1

The University of Calgary
Department of Geomatics Engineering
ENGO 634: Principles of Photogrammetry

Stereo Viewing and Metric Information from Vertical Imagery

Lab Teaching Assistant:
Armin Ghayur Sadigh

Assignment date:
28/01/2025

Submission date:
11/02/2025

Group Members' Names and addresses

Fre Ashal 30231900
(fre.ashal@ucalgary.ca)
Okoye Akachukwu 30198304
akachukwu.okoye1@ucalgary.ca
Chrisantus Innoccent 30240521
chrisantus.innocent@ucalgary.ca
Quinn Ledingham 30092066
(quinn.ledingham@ucalgary.ca)

Winter 2025

ENGO 634: Principles of Photogrammetry
Lab–1: Stereo viewing and metric information from vertical imagery

Objectives

- To practice stereo viewing using mirror and pocket stereoscopes
- To derive metric information from a single vertical image including scale, flying height and the height of a building.

Problem 1a:

Align the provided stereo pair of aerial photographs until you see in 3D through the mirror stereoscope. Describe the steps you followed.

The steps to use the mirror stereoscope are as follows:

1. Set up workspace.
2. Proper positioning of the stereo pair, placing it on a flat surface at a comfortable viewing distance.
3. Adjustment of the stereoscope mirror, sliding the two lenses closer or farther apart until both images are aligned and appear in focus.
4. Alignment of the image for the 3D view, we achieved this by moving the stereo pair until the image appears in 3D.
5. Proper observation of the terrain and record the observations on depth perception.
6. Record observations.

Problem 1b:

View the following stereo pair using a pocket stereoscope. Order the circles based on the elevation of the outer rings. Start with the highest ring.

7, 6, 5, 1, 4, 2, 3, 8

View the following stereogram using a pocket stereoscope. What "question & answer" can you visualize?

What Danial did first
sent an email

Problem 1c:

View the following stereo pair using a pocket stereoscope. What is the geometric primitive that can be seen?

Square

Problem 2a:

Fiducial mark measurement and refinement.

The fiducial marks on the aerial image were measured to establish a reference coordinate system. These marks are critical for scaling and aligning the image. We measured the coordinates using image viewing software and converted them from left-handed to right-handed coordinates. The pixel spacing was derived by comparing the distances between fiducial marks in both pixel and calibration coordinate systems. The fiducial center was calculated by intersecting the lines joining the corner fiducial marks.

Table 1 below describes the pixel coordinates of the fiducial marks.

Table 1: Coordinates of fiducial marks in left-handed image coordinate system

Point ID	X (pixels)	Y (pixels)
1	1346	19285
2	19170	1478
3	1353	1472
4	19163	19291
5	846	10378
6	19670	10385
7	10262	971
8	10254	19792

Table 2 shows the fiducial marks after they were converted to the right-handed image coordinate system. With Image Dim = (20456, 20488) this was calculated using:

$$X = table1\_X - (20456/2)$$
$$Y = (20488/2) - table1\_Y$$

Table 2: Coordinates of fiducial marks in right-handed image coordinate system

Point ID	X (pixels)	Y (pixels)
1	-8882	-9041
2	8942	8766
3	-8875	8772

4	8935	-9047
5	-9382	-134
6	9442	-141
7	34	9273
8	26	-9548

Table 3 shows the values used to calculate the pixel spacing. The pixel spacing was calculated by comparing the distances between fiducial marks in the image (in pixels) with their known calibration distances (in mm). The mean pixel spacing was derived from the two diagonals (1-2 and 3-4) and the two mid-side distances (5-6 and 7-8).

$$Distance = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

$$Pixel\ Spacing\ Estimate = \frac{Calibration\ Distance}{Measured\ Distance}$$

Table 3: Measured distances and pixel spacing estimate

	Measured Distance (pixels)	**Calibration Distance (mm)**	**Pixel Spacing Estimate (mm / pixels)**
1-2	25194.295	299.816	0.0118999
3-4	25193.508	299.825	0.0119009
5-6	18824.001	224.005	0.01189997
7-8	18821.002	224.006	0.0119019

Mean Pixel Spacing Estimate = 0.0119019 mm / pixels

Table 4 shows the coordinates after they were multiplied by the mean pixel spacing estimate.

