The open source software underpinning the 3DBAG

Balázs Dukai, Ravi Peters, Gina Stavropoulou

FOSS4G BE+NL

26-09-2024, Baarle-Nassau

3DGI

TU Delft 3Dgeoinfo

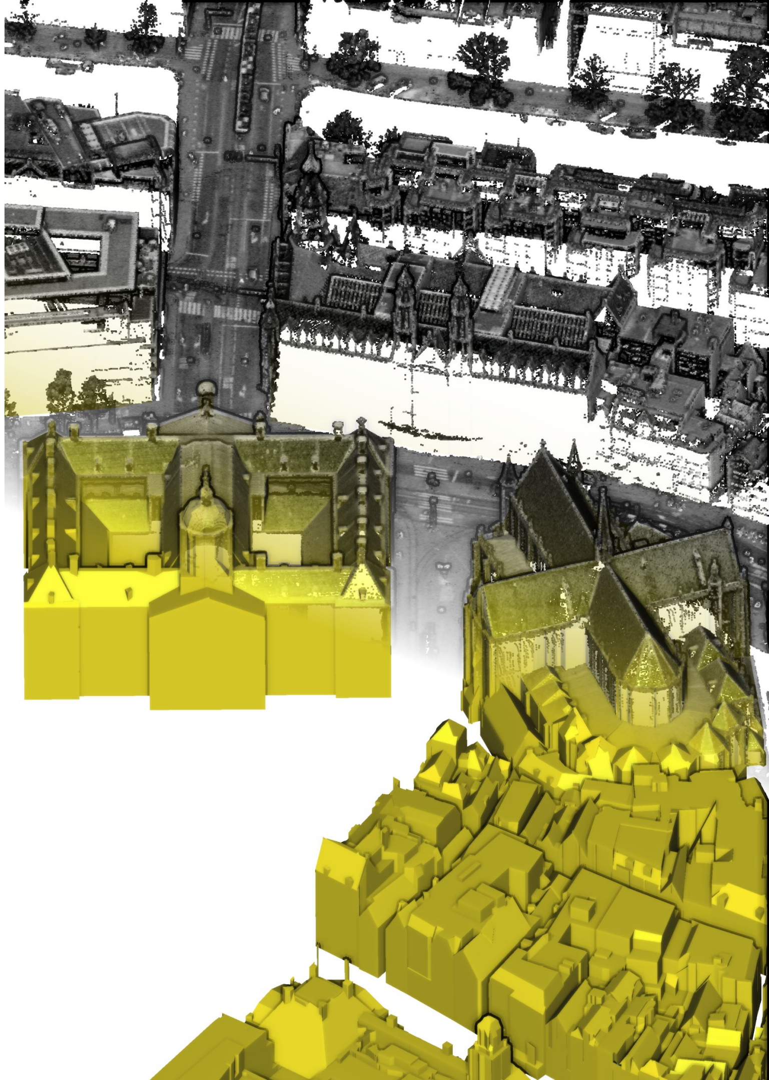
What is the 3DBAG?

A dataset with **detailed 3D models** for **all buildings** in the Netherlands available at 3dbag.nl

- ~10 million buildings
- Detailed "LoD2" roof structures

- Made using open data (AHN, BAG)
- Available as open data

- Built and maintained by a small team (3-5 persons) @ TU Delft 3D geoinformation + 3DGI



3DBAG open source software fundament

- To make 3DBAG possible we wrote a lot of code
- Over time this code became overly complex and increasingly difficult to use and maintain
- To improve this situation we spent last summer to rework and refactor the core 3DBAG software components
- This effort was funded by Kadaster
- Future 3DBAG software maintenance through *3DBAG innovation platform*

This presentation

1. **roofer**
   Our standalone automatic buildings reconstruction software
2. **3dbag-pipeline**
   Coordinates and manages the entire 3DBAG release process including downloading input data, data preprocessing, building reconstruction, data postprocessing, format conversions and deployment

