Computer Vision and Multiview Geometry
DT084A Lab - 1 Report

Student: Kaan Tekin Öztekin

Date of submission: 26.01.2026

Task 1: Derivation of the Camera Field of View

Figure 1 illustrates the pinhole camera model, where light rays from a 3D object pass through a single focal point and are projected onto the image plane. The distance between the pinhole and the image plane defines the focal length, which directly determines the scale and field of view of the projected image.

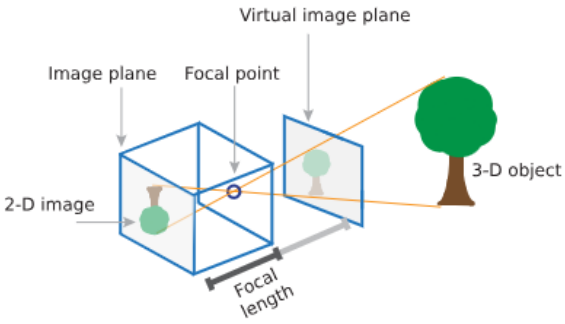

Figure 1: Pinhole camera model illustrating perspective projection from a 3D scene onto a 2D image plane.

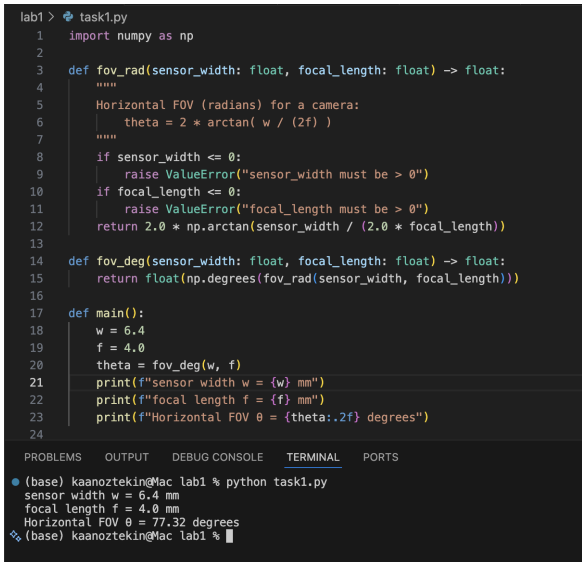

Figure 2: In the implementation, this formula is used to compute the field of view numerically.

Considering the image sensor is placed at a distance f from the pinhole, and half of the sensor width is given by $w/2$. The half field of view angle $\theta/2$ can be obtained using basic trigonometry:

$$\tan\left(\frac{\theta}{2}\right) = \frac{w/2}{f} \tag{1}$$

Solving for θ, the horizontal field of view is given by:

$$\theta = 2\arctan\left(\frac{w}{2f}\right) \tag{2}$$

This equation shows that the field of view increases as the sensor width increases or the focal length decreases. The resulting angle θ is expressed in radians.

Task 2: FOV vs Focal Length for Two Different Sensor

In this task, I compared two cameras with different sensor widths (w_1 and w_2) while varying the focal length f. Using the pinhole camera model derived in Task 1, the horizontal field of view is:

$$\theta(f, w) = 2 \arctan\left(\frac{w}{2f}\right) \tag{3}$$

For each camera, I compute θ over a range of focal lengths $f \in [1.5, 80]$ mm and plot the resulting curves in a single graph (Figure 3). As expected, the field of view decreases as focal length increases. Additionally, for the same focal length, the camera with the larger sensor width (w_2) produces a larger field of view than the smaller sensor (w_1).

