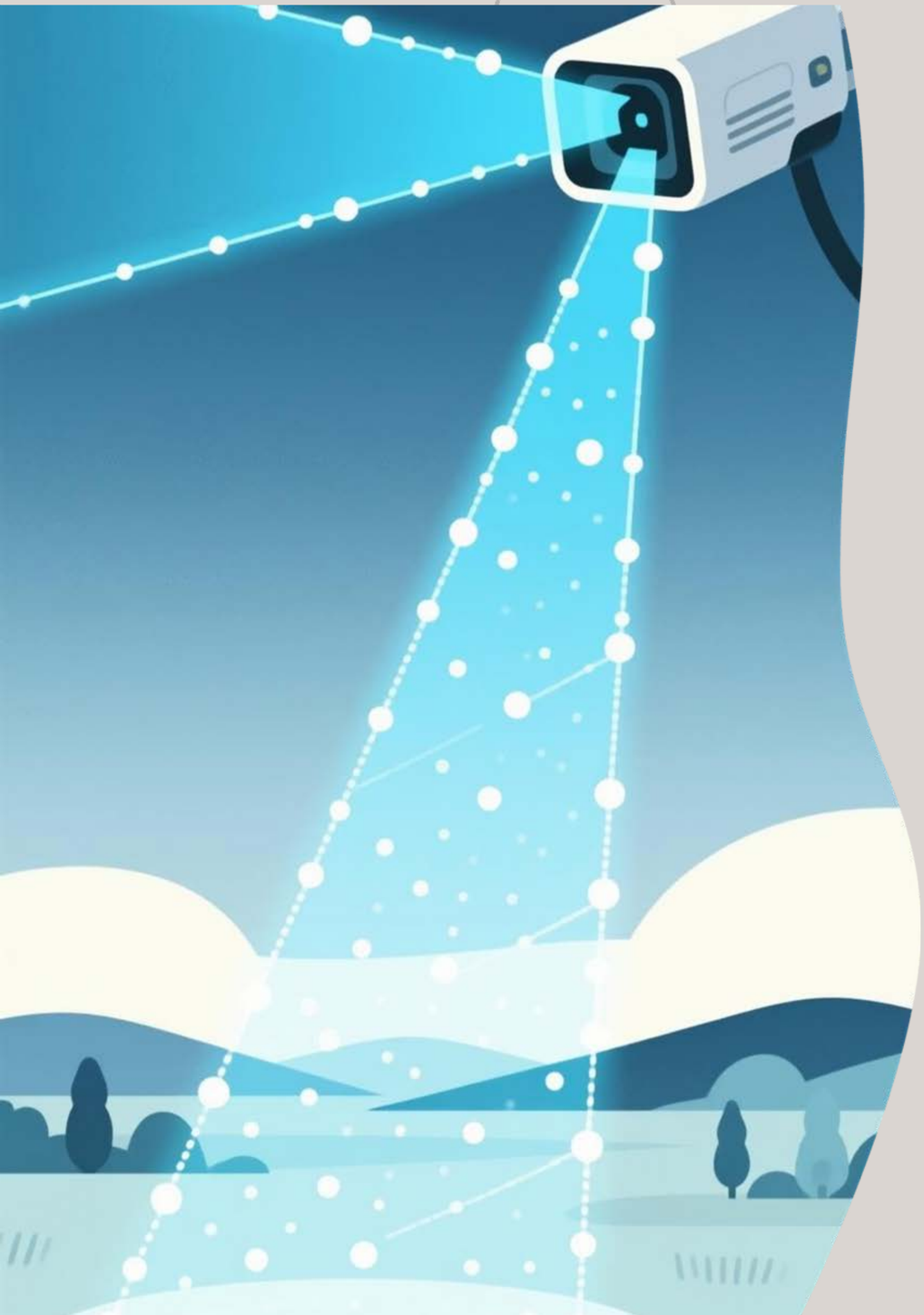
Міністерство освіти та науки України
Київськийнаціональнийуніверситетбудівництва та архітектури

