Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №8

з дисципліни «Технології Computer Vision»

на тему

«ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ТРИВИМІРНОЇ РЕКОНСТРУКЦІЇ ОБ'ЄКТІВ ЗА ЦИФРОВИМИ ЗОБРАЖЕННЯМИ»

Виконала:
студентка групи ІС-12
Павлова Софія

Перевірив:
Баран Д. Р.

Київ – 2024

1. Постановка задачі

Мета роботи:

Дослідити методологію і технології реконструкції 3D просторових об’єктів за їх 2D зображеннями методами багатовидової (стерео / сигнатурна) обробки.

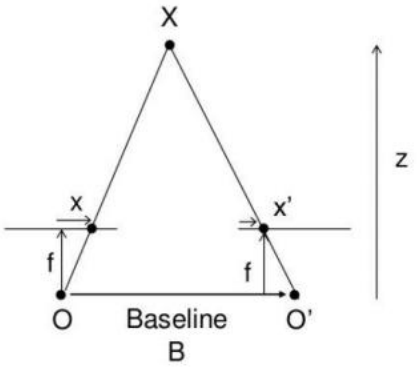
Завдання І рівня:

Обрати із відкритих джерел результати роботи стереопари. Здійснити 3D реконструкцію обраного об’єкту та дослідити якість результату від параметрів стереопари: база, ракурс на об’єкт (за умов наявності стереопари, або відомих параметрів, що супроводжують відкриті джерела даних від стереопари).

2. Виконання

2.1. 3D-реконструкція за допомогою стереопари

Використовує два зображення, отриманих за допомогою стерео-камери для побудови образу 3D об'єкта шляхом вибору точок відповідності, їх зіставлення та виконання геометричних перетворень.



Алгоритм:

1. Визначення растрових матриць зображень.
2. Визначення карти глибин зображень.
3. Формування просторового зображення.

Математична модель глибини сприйняття зображення

Щоб визначити глибину X у синтезованому 3D зображенні, визначаються координати двох точок O і O' у лівому та правому 2D зображеннях.

Глибина z 3D зображення визначається за допомогою теореми Піфагора: $z = \sqrt[2]{x^2 - B/2}$.

Алгоритм програми

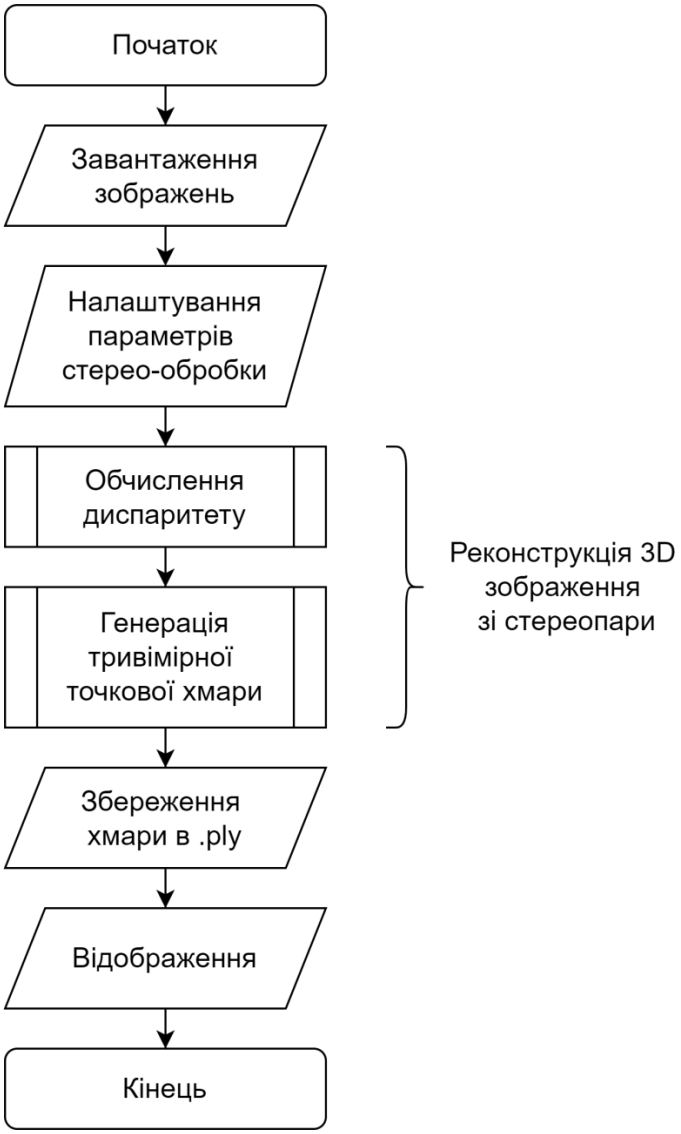


Рисунок 1 – Блок схема програми

Стереопара

Візьмемо стереопару, знайдену в інтернеті на сайті flickr.