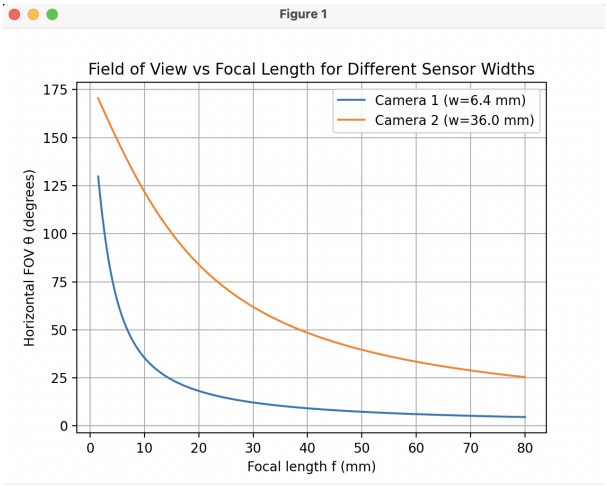

Figure 3: Horizontal field of view θ as a function of focal length f for two different sensor widths. A larger sensor width results in a wider field of view for the same focal length.

Task 3: Effect of focal length and depth on projected point distance

Considering two 3D world points:

$$\mathbf{x}_1 = (x, y, z), \quad \mathbf{x}_2 = (x + \Delta x, y, z),$$

and their projections on the image/sensor plane $\mathbf{x}_1', \mathbf{x}_2'$. Using the pinhole camera model (assuming the principal point is at the origin for simplicity), the projection of the x-coordinate is:

$$x' = f\frac{x}{z},$$

where f is the focal length and z is the depth (distance to the camera along the optical axis). The distance between the projected points along the image x-axis becomes:

$$|x_2' - x_1'| = \left| f\frac{x + \Delta x}{z} - f\frac{x}{z} \right| = \left| f\frac{\Delta x}{z} \right|.$$

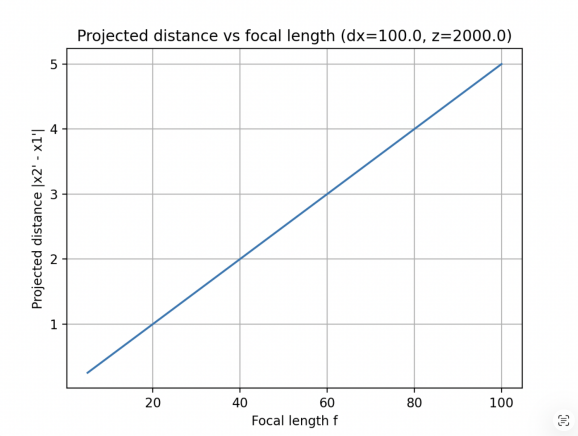

Figure 4: Projected distance $|x'_2 - x'_1|$ versus focal length f (fixed Δx and z). The relationship is linear.

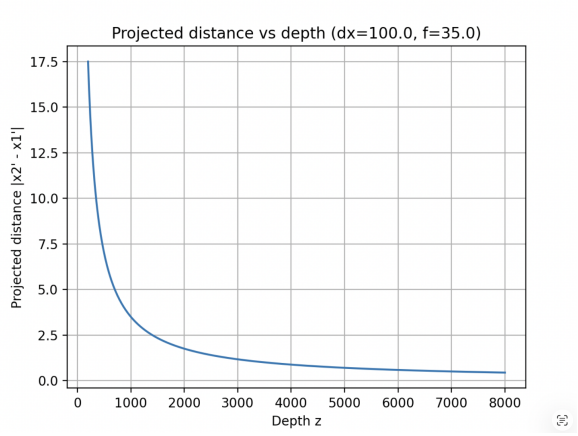

Figure 5: Projected distance $|x'_2 - x'_1|$ versus depth z (fixed Δx and f). The relationship follows an inverse decay $\sim 1/z$.

- **Dependence on focal length f:** For fixed Δx and z, the projected distance increases *linearly* with f. This corresponds to a zoom-in effect: a larger focal length magnifies the image, making the same real-world separation appear larger on the sensor.
- **Dependence on depth z:** For fixed Δx and f, the projected distance decreases as $\frac{1}{z}$. This is the core reason why *depth differences become less noticeable at larger distances*: when z grows, changes in $1/z$ become smaller, so two points with the same real-world separation appear closer together in the image.

Figure 4 shows that $|x'_2 - x'_1|$ increases approximately as a straight line when f increases (since $|x'_2 - x'_1| = f\Delta x/z$). Figure 5 shows a decaying curve versus z, matching the inverse relationship $|x'_2 - x'_1| \sim 1/z$. The curve drops quickly for small depths and flattens out for large depths, meaning that far-away objects exhibit much smaller changes in projected spacing.