# Розробка методів автоматизованої обробки LiDAR-даних для геопросторового картографування

Виконала: студентка групи ГСТм-24
Демченко М.А.
Керівник: к.т.н. Горковчук Ю.В

Київ-2025



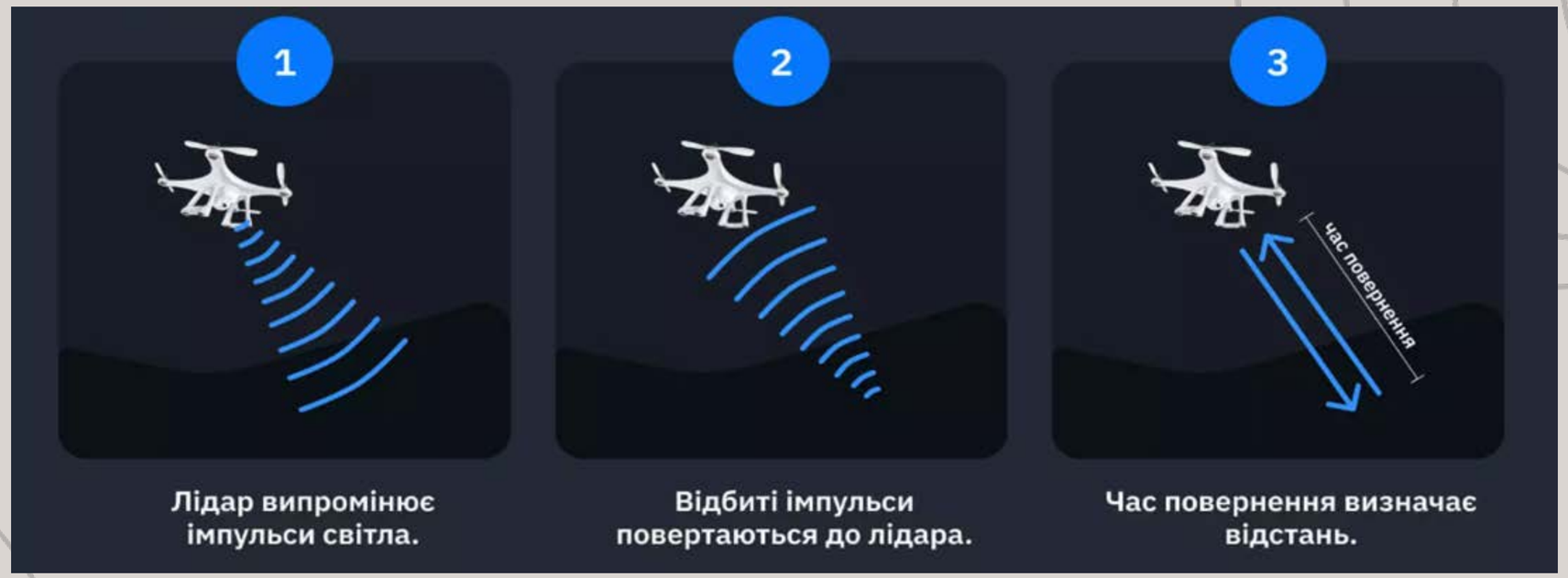
# Вступ

LiDAR є ключовою технологією сучасного топографічного картографування, оскільки забезпечує точні 3D-дані про рельєф та об’єкти місцевості. Попри наявність програмних рішень, обробка таких даних часто потребує ручної роботи, що знижує ефективність.

Метою роботи було розробити автоматизовану методику та програмний інструмент для швидкої й стандартизованої обробки LiDAR-даних у створенні цифрового топографічного плану. Об’єктом роботи був процес обробки LiDAR-даних.

Предметом були методи та програмні засоби автоматизації.