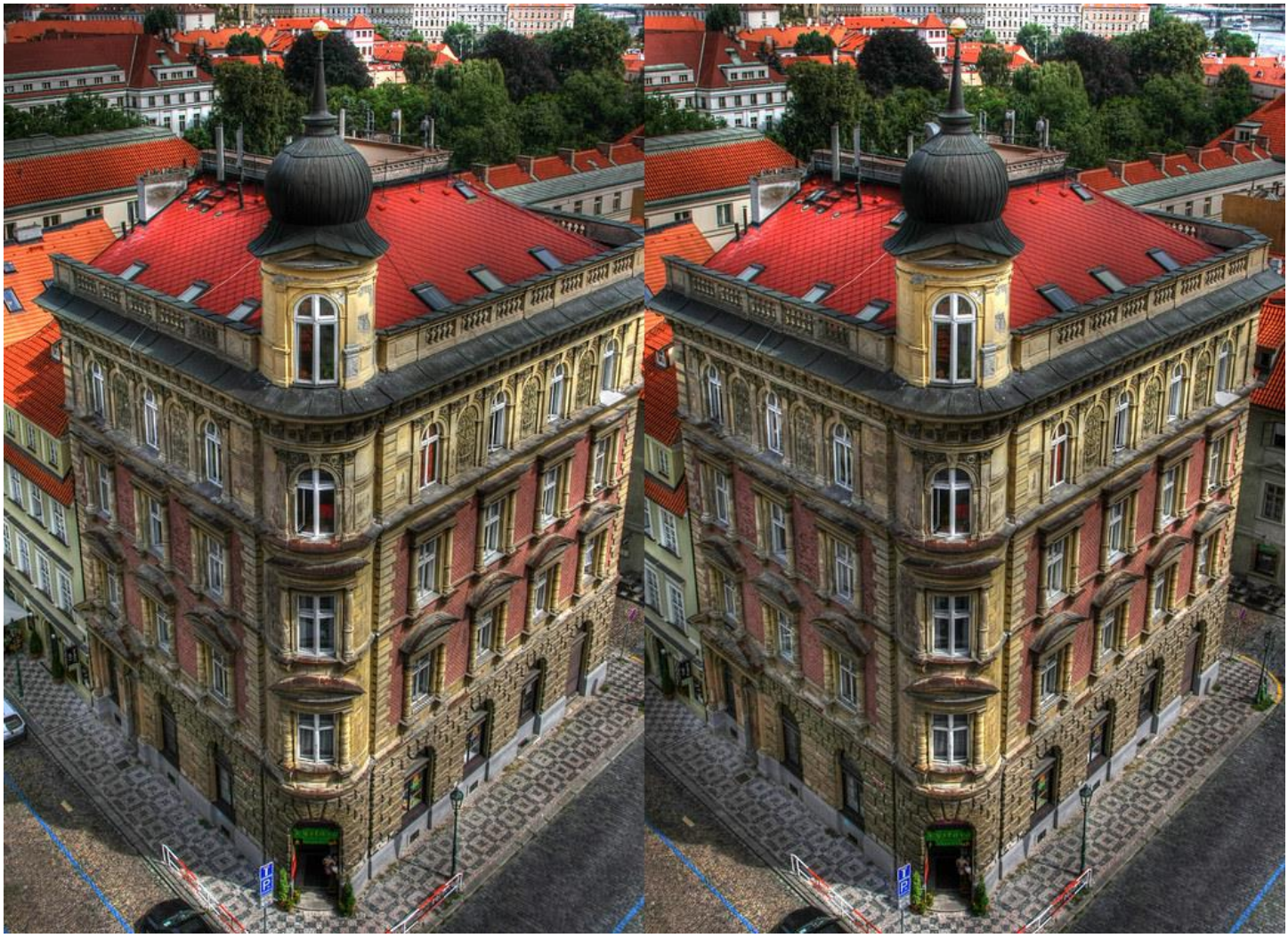


Рисунок 2 – Стереопара

Розділимо її на 2 зображення (ліве та праве) за допомогою онлайн інструменту imgonline.