Table 4: Scaled fiducial coordinates

Point ID	**X (mm)**	**Y (mm)**
1	-105.702	-107.594
2	106.416	104.321
3	-105.618	104.393
4	106.332	-107.665
5	-111.652	-1.595
6	112.366	-1.678
7	0.405	110.355
8	0.309	-113.627

The fiducial center was calculated by intersecting the lines joining the corner fiducial marks. The equations of the lines were parameterized as y = mx + b, and the intersection point was determined.

Lines in Slope Intercept Form
y = 0.9990462298025133x + -1.993019386907207
y = -1.0005053340819763x + -1.2791401196751622

Solving for x and y:
Fiducial Center = 0.357, -1.636

Table 5 is showing the points after being reduced to the fiducial center.

X = table4_X – 0.357
Y = table4_Y – (-1.636)

Table 5: Reduced fiducial coordinates to fiducial centre

Point ID	X (mm)	Y (mm)
1	-106.059	-105.957
2	106.059	105.957
3	-105.975	106.029
4	105.975	-106.029
5	-112.009	0.042
6	112.009	-0.042
7	0.048	111.991
8	-0.048	-111.991

Principal Point Offset = (-0.006, 0.006) from Calibration Certificate

X = table5_X – (-0.006)
Y = table5_Y – 0.006

Table 6: Reduced fiducial coordinates to Principal Point (PP)

Point ID	X (mm)	Y (mm)
1	-106.053	-105.963
2	106.065	105.951
3	-105.969	106.023
4	105.981	-106.035
5	-112.003	0.036
6	112.015	-0.048
7	0.054	111.985
8	-0.042	-111.997

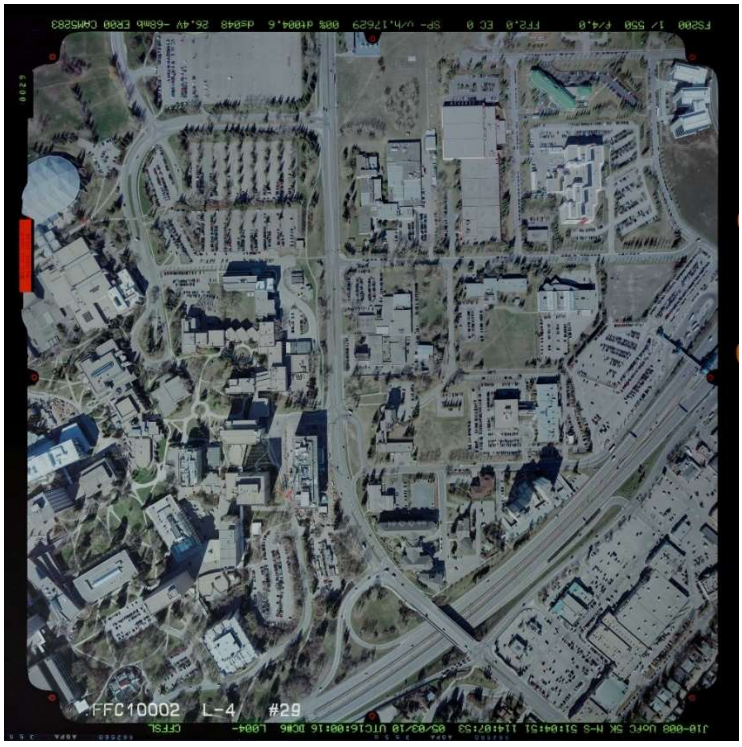

Figure 1: Image provided for lab 1 (dec_1_029.jpg)

Problem 2b:

Flying height estimation.

Table 7 shows the pixel coordinates measured for each of the ground control points.

Table 7: Raw ground control point measurements

Control Point ID	X (pixels)	Y (pixels)
300	4544	17489
301	4167	13362
303	3937	3624
304	2547	5778

Table 8 shows the ground control point coordinates after they were converted to right-hand coordinates and scaled like the measurements in Problem 2a were.

Table 8: Right-handed and scaled ground control point coordinates

Control Point ID	X (mm)	Y (mm)
300	-67.643	-86.220
301	-72.130	-37.106
303	-74.867	78.782
304	-91.409	53.148

Table 9 shows the ground control point coordinates after they were reduced to both the fiducial center and the principal point coordinates from Problem 2a in the same way the coordinates were reduced in Problem 2a.

