

Ray Tracing Hardware Implementation

- Ray tracing has moved from offline rendering to real-time applications thanks to specialized hardware acceleration in modern GPUs and advanced software frameworks.

- Why Hardware Acceleration for Ray Tracing?
  - Performance Needs:
    - Ray tracing involves complex calculations (e.g., ray intersections, shading).
    - Real-time rendering demands efficient computation.

- Modern GPU Support
    - NVIDIA RTX Series (2000, 3000, 4000 series)
    - AMD Radeon RX 6000 & 7000 series
    - Intel Arc GPUs
    - Dedicated RT (Ray Tracing) cores

Ray Tracing: Applications and Usage

Ray Tracing in video games

Ray Tracing: Applications and Usage

Entertainment & Media

- **Video games:** Real-time graphics, shadows, reflections
- **Films:** Special effects and photorealistic animation
- **VR/AR:** Immersive visual experiences

Professional Applications

- **Architecture:** Building visualization and lighting design
- **Product Design:** Virtual prototyping and showcasing
- **Scientific:** Medical imaging and research visualization