# LIDAR

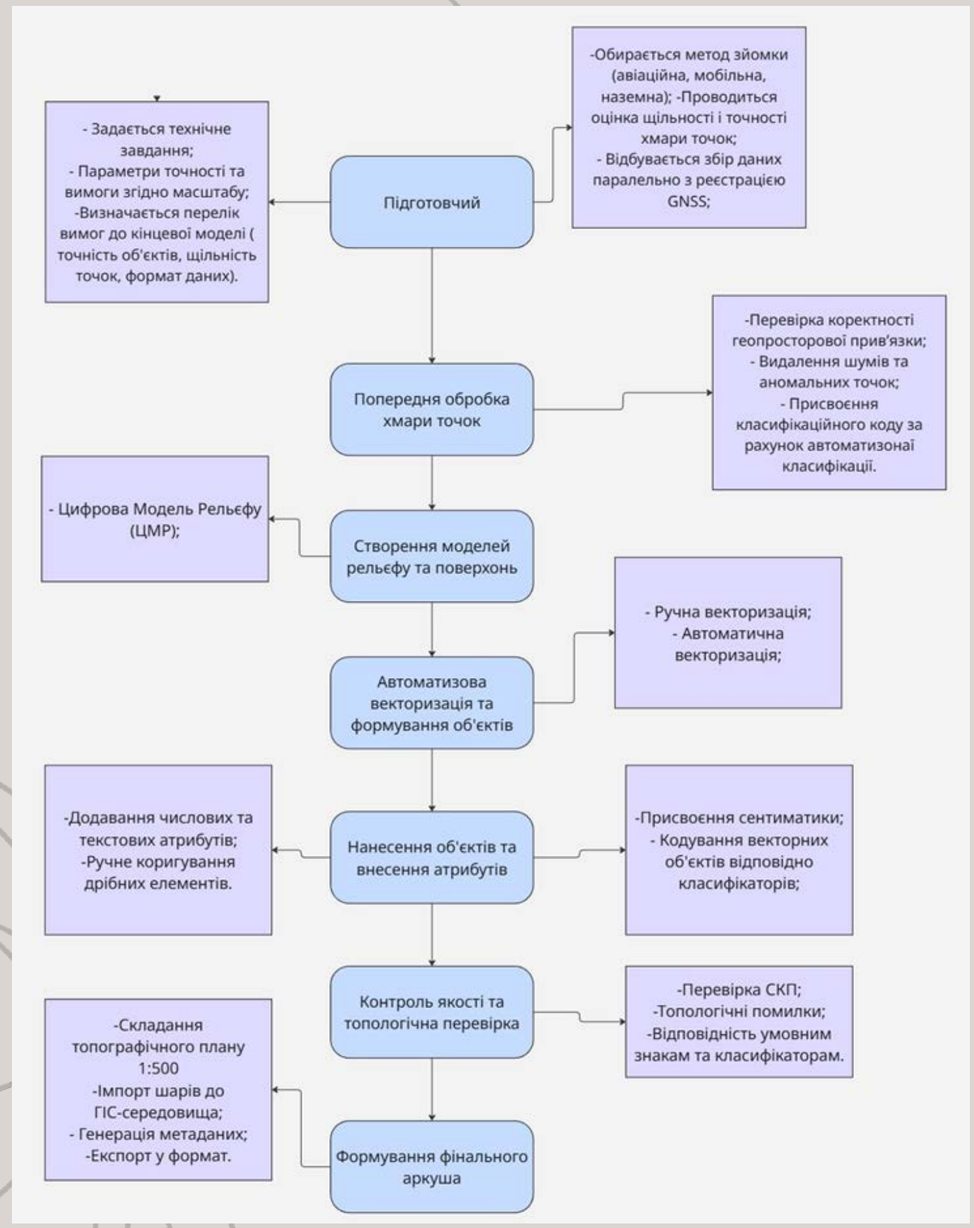



Принципи роботи лідару