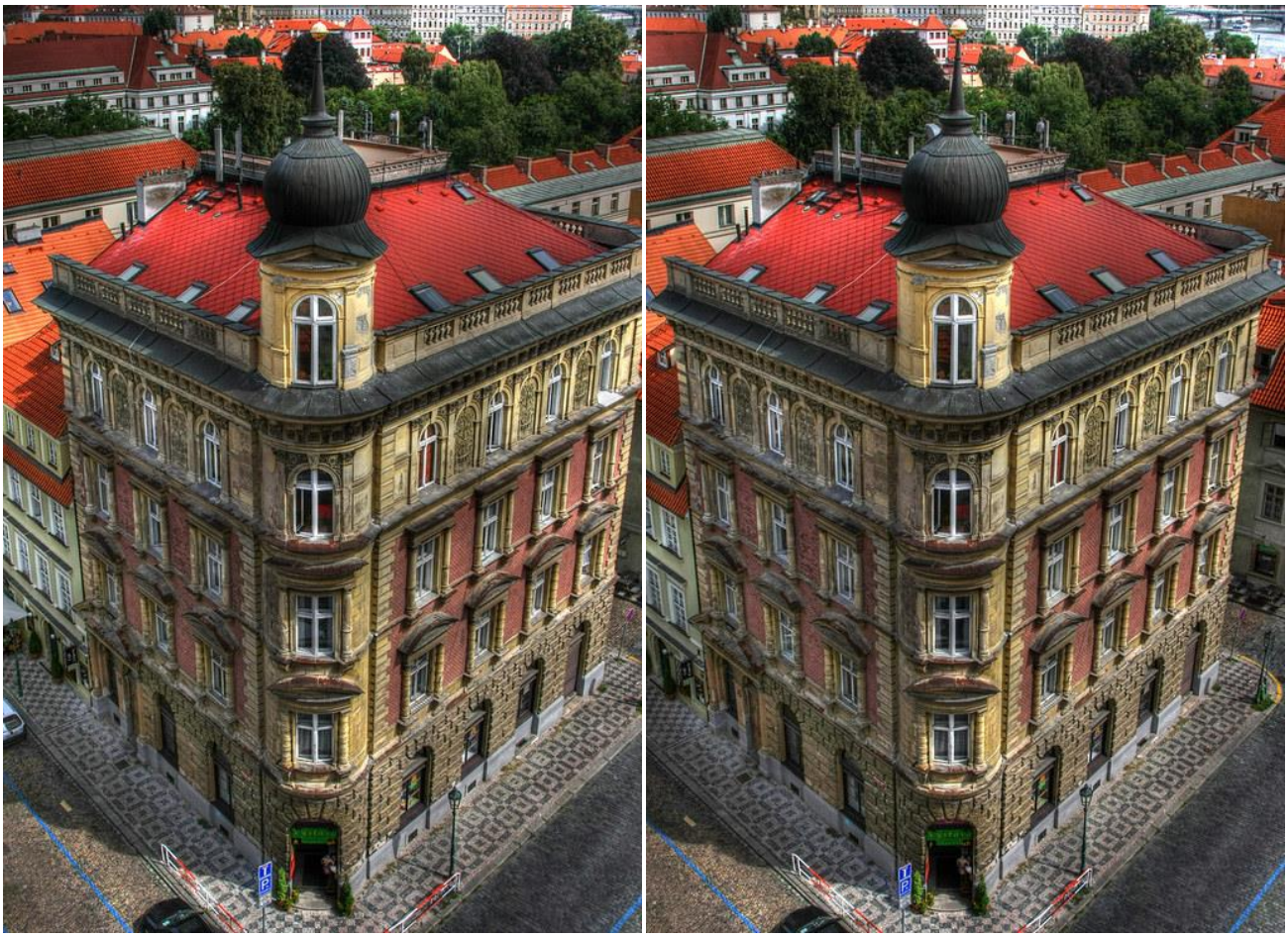


Рисунок 3 – Ліве та Праве зображення зі стереопари

Програмна реалізація

Напишемо програмний скрипт, який буде за заданими вручну параметрами стерео-обробки будувати тривимірну точкову хмару (наш 3D об’єкт) із двох зображень стереопари.

Скрипт взято з github opencv і дещо модифіковано під поточне завдання.

Лістинг коду:

```
from __future__ import print_function
import numpy as np
import cv2
import open3d as o3d

ply_header = '''ply
format ascii 1.0
element vertex %(vert_num)d
property float x
property float y
```

```
property float z
property uchar red
property uchar green
property uchar blue
end_header
'''

def write_ply(fn, verts, colors):
    verts = verts.reshape(-1, 3)
    colors = colors.reshape(-1, 3)
    verts = np.hstack([verts, colors])
    with open(fn, 'wb') as f:
        f.write((ply_header % dict(vert_num=len(verts))).encode('utf-8'))
        np.savetxt(f, verts, fmt='%f %f %f %d %d %d ')


if __name__ == '__main__':
    print('loading images...')
    # Зменшення масштабу зображень для швидшої обробки
    imgL = cv2.pyrDown(cv2.imread('houseL.jpg'))
    imgR = cv2.pyrDown(cv2.imread('houseR.jpg'))

    # Налаштування параметрів стерео-обробки
    window_size = 5
    min_disp = 3
    num_disp = 30 - min_disp
    stereo = cv2.StereoSGBM_create(
        minDisparity=min_disp,
        numDisparities=num_disp,
        blockSize=14,
        P1=8 * 3 * window_size ** 2,
        P2=32 * 3 * window_size ** 2,
        disp12MaxDiff=1,
        uniquenessRatio=10,
        speckleWindowSize=100,
        speckleRange=32
    )

    print('computing disparity...')
    disp = stereo.compute(imgL, imgR).astype(np.float32) / 16.0

    print('generating 3d point cloud...')
    h, w = imgL.shape[:2]
    f = 0.8 * w                          # Фокусна відстань
    Q = np.float32([[1,  0, 0, -0.5 * w],
                    [0, -1, 0,  0.5 * h],   # поворот точок на 180 градусів навколо X,
                    [0,  0, 0, -f      ],   # щоб вісь Y дивилась угору
                    [0,  0, 1,  0      ]])
    points = cv2.reprojectImageTo3D(disp, Q)
    colors = cv2.cvtColor(imgL, cv2.COLOR_BGR2RGB)
    mask = disp > disp.min()
    out_points = points[mask]
    out_colors = colors[mask]
    out_fn = 'out.ply'
    write_ply('out.ply', out_points, out_colors)
    print('%s saved' % 'out.ply')

    cv2.imshow('left', imgL)
    cv2.imshow('disparity', (disp - min_disp) / num_disp)
    cv2.imshow('disparity', (disp - min_disp) / num_disp)

    print("Load a ply point cloud, print it, and render it")
    pcd = o3d.io.read_point_cloud("out.ply")
```

```
    print(pcd)
    print(np.asarray(pcd.points))
    o3d.visualization.draw_geometries([pcd], width=650, height=650, left=20, top=20)

    cv2.waitKey()
    cv2.destroyAllWindows()
```

Але нас в контексті завдання цікавить більше не скрипкова реалізація, а дослідження впливу параметрів стерео-обробки на результат.

Дослідження впливу параметрів

Параметри, що впливають на точність отриманої тривимірної точкової хмари можна поділити на 2 групи:

Параметри стерео-обробки:

- $window\_size$ – Розмір вікна для порівняння блоків між двома зображеннями;
- min_disp – Мінімальне значення диспаритету;
- num_disp – Кількість можливих значень диспаритету (має бути кратним 16);
- $blockSize$ – Розмір блоків для порівняння;
- $P1$ – Параметр для контролю згладжування диспаритету (менші значення);
- $P2$ – Параметр для контролю згладжування диспаритету (більші значення);
- $disp12MaxDiff$ – Максимальна різниця між лівим і правим диспаритетом для визнання пари валідною;
- $uniquenessRatio$ – Відсоток унікальності для фільтрації поганих матчів;
- $speckleWindowSize$ – Розмір вікна для видалення малих "плям" шуму;
- $speckleRange$ – Максимальна різниця диспаритету в межах вікна для видалення "плям".

