Homework Spring 2026

Applied Computer Vision (CMU-Africa)

Release Date: Monday, January 20th, 2026
DUE: **Sunday Jan 1st, 2026, 11:59 PM CAT**

1. **Collaboration policy:**Discussion among students is allowed; however, the submitted work must be composed in your own words and should not be copied from any external sources. We recommend using latex for your report.

2. **Submitting your work:** Assignment will be submitted through Canvas. Please ensure that you include all the necessary files to reproduce your output. If you believe a specific environment is required to run your code, kindly submit a readme file detailing the procedures for setting up this environment. Submit a notebook and a report pdf file separately on canvas. Do not zip them!

3. **Getting Help**: Please use office hours and Piazza to ask any questions related to this assignment.

4. **Refrain from using GenAI**: You can use generative AI for learning assignment concepts but complete them yourself. We will strictly monitor the use of GenAI in your reporting.

Learning Objectives

By completing this assignment, students will be able to:

- Explain the image formation pipeline from a 3D scene to a digital image, including optical, geometric, and signal processing stages.
- Model perspective projection using the pinhole camera model and analytically project 3D points onto 2D image planes.
- Formulate and compute ray–plane intersections for both standard and tilted image planes.
- Analyze the impact of camera parameters such as focal length, image plane orientation, and camera pose on image projections.
- Map projected points to 2D image coordinates using camera intrinsic parameters.
- Simulate camera motion and render multiple views of a static 3D point cloud by varying camera extrinsics.
- Interpret perspective distortion and viewpoint changes in the context of graphics engines, simulators, and vision systems.
- Implement a basic image demosaicing algorithm using bilinear interpolation within an image signal processing pipeline.
- Evaluate the limitations of simple demosaicing methods and discuss potential improvements and alternative approaches.

Image Formation

In this section, we study the process by which a camera forms an image from a three-dimensional scene. Figure 1 illustrates a typical image sensing pipeline, from scene radiance to a digital image. In this assignment, we will simulate selected components of this pipeline. Part I focuses on the *optical image formation process*, modeling how a 3D scene is projected onto an image plane using a pinhole camera. In Part II, you will implement a basic simulation/gaming tool and in Part III you will implement a basic demosaicing algorithm as part of the Image Signal Processor (ISP). For additional background reading, refer to Chapter 2 of *Computer Vision: Algorithms and Applications, Second Edition* by Richard Szeliski.

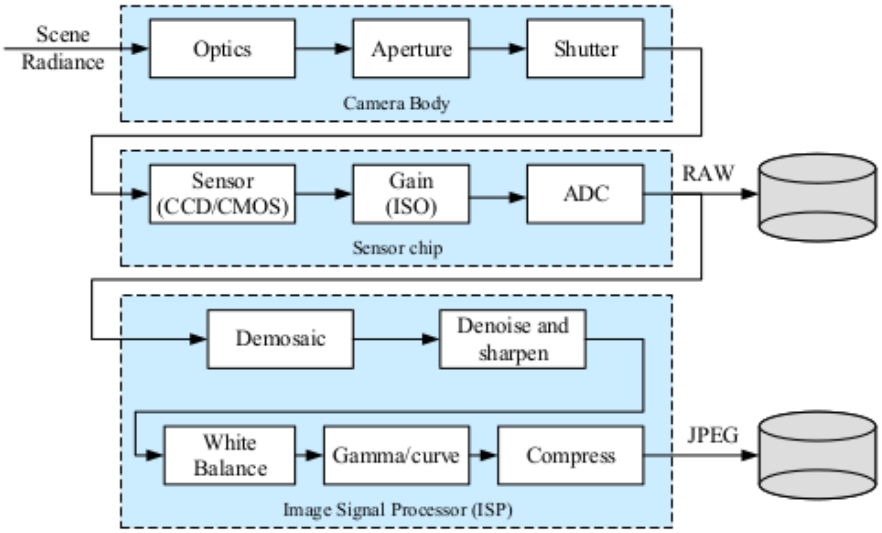

Figure 2.23 Image sensing pipeline, showing the various sources of noise as well as typical digital post-processing steps.