Table 9: GCP coordinates reduced to fiducial center and PP coordinates

Control Point ID	**X (mm)**	**Y (mm)**
300	-67.994	-84.590
301	-72.481	-35.476
303	-75.218	80.413
304	-91.760	54.779

Table 10: RLG control point coordinates

Control Point ID	X (m)	Y (m)	Z (m)
300	497.49	-46.90	1090.56
301	258.21	-63.79	1092.84
303	-311.55	-66.89	1094.69
304	-186.74	-151.54	1093.12

Distances calculated using control points 300 and 304 because they have the biggest separation between them in the x direction and large difference in the y direction, reducing sensitivity to small pixel inaccuracies in both directions.

$$Image\ Distance = \sqrt{(-67.994 - (-91.760))^2 + (-84.590 - 54.779)^2} = 0.14138m$$

$$Object\ Distance = \sqrt{(497.49 - (-311.55))^2 + (-46.90 - (-66.89))^2 + (1090.56 - 1094.69)^2} = 692.18984m$$

$$Scale = \frac{692.18984}{0.14138} = 4895.94055$$

Calibrated Focal Length = 153.358mm from Calibration Certificate

Flying Height Estimate = 4895.94055 * 0.153358 = 750.83165m

Problem 2c:

Building height estimation.

Tables 11 – 14 show the pixel coordinate measurements being converted into right-handed coordinates, scaled, and reduced to fiducial center and principal point in the same manner done in the previous two problems.

Table 11: Top and bottom image measurements of ICT building

Point	X (pixels)	Y (pixels)
Top	6148	11608
Bottom	6346	11558

In Figure 2 the top left corner pixel of each red square is the pixel that was measured.

Figure 2: Image of points measured on the ICT building

Table 12: Right-handed and scaled building measurements

Point	X (mm)	Y (mm)
Top	-48.555	-16.232
Bottom	-46.198	-15.637

Table 13: Reduced to fiducial center building measurements

Point	X (mm)	Y (mm)
Top	-48.912	-14.596
Bottom	-46.555	-14.001

Table 14: Reduced to principal point building measurements

Point	X (mm)	Y (mm)
Top	-48.906	-14.602
Bottom	-46.198	-14.007

$$Radial\ distance\ top = \sqrt{-48.906^2 + -14.602^2} = 51.03910mm$$
$$Radial\ distance\ bottom = \sqrt{-46.198^2 + -14.007^2} = 48.61114mm$$

$$Estimated\ building\ height = \frac{750831.65 * (51.03910 - 48.611114)}{51.03910} = 35717.54mm$$

Estimated building height = 35.717m

The surveyed building height is 35.2m so the estimated building height is wrong by less than a meter. This difference could be caused by difficulties determining the bottom of the building because of the shadow or slight errors in the measurement of ground control points.