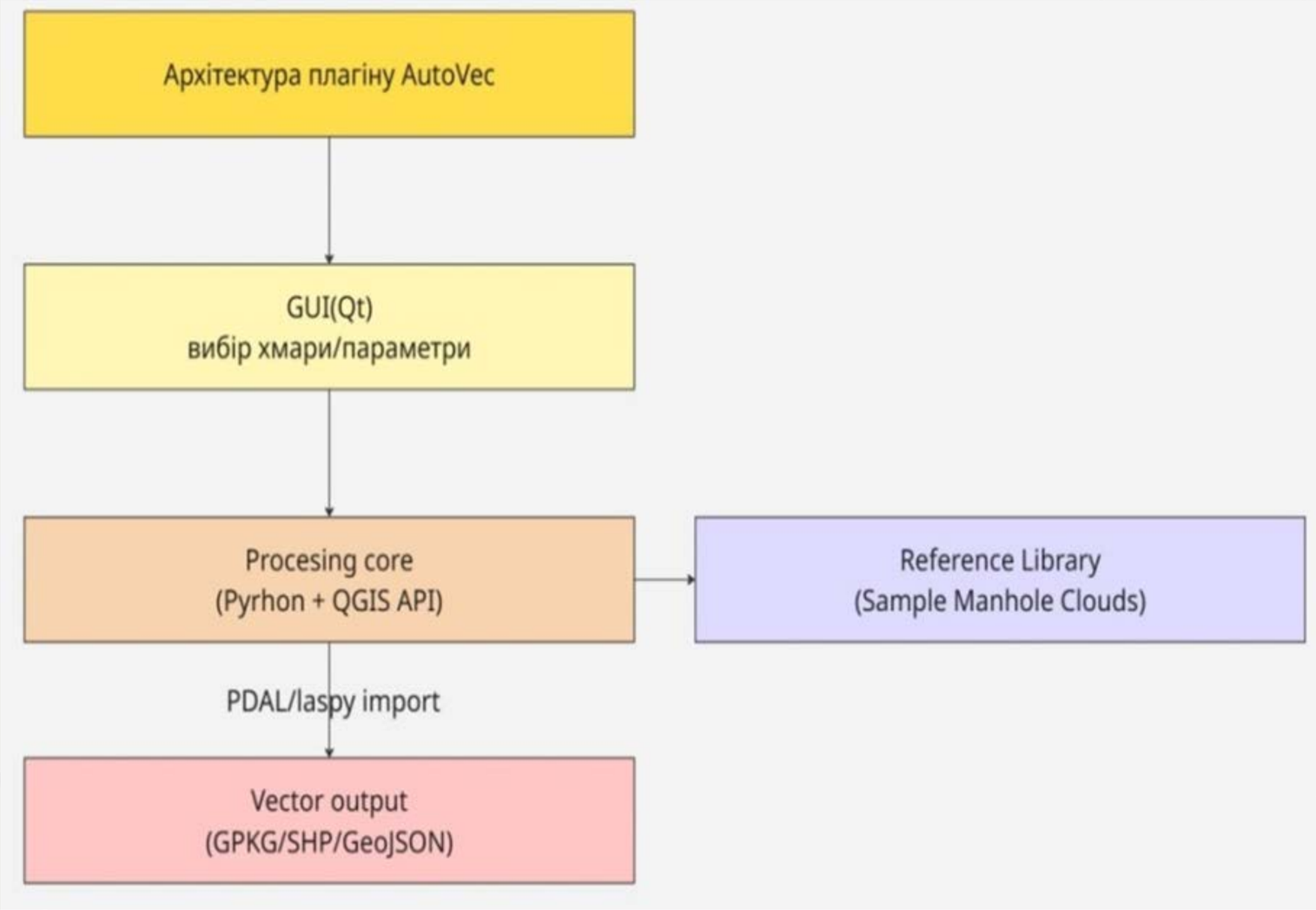
# Порівняльна таблиця ПЗ для перегляду, аналізу та ручної обробки