Figure 1: Image sensing pipeline (source: *Computer Vision: Algorithms and Applications*, R. Szeliski)

Part I: Pinhole Camera Model (50 Points)

A pinhole camera is a simple optical imaging device consisting of a light-tight enclosure with a small aperture (the pinhole) on one side and an image plane on the opposite side. Light rays from the scene pass through the pinhole and form an inverted image on the image plane. Despite its simplicity, the pinhole camera captures the essential geometry of perspective projection and serves as a fundamental model in computer vision.

To build intuition, we first consider an early optical device known as the Camera Obscura, which operates under the same geometric principles. Perspective projection in a pinhole camera is illustrated in Figure 2.

Using the pinhole camera model, we can analytically determine how points in the 3D world are projected onto a 2D image plane. In this assignment, we assume that for each point in the scene, a ray passes through the pinhole and intersects the image plane. The position of the light source is neglected.

For simplicity, objects in the scene are represented as points with coordinates (x, y, z). You are provided with a set of 3D points (blue and black points) stored in a NumPy pickle file.

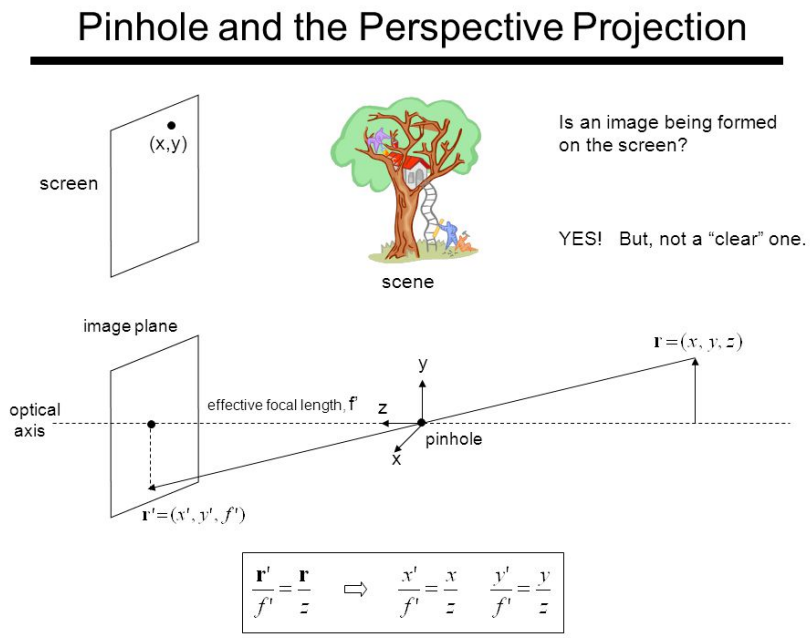

Figure 2: Pinhole camera and perspective projection

Your task is to project these points onto one or more image planes using the pinhole camera model.
You will experiment with different camera parameters, image planes, and focal lengths, and explore the use of homogeneous coordinates for a more general formulation.

1. **Step 1: Define the camera and image plane parameters**

   Choose the parameters of your pinhole camera:

   - Camera center $C = (C_x, C_y, C_z)$
   - Focal length f (distance from the camera center to the image plane)
   - Orientation of the optical axis (aligned with or deviating from the Z-axis)
   - Position and equation of the image plane

   Tasks:

   (a) Write down the equation of your image plane using:
   - A constant-depth plane: $Z = Z_0$
   - A general plane: $ax + by + cz + d = 0$

   (b) Experiment with at least two image planes:
   - A standard plane perpendicular to the Z-axis
   - A rotated or slanted plane

2. **Step 2: Trace rays from 3D points to the image plane**

   For a given 3D point $P = (X, Y, Z)$, consider the ray originating from the camera center C and passing through P.