Appendix A: Code

calc.py

```
import math

def dist(a, b):
    return math.sqrt((a[0] - b[0])**2 + (a[1] - b[1])**2)

def dist_3D(a, b):
    return math.sqrt((a[0] - b[0])**2 + (a[1] - b[1])**2 + (a[2] - b[2])**2)

def slope(p1, p2):
    return (p2[1] - p1[1]) / (p2[0] - p1[0])

class Line:
    def __init__(self, slope, point):
        self.m = slope
        self.b = -slope * point[0] + point[1]

    def output(self):
        print(f"y = {self.m}x + {self.b}")

    @staticmethod
    def intersection(line_a, line_b):
        x = (line_b.b - line_a.b) / (line_a.m - line_b.m)
        y = (line_b.m * line_a.b - line_a.m * line_b.b) / (line_b.m - line_a.m)
        return [x, y]

def right_hand(a, image_dim):
    output = []
    for coords in a:
        x = coords[0] - (image_dim[0]/2)
        y = (image_dim[1]/2) - coords[1]
        output.append([x, y])
    return output

def scale(a, scale):
    for coords in a:
        coords[0] = scale * coords[0]
        coords[1] = scale * coords[1]

def reduce_fc(a, f_c):
    for coords in a:
        coords[0] = (coords[0] - f_c[0])
        coords[1] = (coords[1] - f_c[1])

def reduce_ppo(a, ppo):
    for coords in a:
        coords[0] = (coords[0] - ppo[0])
        coords[1] = (coords[1] - ppo[1])

def reduce(a, f_c, ppo):
    reduce_fc(a, f_c)
    print(f"reduced fc:\n{a}")
    reduce_ppo(a, ppo)
    print(f"reduced ppo:\n{a}")

def pixel_to_image(a, image_dim, pixel_spacing, fiducial_center, principal_point_offset):
    print(f"og:\n{a}")
    a = right_hand(a, image_dim)
    print(f"right-handed:\n{a}")
```

```
    scale(a, pixel_spacing)
    print(f"scaled:\n{a}")
    reduce(a, fiducial_center, principal_point_offset)

    return a

# part a: Fiducial mark measurement and refinement
print("\npart a:\n")
cali_dists = [
    299.816,
    299.825,
    224.005,
    224.006,
] # mm

fiducial_pixels = [
    [1346, 19285],
    [19170, 1478],
    [1353, 1472],
    [19163, 19291],
    [846, 10378],
    [19670, 10385],
    [10262, 971],
    [10254, 19792],
]

image_dim = [20456, 20488]

# calculate mm / px aka pixel spacing
print("distances:")
pixel_spacing = 0
for i in range(4):
    cali_dist = cali_dists[i]
    it = i * 2
    pix_dist = dist(fiducial_pixels[it + 1], fiducial_pixels[it])
    print(f"measured: {pix_dist}")
    pixel_spacing_estimate = cali_dist / pix_dist
    print(f"spacing: {pixel_spacing_estimate}")
    pixel_spacing += pixel_spacing_estimate
pixel_spacing = pixel_spacing / 4 # mm / px

print(f"left-handed fiducial:\n{fiducial_pixels}")

fiducial = right_hand(fiducial_pixels, image_dim)
print(f"right-hand:\n{fiducial}")
scale(fiducial, pixel_spacing)
print(f"scaled:\n{fiducial}")

m1_2 = slope(fiducial[1], fiducial[0])
m3_4 = slope(fiducial[3], fiducial[2])

line_1 = Line(m1_2, fiducial[0])
line_2 = Line(m3_4, fiducial[2])

print("Line 1:")
line_1.output()
print("Line 2:")
line_2.output()

fiducial_center = Line.intersection(line_1, line_2)
print(f"Fiducial Center: {fiducial_center}")

ppo = [-0.006, 0.006] # mm

reduce(fiducial, fiducial_center, ppo)
```

```
# part b: Flying height estimation
print("\npart b:\n")

cp = [
    [4544, 17489], # 300, px
    [4167, 13362], # 301, px
    [3937, 3624],  # 303, px
    [2547, 5778],  # 304, px
]

rlg_cp = [
    [497.49, -46.90, 1090.56],   # 300, m
    [258.21, -63.79, 1092.84],   # 301, m
    [-311.55, -66.89, 1094.69],  # 303, m
    [-186.74, -151.54, 1093.12]  # 304, m
]

cp = pixel_to_image(cp, image_dim, pixel_spacing, fiducial_center, ppo)
cfl = 153.358 # mm

cp_dist = dist(cp[0], cp[3]) * 1E-3   # m
rlg_dist = dist_3D(rlg_cp[0], rlg_cp[3]) # m
print(f"Image Distance {cp_dist}, Object Distance {rlg_dist}")

S = rlg_dist / cp_dist
print(f"Scale: {S}")

flying_height = S * (cfl * 1E-3) # m
print(f"Flying Height: {flying_height}")

# part c: Building height estimation
print("\npart c:\n")

ict = [
    [6148, 11608], # Top, px
    [6346, 11558]  # Bottom, px
]

ict = pixel_to_image(ict, image_dim, pixel_spacing, fiducial_center, ppo)

rt = dist(ict[0], [0, 0])
rb = dist(ict[1], [0, 0])

print(f"rt: {rt}")
print(f"rb: {rb}")
h = ((flying_height * 1E3) * (rt - rb)) / rt # mm

print(f"ICT Height: {h * 1E-3}")
```