| Назва ПЗ | Тип ліцензії | Основні функції | Переваги | Сфера застосування |
|---|---|---|---|---|
| Autodesk ReCap Pro | Комерційна | Імпорт, реєстрація, 3D-моделювання | Інтеграція з Revit | BIM, геодезія |
| Hexagon 3DReshaper | Комерційна | Mesh, аналіз поверхонь | Висока точність | Топографія |
| CREDO 3D СКАН | Комерційна | Класифікація, ЦМР/ЦМП | Оптимізація для доріг | Інженерна геодезія |
| VisionLidar | Комерційна | Фільтрація, аналіз у реальному часі | Швидка обробка | Містобудування |
| Pix4Dmapper | Комерційна | Хмари точок, ортофото | Фотограмметрія | Будівництво, агро |
| CloudCompare | Відкрита | Редагування, сегментація | Широкий формат, швидкість | Геодезія |

# Етапи цифрового картографування на основі LiDAR




# Архітектура плагіну «AutoVec»



# Структура програмного продукту та середовище розроблення

```
# 1. Заповнення полів вибору формату (comboBox_2)
self.comboBox_2.addItems(["GeoPackage (*.gpkg)", "Shapefile (*.shp)", "GeoJSON (*.geojson)"])
```

Заповнення полів вибору формату (comboBox_2)

```
class AutoVecDialog(QDialog, Ui_AutoVecDialog):
    """
    Клас діалогового вікна, який керує інтерфейсом та запускає векторизацію.
    """

    def __init__(self, parent=None):
        """Конструктор."""
        super(AutoVecDialog, self).__init__(parent)
        self.setupUi(self)
        self.iface = parent
```

Клас діалогового вікна, який керує інтерфейсом та запускає векторизацію

```
def select_cloud_file(self):
    """Відкриває діалог для вибору вхідної хмари точок (LAS/LAZ/XYZ)."""
    options = QFileDialog.Options()
    fileName, _ = QFileDialog.getOpenFileName(
        self,
        "Виберіть файл хмари точок",
        "",
        "Cloud Files (*.las *.laz *.xyz);;All Files (*)",
        options=options
    )
    if fileName:
        self.lineEdit.setText(fileName)  # lineEdit - поле Cloud Path
```

Відкриває діалог для вибору вхідної хмари точок (LAS/LAZ/XYZ)

```
# 3. Завантаження останнього використаного CRS
s = QgsSettings()
last_crs = s.value("AutoVec/last_output_crs", "EPSG:4326")
self.lineEdit_3.setText(last_crs)  # lineEdit_3 - поле Output CRS
```

Завантаження останньої використаної СК

```
def run_autovectorization(self):
    """Основна функція, яка збирає параметри та запускає алгоритм."""

    if not HAS_LASPY:
        QMessageBox.critical(self, "Помилка",
                             "Бібліотека 'laspy' не знайдена. Необхідна для обробки хмар точок. Встановіть її.")
        return

    # --- 1. Збір та валідація параметрів ---
    input_cloud_path = self.lineEdit.text()
    output_vector_path = self.lineEdit_2.text()
    output_crs_text = self.lineEdit_3.text()

    voxel_size = self.doubleSpinBox.value()
    min_pts = self.doubleSpinBox_2.value()
    max_pts = self.doubleSpinBox_3.value()
    max_dist = self.doubleSpinBox_4.value()

    if not input_cloud_path or not output_vector_path:
        QMessageBox.warning(self, "Помилка", "Будь ласка, оберіть вхідний файл та шлях для вихідного файлу.")
        return

    crs = QgsCoordinateReferenceSystem(output_crs_text)
```