   Tasks:

   (a) Derive the parametric equation of the ray

(b) Compute the intersection of this ray with the image plane

3. **Step 3: Project points onto the image plane**

   Using the ray equation and the image plane equation, compute the intersection point between the ray and the image plane.

   Tasks:

   (a) Compute the intersection point $I = (I_x, I_y, Z_0)$ (or the equivalent for a tilted plane)

   (b) Repeat this process for all 3D points and all defined image planes

4. **Step 4: Map projected points to 2D image coordinates**

   Map each intersection point on the image plane to 2D image coordinates (u, v) using:

   - Principal point (c_x, c_y)
   - Pixel scaling factors s_x and s_y

   Tasks:

   (a) Use the mapping equations:

   $$u = s_x(I_x - O_x) + c_x, \quad v = s_y(I_y - O_y) + c_y$$

   Assume $O_x = O_y = 0$ and $s_x = s_y = 1$.

   $$u = I_x + c_x$$
   $$v = I_y + c_y$$

   (b) Compute (u, v) for all projected points

Experimentation Tasks

Projection with a Standard Image Plane

Use $Z = Z_0$ as the image plane. Experiment with different focal lengths (e.g., $f = 10, 50, 100$). Plot the 2D projections of all points for each focal length.

Projection with a Tilted Image Plane

Define a tilted image plane of the form $ax + by + cz + d = 0$. Choose reasonable values for a, b, c, and d, compute the ray-plane intersections, and plot the resulting 2D projections.

Analysis of Parameters

- How does changing the focal length f affect the projected image?
- How do the position and orientation of the image plane influence the projection?
- Compare and discuss the differences between projections obtained from the two image planes.

Ideal Points

Compute the Euclidean (L2) distance between:

- The first black point and the first blue point
- The last black point and the last blue point

Repeat the same distance calculations using the corresponding points *after projection onto the image plane but before pixel coordinate mapping.*

Questions:

- What do you observe?
- Why are the distances preserved or altered after projection?

Part II: Rendering Views from a 3D Point Cloud (30 Points)

In Part I, you modeled a pinhole camera and projected isolated 3D points onto an image plane using perspective geometry. In this part, you will extend the same principles to a full 3D point cloud and generate multiple images by changing the camera pose. This experiment illustrates how images in video games and simulators are formed by moving a camera through a static 3D world.

You are provided with a `.ply` point cloud file representing a 3D scene. We captured this scene using a 3D Camera. Treat the point cloud as a fixed world coordinate system and simulate a moving pinhole camera that observes the scene from different viewpoints.

Objective

Using the pinhole camera model developed in Part I, generate 2D images of the point cloud from the following six canonical viewpoints:

- Front view
- Back view
- Left view
- Right view
- Top view
- Bottom view

You must not rotate the point cloud. All changes in the rendered images must result solely from changes in the camera position and orientation.

Step 5: Define Camera Poses

Define a camera pose for each viewpoint by specifying:

- Camera center $C = (C_x, C_y, C_z)$
- Camera orientation (rotation matrix R)

The camera coordinate transformation should follow:

$$P_{cam} = R(P_{world} - C)$$

Choose camera positions such that the entire point cloud is visible in each view.

Tasks:

(a) Define a consistent world coordinate system for the point cloud.

(b) Specify camera centers and orientations for all six views.

(c) Clearly state the viewing direction for each camera.

Step 6: Perspective Projection of the Point Cloud

For each camera pose:

- Transform all 3D points into the camera coordinate frame.
- Apply the pinhole projection equations:

$$x = f\frac{X_c}{Z_c}, \quad y = f\frac{Y_c}{Z_c}$$

- Discard points with $Z_c \leq 0$.

Tasks:

(a) Project all visible points onto the image plane.

(b) Map the projected points to 2D image coordinates (u, v).

Step 7: Image Formation

Create a 2D image for each viewpoint by plotting the projected points on an image grid.