Параметри проекції 3D точок:

- f – Фокусна відстань камери;
- Q – Матриця проекції для перетворення 2D диспаритету в 3D точки.

Дослідимо вплив параметрів $window\_size$, min_disp, num_disp та $blockSize$.

Параметр $window\_size$

Запустимо скрипт з параметрами:

```
window_size = 3
min_disp = 16
num_disp = 112 - min_disp
stereo = cv2.StereoSGBM_create(
    minDisparity=min_disp,
    numDisparities=num_disp,
    blockSize=16,
    P1=8 * 3 * window_size ** 2,
    P2=32 * 3 * window_size ** 2,
    disp12MaxDiff=1,
    uniquenessRatio=10,
    speckleWindowSize=100,
    speckleRange=32
)
```

Де $window\_size$ = 3.

Результат:

У результаті контур 3D об’єкта не вибудовується зовсім. Утім структуру дальніх об’єктів (місто на фоні) передано.

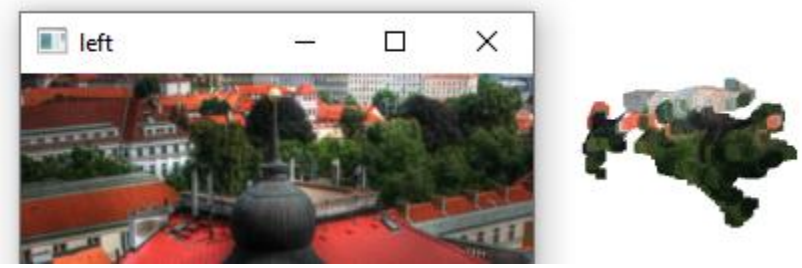


Рисунок 4 – Реконструкція дальніх об’єктів

Залишимо всі вхідні параметри такими ж, але змінимо $window\_size$ = 5.

Результат:

Контур 3D об’єкта усе ще не вибудовується. Утім структуру дальніх об’єктів реалізовано краще, уже видно об’єм листви дерев.

Рисунок 5 – Детальніша реконструкція дальніх об’єктів

Подальше збільшення значення цього параметру призводить до появи більшої кількості шумів та артефактів, але покращенню деталізації 3D об’єкта не сприяє.

Бачимо, що в нашому випадку, **збільшення** параметру **розміру вікна** до появи шуму **сприяє більшій деталізації об’єктів заднього фону**.

Параметр *min_disp*

Запустимо скрипт з параметрами:

```
window_size = 5
min_disp = 8
num_disp = 104 - min_disp
stereo = cv2.StereoSGBM_create(
    minDisparity=min_disp,
    numDisparities=num_disp,
    blockSize=16,
    P1=8 * 3 * window_size ** 2,
    P2=32 * 3 * window_size ** 2,
    disp12MaxDiff=1,
    uniquenessRatio=10,
    speckleWindowSize=100,
    speckleRange=32
)
```

Де min_disp = 8.

Результат:

У результаті починає вибудовуватись контур 3D об’єкта. У профілі видно, що об’єкт набуває об’єму. Але більшість об’єкту все ще залишається невизначеною.

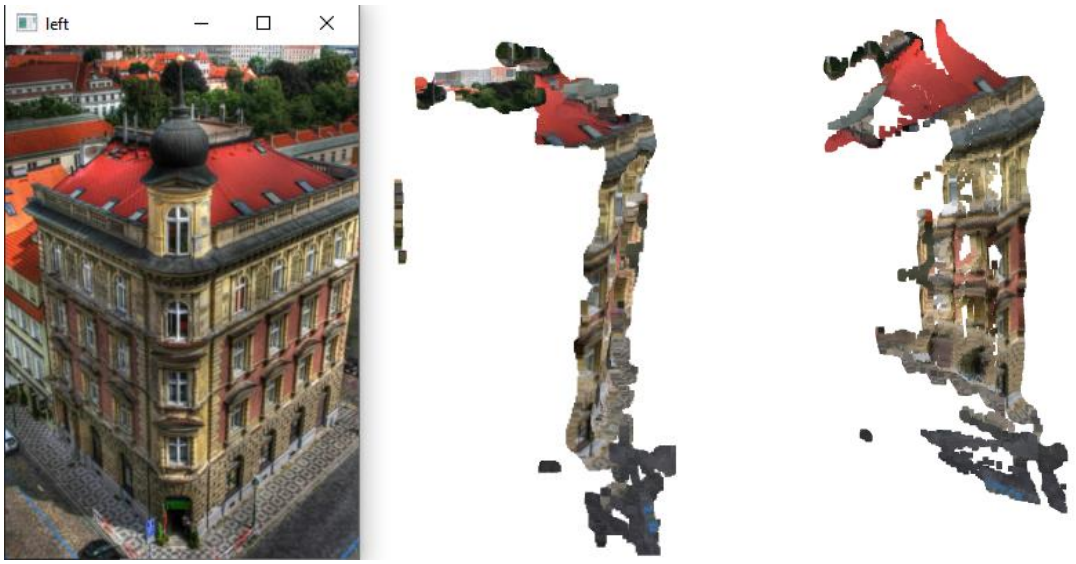


Рисунок 6 – Реконструкція контуру об’єкта

Подальше зменшення цього параметру призведе до появи шумів, а збільшення – до втрати важливих точок і контура об’єкта.

Для нашого зображення видно, що **зменшення мінімального значення диспаритету** до появи шумів **призводить до збільшення діапазону виявлення об’єкта** і покращення точності тривимірної точкової хмари.