Основна функція, яка збирає параметри та запускає алгоритм

# Функція PDAL-пайплайн на основі JSON-шаблону

```python
def run_pdal_pipeline(input_cloud, json_template_path, output_cloud):

    #Виконує PDAL-пайплайн на основі JSON-шаблону,
    #що використовується для кольорокодової фільтрації та попередньої
    #ідготовки хмари точок перед векторизацією.

    #:param input_cloud: шлях до вхідного LAS/LAZ
    #:param json_template_path: шлях до JSON-конфігурації
    #:param output_cloud: шлях до файлу, куди буде записано очищену хмару
    # 1. Завантаження JSON-шаблону
    with open(json_template_path, "r", encoding="utf-8") as f:
        pipeline_json = json.load(f)

    # 2. Підміна вхідного та вихідного файлів
    pipeline_json["pipeline"][0]["filename"] = input_cloud
    pipeline_json["pipeline"][-1]["filename"] = output_cloud

    # 3. Створення тимчасового JSON
    tmp_json = tempfile.NamedTemporaryFile(delete=False, suffix=".json")
    with open(tmp_json.name, "w", encoding="utf-8") as f:
        json.dump(pipeline_json, f, indent=4)

    # 4. Виконання PDAL через subprocess
    cmd = ["pdal", "pipeline", tmp_json.name]
```

```python
    json.dump(pipeline_json, f, indent=4)

# 4. Виконання PDAL через subprocess
cmd = ["pdal", "pipeline", tmp_json.name]

process = subprocess.Popen(
    cmd,
    stdout=subprocess.PIPE,
    stderr=subprocess.PIPE,
    shell=False,
    text=True
)

out, err = process.communicate()

if process.returncode != 0:
    raise Exception(f"PDAL pipeline failed: {err}")

return output_cloud
```

# JSON-шаблон

```
#JSON-шаблон задає логіку фільтрації
{
  "pipeline": [
    { "type": "readers.las", "filename": "" },
    {
      "type": "filters.range",
      "limits": "Red[0:80],Green[0:80],Blue[0:80]"
    },
    {
      "type": "filters.voxelcenter",
      "cell": 0.20
    },
    { "type": "writers.las", "filename": "" }
  ]
```