Requirements:

- Use a fixed image resolution for all views.
- Ensure consistent camera intrinsics across views.
- Each pixel may represent one or more projected points.

Experimentation Tasks

Multi-View Rendering

Generate and display six images corresponding to the six camera viewpoints. Label each image clearly.

Camera Motion Interpretation

Explain how changing the camera pose alone produces different images of the same 3D scene.

Analysis Questions

- How does camera orientation affect the perceived structure of the scene?
- Why does perspective distortion vary across views?
- How does this experiment relate to camera navigation in video games or simulators?
- Use the ReVo 3D Camera (Contact TA's) to capture another scene and simulate it.

Optional Extension (20 Bonus Points)

- Color-code points based on depth.
- Animate a smooth camera orbit around the point cloud. (Combine Frames to create a video)
- Compare perspective projection with orthographic projection.

Part III (Image Demosaicing) (20 Points)

Image demosaicing is a crucial process that reconstructs a full-color image from incomplete color information captured by a digital camera's sensor, often using a color filter array (CFA). As image sensors do not directly measure full-color details at each pixel, demosaicing algorithms play a vital role in filling in missing color information. This enhances visual appeal, ensures accurate color representation in the final image, and is critical for overall image quality. The efficiency of demosaicing significantly impacts color accuracy, sharpness, and the reproduction of fine details in digital photographs.

In regions such as Africa, where the cost of obtaining high-quality images is prohibitive, efficient demosaicing algorithms become essential for improving the visual quality of images captured by low-cost cameras. You will implement a simple demosaicing algorithm as part of this process. You will be implementing a straightforward Bilinear Interpolation algorithm to fill the missing color information in a raw image. Please refer strictly to the instructions provided in the attached notebook, as we will not cover all the steps outlined in the referenced paper.

Sample images are provided for your reference; however, it's important to note that we will assess your implementation using other images not included in this assignment. Ensure that your algorithm generalizes well to various images beyond the provided samples.

In your report, consider discussing potential improvements to the demosaicing process. Explore how this process could be enhanced and highlight the limitations of the implemented algorithm. Additionally, provide insights into other demosaicing algorithms, including their advantages and limitations.

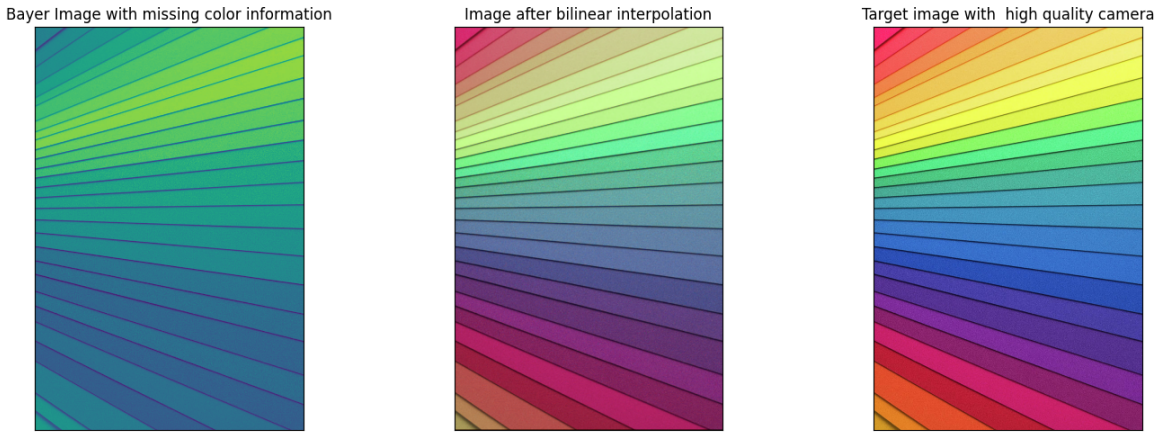

Figure 3: Image Demosaicing Results