Параметр *num_disp*

Запустимо скрипт з параметрами:

```
window_size = 5
min_disp = 8
num_disp = 40 - min_disp
stereo = cv2.StereoSGBM_create(
    minDisparity=min_disp,
    numDisparities=num_disp,
    blockSize=16,
    P1=8 * 3 * window_size ** 2,
    P2=32 * 3 * window_size ** 2,
    disp12MaxDiff=1,
    uniquenessRatio=10,
    speckleWindowSize=100,
    speckleRange=32
)
```

Де num_disp = 32.

Результат:

У нас починає вибудовуватись 3D об’єкт – це добре видно з карти диспаритету (перепади глибини об’єкта стають плавнішими). Утім ближні об’єкти усе ще не відображаються.

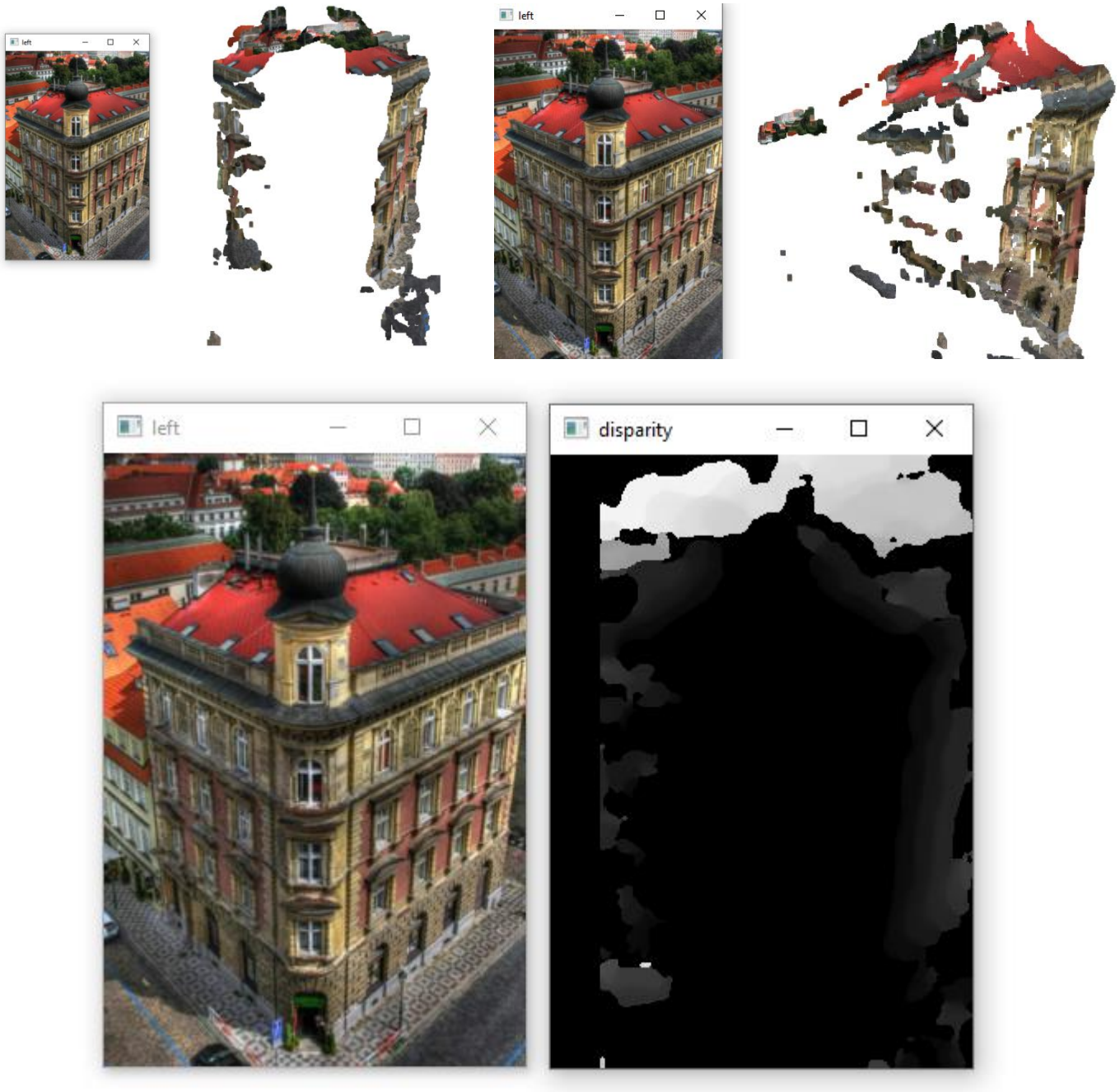


Рисунок 7 – Краща передача глибина об’єкта

Подальше зменшення цього параметру призведе до втрати важливих точок, а збільшення – появи шумів.

Для нашого зображення видно, що **зменшення кількості можливих значень диспаритету** до втрати важливих точок **призводить до покращення передачі глибини об'єкту**.

Параметр *blockSize*

Запустимо скрипт з параметрами:

```
window_size = 5
min_disp = 8
num_disp = 40 - min_disp
stereo = cv2.StereoSGBM_create(
    minDisparity=min_disp,
    numDisparities=num_disp,
    blockSize=14,
    P1=8 * 3 * window_size ** 2,
    P2=32 * 3 * window_size ** 2,
    disp12MaxDiff=1,
    uniquenessRatio=10,
    speckleWindowSize=100,
    speckleRange=32
)
```

Де $blockSize = 14$.

Результат:

3D об'єкт передається ще краще. Збільшилась деталізація дальніх об'єктів (країв будівлі), зменшились пропуски даних – це добре видно в профіль об'єкту, додалась невелика кількість шумів – видно на карті диспаритету. Утім ближні об'єкти усе ще не відображаються.

