Computer Vision: Research Paper Reproduction Project

Template Instructions

Course: Computer Vision

Due date: 21-12-2025

Project overview

The goal of this project is to reproduce the results from a peer-reviewed computer vision research paper. Students must implement the algorithm, run experiments on the same or similar datasets, and produce:

1. Well-documented source code (easy to run, uses relative paths only).
2. A short **algorithm description** (pseudocode) submitted separately to the instructor.
3. A written **report** describing data acquisition, preprocessing, implementation details, results (quantitative and qualitative), and discussion.

High-level requirements

- **Paper selection:** Instructor-approved research paper in computer vision (classification, detection, segmentation, depth, tracking, etc.). Provide the citation and a PDF in the project folder.
- **Reproducibility:** Your code must run from the command line with a single command (e.g. `bash run.sh`) and must not require modifying file paths inside source files.
- **Documentation:** Provide a README with detailed instructions, a requirements file (`requirements.txt` or `environment.yml`), and a short usage example.
- **Submission contents:** code/, data/ or data_link.txt (if dataset is large), report.pdf, algorithm.txt (pseudocode), README.md, requirements.txt, run.sh, checkpoints/ (optional), results/ (images/tables).

Directory structure

```
project-name/
|-- code/
|   |-- utils.py
|   |-- main.py
|   |-- run.sh
|-- data/          # small sample data or pointers
|   |-- README-data.txt
|-- results/
|   |-- figures/
|   |-- metrics.csv
```

```
|-- paper.pdf
|-- report.pdf
|-- algorithm.txt
|-- README.md
|-- requirements.txt
```

Important implementation rules

- **Relative paths only:** Use paths relative to the project root. Example in Python:

  ```
  import os
  BASE_DIR = os.path.dirname(os.path.abspath(__file__)) # code/ directory
  DATA_DIR = os.path.join(BASE_DIR, ’..’, ’data’)
  # then use os.path.join(DATA_DIR, ’train’, ’images’)
  ```

- **Single-run script:** Provide a wrapper script (`run.sh`) that installs needed packages (if safe) and runs experiments. Example:

  ```
  # run.sh (make executable)
  #!/usr/bin/env bash
  set -e
  python3 code/main.py --config config.yaml
  ```

- **Seed and randomness:** Set random seeds and report them in the report so results are reproducible.

- **Hardware and runtimes:** Report hardware used (GPU model, memory), and approximate training/inference time.

Data acquisition and processing

Your report must explicitly state how the data was obtained. Options:

1. Provide subset of the original dataset inside data/
2. If dataset is large or licensing-restricted, provide a script or instructions to download and prepare the dataset automatically (e.g. download URLs and preprocessing commands). Put links and step-by-step instructions in `data/README-data.txt`.

Also include a **preprocessing** subsection in the report detailing image resizing, point cloud downsampling, and any filtering.

Algorithm submission

Provide a plain-text file `algorithm.txt` containing polished pseudocode of the core algorithm. Use indentation and brief comments. Example pseudocode format:

Algorithm 1 High-level pseudocode for training

```
procedure TRAIN($model, train\_loader, optimizer, epochs$)
    $model$.train()
    for $e = 1$ to $epochs$ do
        for each batch $(x, y)$ in $train\_loader$ do
            $pred \leftarrow model(x)$
            $loss \leftarrow L(pred, y)$
            $optimizer$.zero_grad()
            $loss$.backward()
            $optimizer$.step()
```

Report template (use this structure for report.pdf)

1. **Title, authors, article**

2. **Summary** (max 200 words)

3. **Implementation**: No need to go in depth, explain in your own words how each step looks like.
   - Data acquisition (sources, links)
   - Preprocessing steps
   - Algorithm
   - Any differences from the original paper (if you made changes)

4. **Results**:
   - Quantitative results (tables with similar tests as paper)
   - Qualitative results (figures: sample outputs, failure cases)

5. **Discussion** (compare your results to the paper; possible reasons for differences)

6. **Reproducibility checklist** (below)

Reproducibility checklist (to include in report)

Code: Provided and documented (README).

Data: Included or download script provided.

Environment: requirements.txt or environment.yml included.

Run instructions: Single command `bash run.sh` (documented).

Random seeds: values and how they were set.

Expected outputs: expected metric numbers.

Submission instructions

Students should create a single compressed archive (`project-name.zip`) containing everything except very large raw datasets (instead include data_link.txt). Send the zip to Bilal.Moussa.Fares@vub.be .

Short checklist

1. Fill in the project root README with: paper citation, contact names, how to run, hardware used.
2. Ensure `run.sh` runs without editing any file.
3. Ensure all paths in the code are relative (no absolute /home/.. or C: paths).
4. Provide expected metric numbers in the README for quick verification.

Small examples (utility snippets)

Python: relative path helper

```
# code/utils.py
import os
ROOT = os.path.abspath(os.path.join(os.path.dirname(__file__), '..'))
DATA_DIR = os.path.join(ROOT, 'data')
def data_file(*parts):
    return os.path.join(DATA_DIR, *parts)

# usage: data_file('train', 'image1.jpg')
```

Example run.sh

```
#!/usr/bin/env bash
set -e
# Activate environment if needed (optional): source venv/bin/activate
python3 code/train.py --config config.yaml
python3 code/evaluate.py --ckpt checkpoints/best.pth --out results/metrics.csv
```