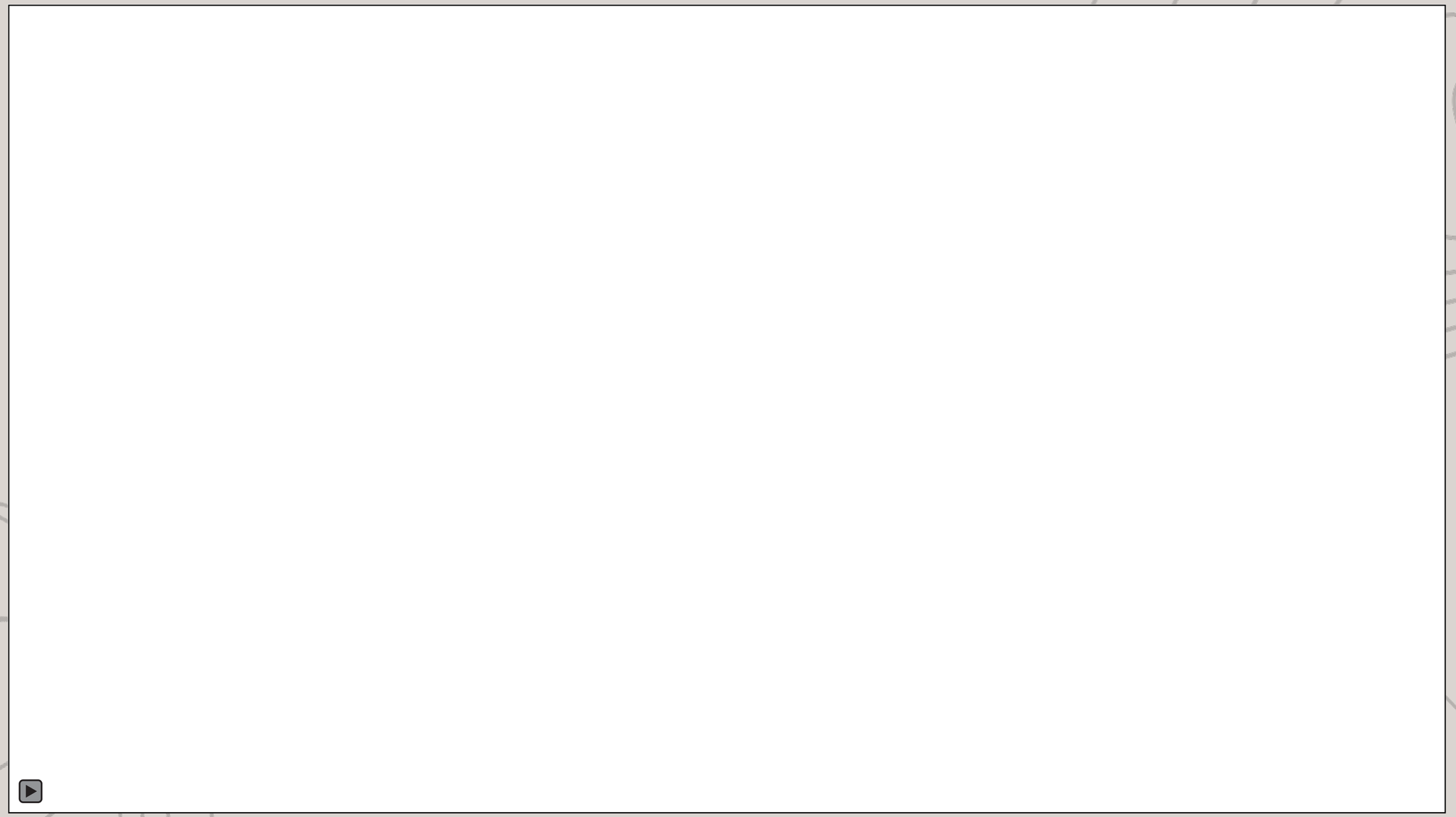

# Вихідні дані

Вихідними даними була хмара точок в форматі .LAS з RGB – атрибутами зі щільністю точок 100 pts/m^2 територія міста Ставангер, Норвегія з системою координат EPSG:4326, розмір ласу довірнював 6 гб.