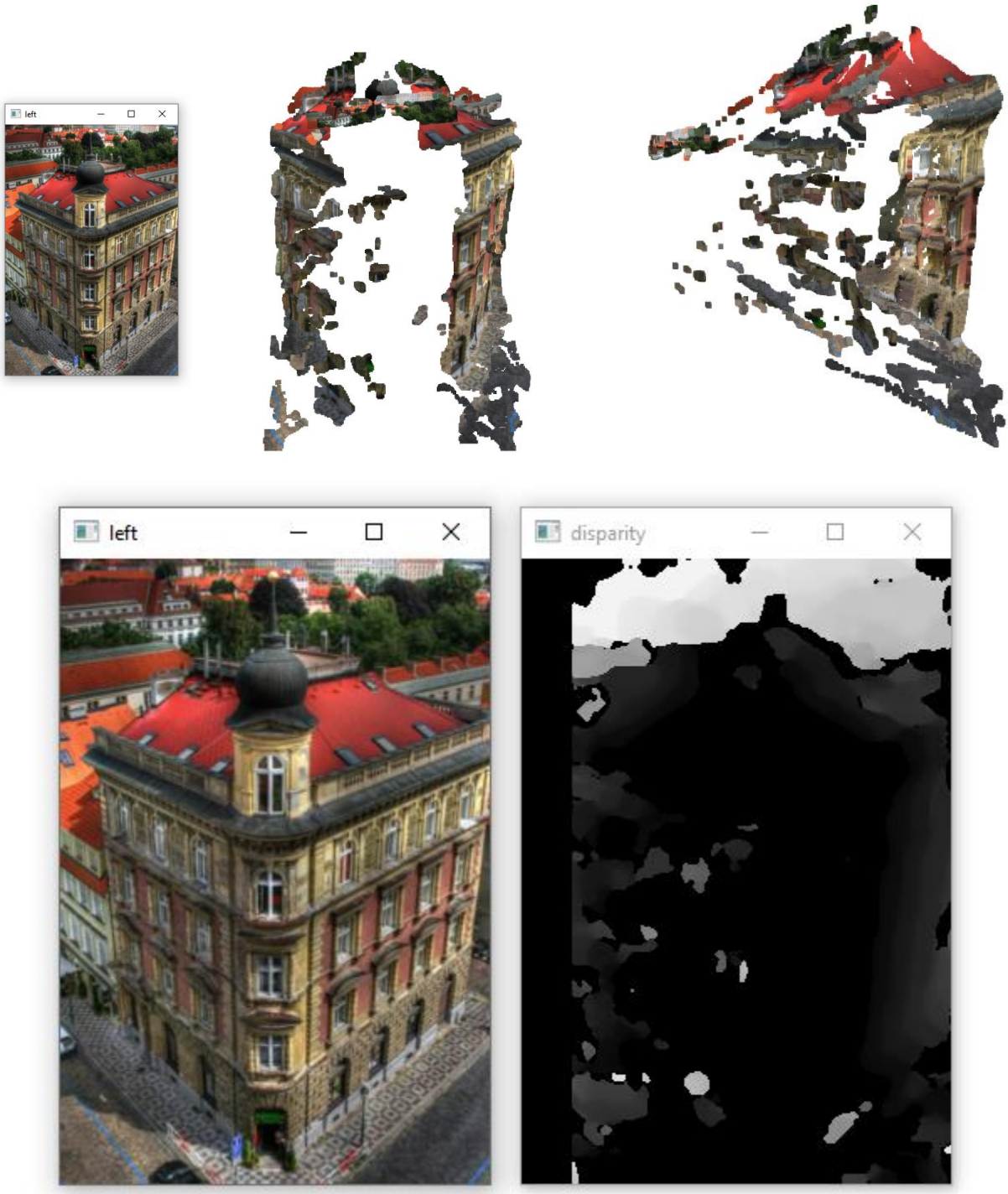


Рисунок 8 – Збільшення деталізації дальніх об’єктів

Подальше зменшення цього параметру підвищить деталізацію, але збільшить шум та кількість артефактів.

Для нашого зображення видно, що **зменшення розміру блоків для порівняння** до появи шумів **призводить до покращення передачі глибини об’єкту**.

Повторний аналіз параметру min_disp

Тепер повторно проаналізуємо вплив параметру мінімального значення диспаритету, використовуючи раніше отримані «оптимальні» значення інших параметрів.

Запустимо скрипт з параметрами:

```
window_size = 5
min_disp = 6
num_disp = 38 - min_disp
stereo = cv2.StereoSGBM_create(
    minDisparity=min_disp,
    numDisparities=num_disp,
    blockSize=14,
    P1=8 * 3 * window_size ** 2,
    P2=32 * 3 * window_size ** 2,
    disp12MaxDiff=1,
    uniquenessRatio=10,
    speckleWindowSize=100,
    speckleRange=32
)
```

Де min_disp = 6.

Результат:

Створена хмара краще передає сфотографований об’єкт. Почали передаватись ближні об’єкти, зменшились пропуски даних – це добре видно в профіль об’єкту. Утім передня частина будівлі усе ще не відображається.