Task 4: Depth Estimation with OpenCV

In this task, I expanded the provided OpenCV boilerplate by implementing a real-time depth estimation pipeline. The goal is to infer scene depth from camera input. Estimating depth from a single monocular camera is inherently ambiguous, since many different 3D scenes can produce similar 2D projections. Therefore, monocular depth estimation is typically addressed using learning-based approaches (e.g., CNN/Transformer models such as MiDaS and MonoDepth2), which learn depth priors from large datasets.

In Task 4, a real-time depth-related visualization is implemented using a single monocular camera. The resulting output does not represent metric depth, but rather a relative depth appearance, where intensity variations are mapped using colormaps to highlight closer and farther regions. Since the system is not calibrated (no baseline and intrinsics), I visualize a *relative depth* by taking the inverse of disparity:

$$D_{\text{rel}} \propto \frac{1}{\text{disparity}} \tag{4}$$

The implementation uses the following OpenCV components:

- **Video I/O:** `cv2.VideoCapture`, `cap.read()`, and camera resolution settings.
- **Image processing:** `cv2.cvtColor` to convert frames to grayscale.
- **Stereo matching:** `cv2.StereoSGBM_create` and `stereo.compute` to obtain disparity.
- **GUI controls:** `cv2.namedWindow` and `cv2.createTrackbar` to interactively tune parameters such as `numDisparities`, `blockSize`, and optional smoothing.
- **Filtering (optional):** `cv2.medianBlur` to reduce speckle noise in the disparity visualization.
- **Visualization:** `cv2.normalize` to map values to $[0, 255]$ and `cv2.applyColorMap` for false-color display.

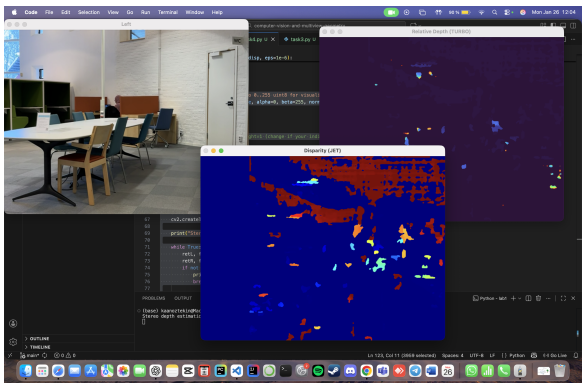

Figure 6: *
Default settings with a red rectangular label on camera

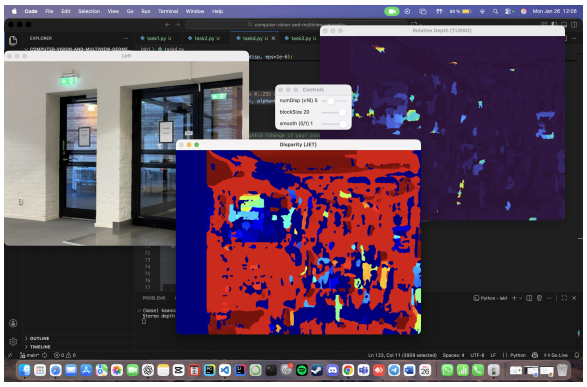

Figure 7: *
Increased amount of box size for better depth-scene understanding

To make the depth/disparity maps easier to interpret, I used two different colormaps:

- `COLORMAP_JET` and `COLORMAP_TURBO` for visualizations.

References

- Pinhole Camera Model overview. `https://medium.com/@nrmin.mrdova.01/pinhole-camera-84`
- OpenCV Python Tutorials. `https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html`
- OpenCV Depth Map and Stereo Vision Tutorial. `https://docs.opencv.org/4.x/dd/d53/tutorial_py_depthmap.html`
- OpenCV Colormap Documentation. `https://docs.opencv.org/4.x/d3/d50/group__imgproc__colormap.html`
- Breckon, T. Stereo SGBM Python example. GitHub repository. `https://github.com/tobybreckon/python-examples-cv/blob/master/stereo_sgbm.py`