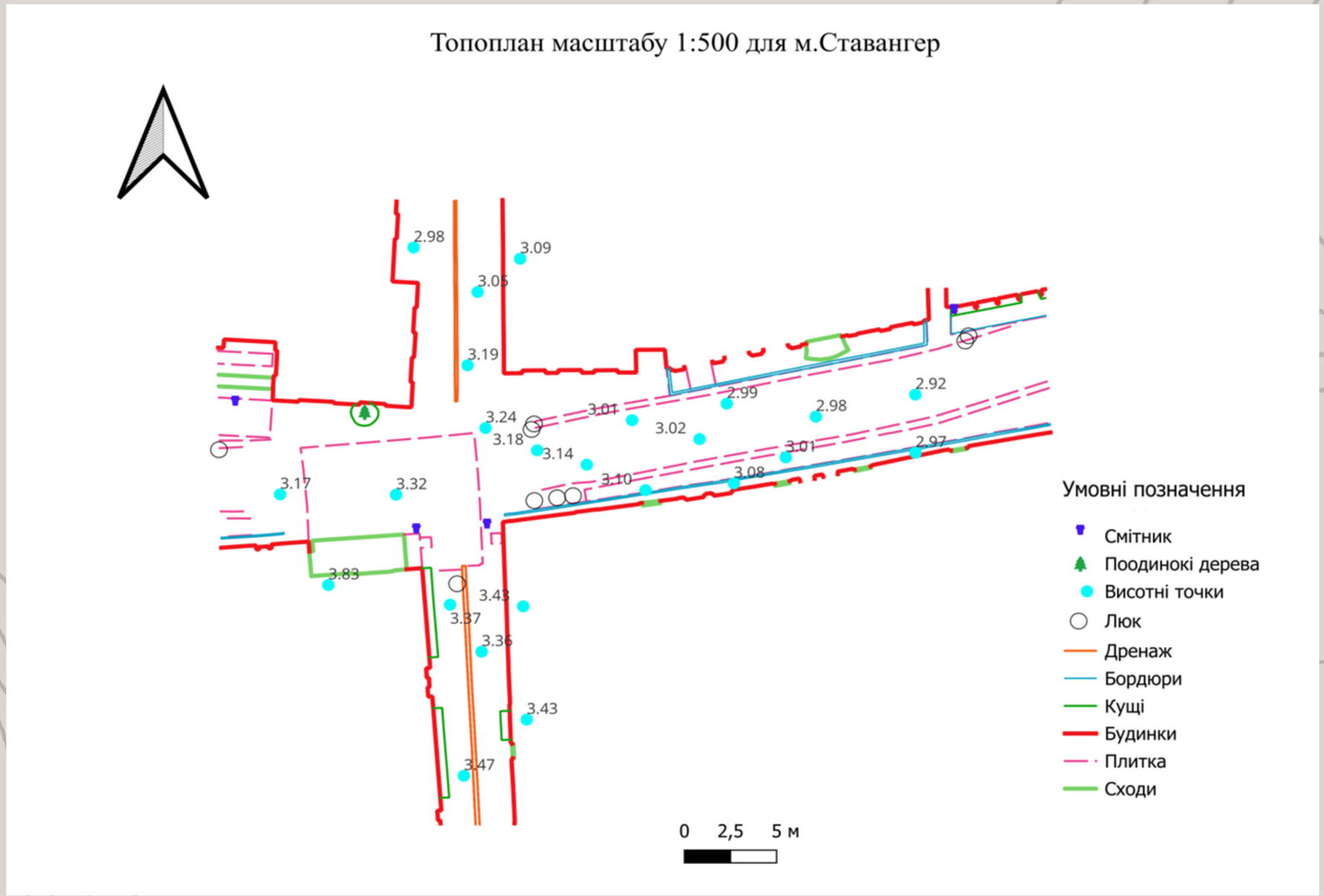
Умовами експерименту було створення топографічного плану масштабу 1:500, до якого необхідно було внести такі об’єкти :

- дренажі(Drainage);
- будинки (houses);
- люки(Hatches);
- висотні точки (High points);
- бордюри (curbs);сходи (stairs);
- мусорні баки(garbage cans);
- дерева (trees);
- плитка (tegel);
- кущі (bushes);

# Відео роботи плагіну «AutoVec»



# Топографічний план масштабу 1:500 м. Ставангер



# Таблиця оцінки роботи плагіну «AutoVec»

| Показник | Норма 1:500 | Фактичний результат | Висновок |
|---|---|---|---|
| Планова похибка | ≤ 0,05 м | 0,8 мм – 4см | Відповідає |
| Висотна | ≤ 0,05 м | Н/З | Не застосовується |
| Ефективність | ≥ 90 % | 62,5% | Вище середньої |
| Точність | ≥ 90 % | 92 % | Добре |
| Точність роботи | ≤ 10 % | Не виявлено | Не Виявлено |
| Продуктивність | Від 30 с. – 1хв. | Досить продуктивно | Відповідає |

# Висновок

У роботі розроблено та реалізовано методику автоматизованої обробки LiDAR-даних для створення топографічних планів масштабу 1:500. Запропонований QGIS-плагін забезпечує повний цикл векторизації хмар точок і відповідає чинним нормативним вимогам. Результати підтвердили підвищення швидкості обробки, зменшення ручної праці та стабільну якість, що робить методику придатною для практичного впровадження.

# Дякую за увагу!