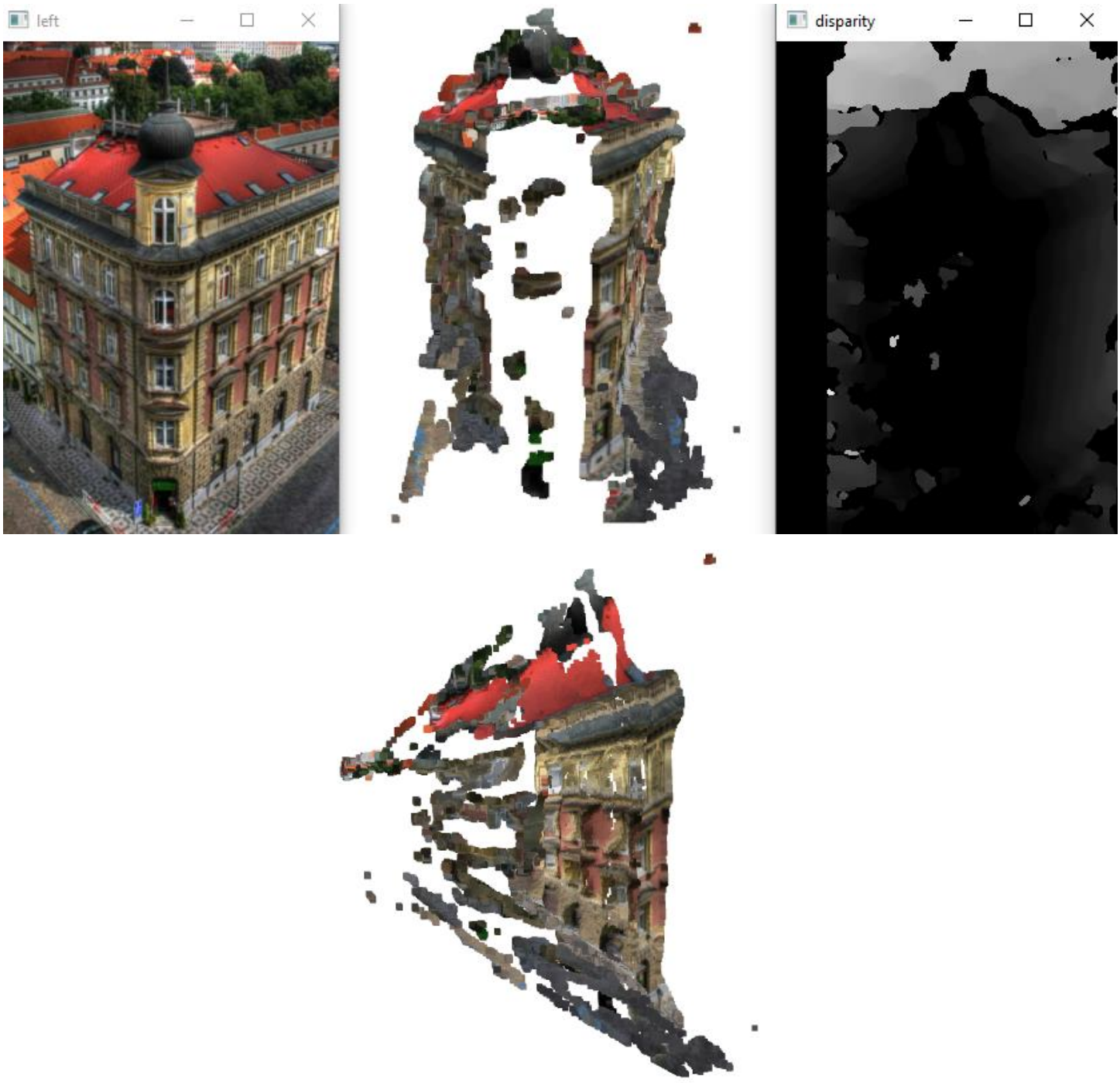


Рисунок 9 – Реконструкція ближніх об’єктів

Залишимо всі вхідні параметри такими ж, але змінимо $min\_disp = 4$.

Результат:

Ближні об’єкти передались краще, утім будівля починає ніби «стискатись».

Рисунок 10 – Детальніша реконструкція ближніх об’єктів

Залишимо всі вхідні параметри такими ж, але змінимо min_disp = 3.

Результат:

Ближні об’єкти передались майже повністю, утім будівля помітно «стислась», а її дальні об’єкти «збігаються» до однієї точки.

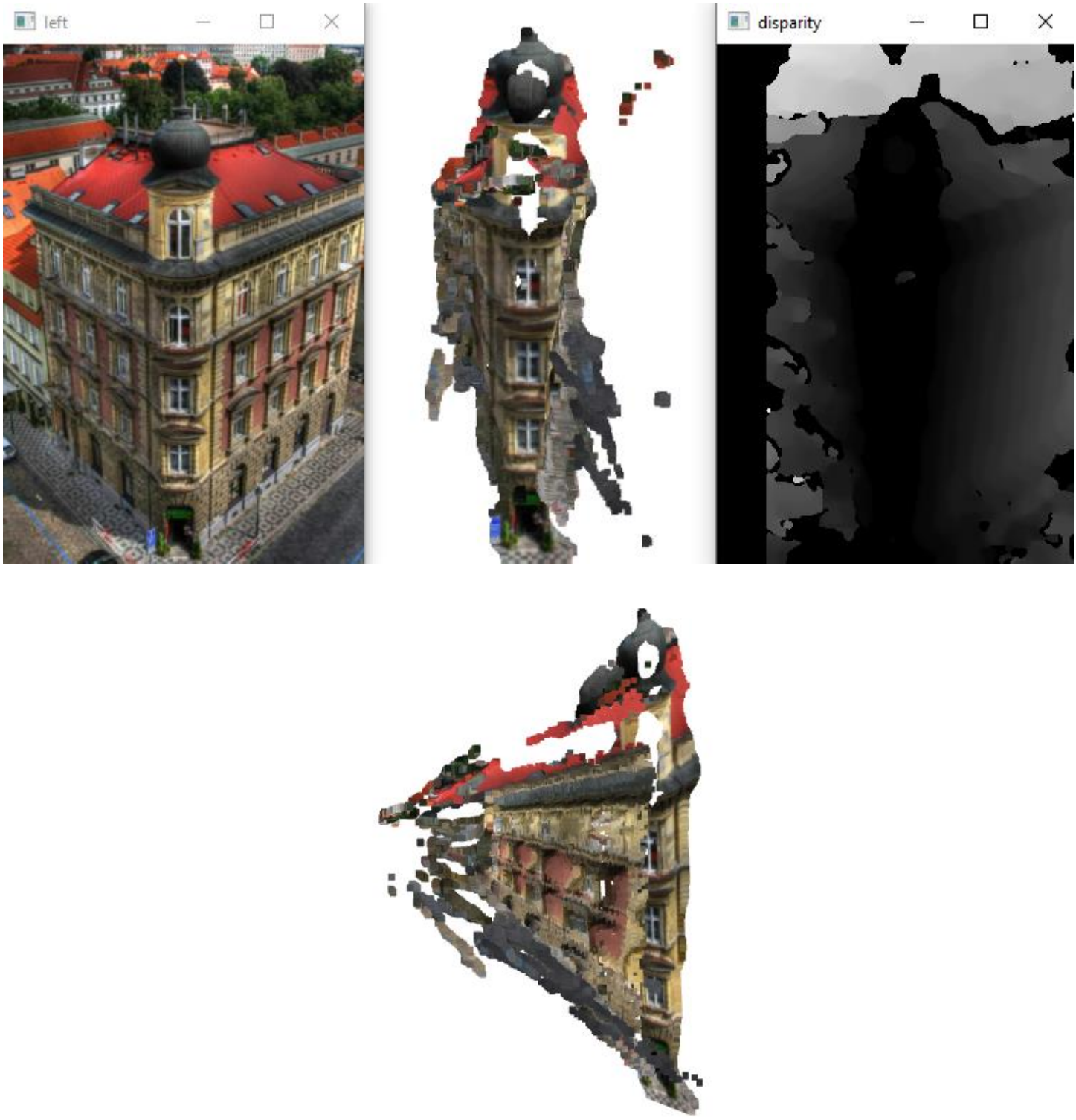


Рисунок 11 – Найкраща реконструкція ближніх об’єктів

Помічено **цікавий ефект зменшення $\boldsymbol{min\_disp}$**.

Зменшення мінімального значення диспаритету також впливає на розрахунок глибини точок. **Коли диспаритет зменшується, об'єкти здаються ближчими до камер**, що призводить до зменшення ширини об'єкта у 3D просторі, так званого «стиснення».

2.2. Аналіз отриманих результатів

Обрано стереопару – підтверджено рисунком.

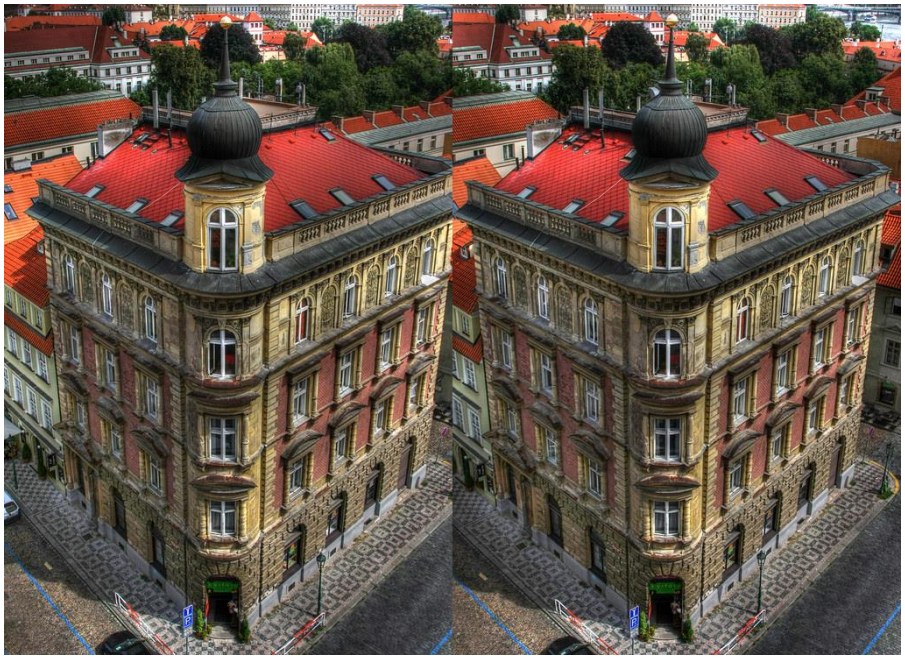


Рисунок 12 – Підтвердження виконання вимог 1

Здійснено 3D реконструкцію будівлі – підтверджено рисунком.

Рисунок 13 – Підтвердження виконання вимог 2

Помічено влив зменшення мінімального диспаритету на розрахунок глибини точок – підтверджено рисунками.

Рисунок 14 – Підтвердження виконання вимог 3

Висновок:

У результаті виконання лабораторної роботи отримано практичні та теоретичні реконструкції 3D об’єктів із стереопари зображень.

Реалізовано програмний скрипт, який виконує визначення растрових матриць зображень, визначення карти глибин зображень та будує на основі диспаритету зображень тривимірну точкову хмару, яка відповідає 3D об’єкту.

У ході виконання завдання проведено аналіз впливу параметрів стерео-обробки на кінцевий результат генерації об’єкту і виявлено цікавий ефект зменшення параметру min_disp (мінімального диспаритету): «Коли диспаритет зменшується – це призводить до зменшення ширини об'єкта у 3D просторі».