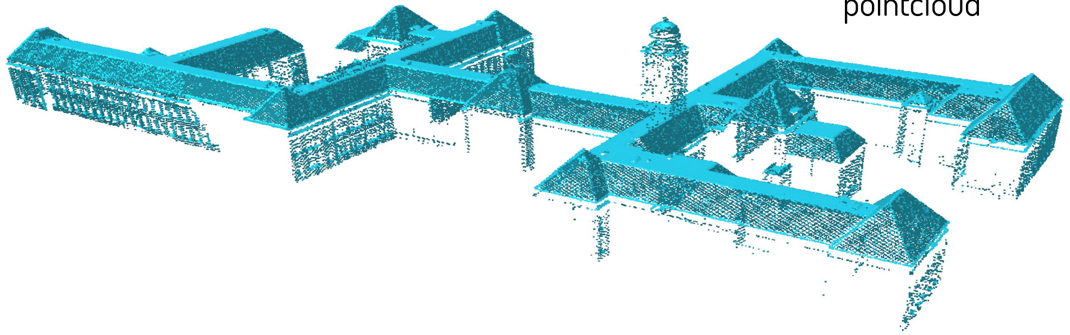
roofer

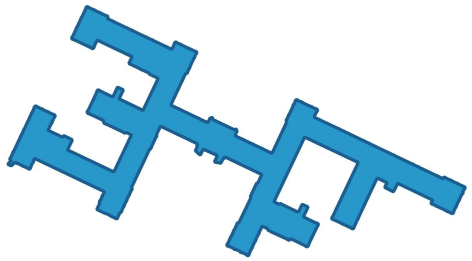
Overview

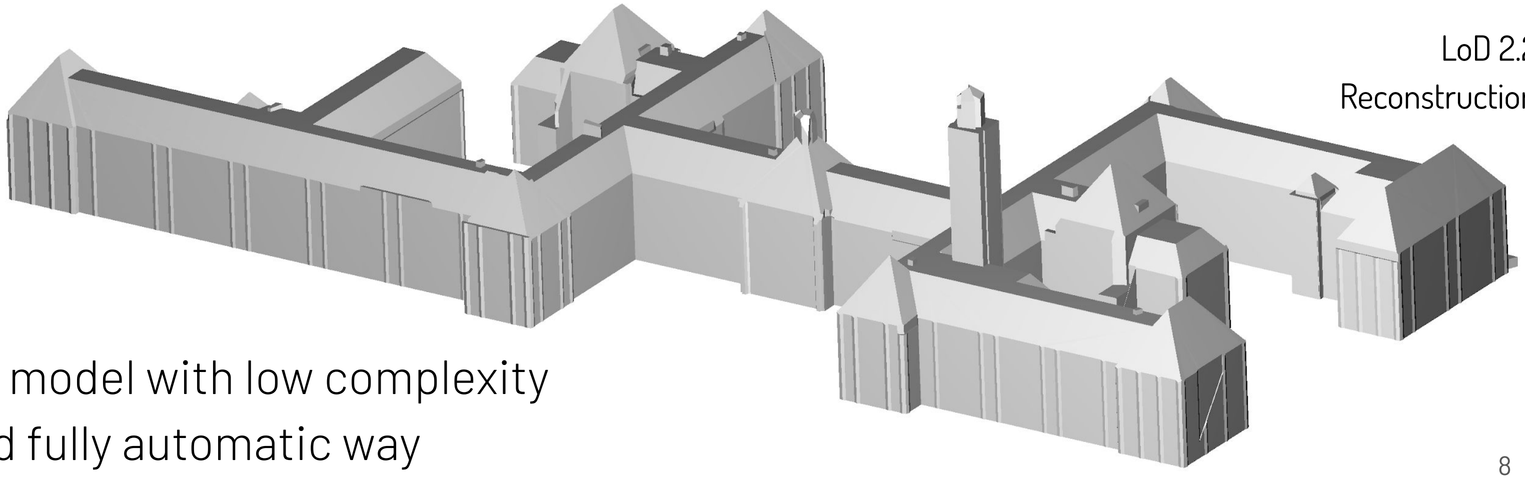
3DBAG/roofer

- What does it do?
  - Automatic LoD2 building reconstruction from aerial Lidar data.
- What is it?
  - Software library (API) with bindings for C++ and Python
  - CLI application
- Previously known as
  - Geoflow + gfp-building-reconstruction plugin (very complex to use)
- Current development status?
  - Near to first public release

Building reconstruction

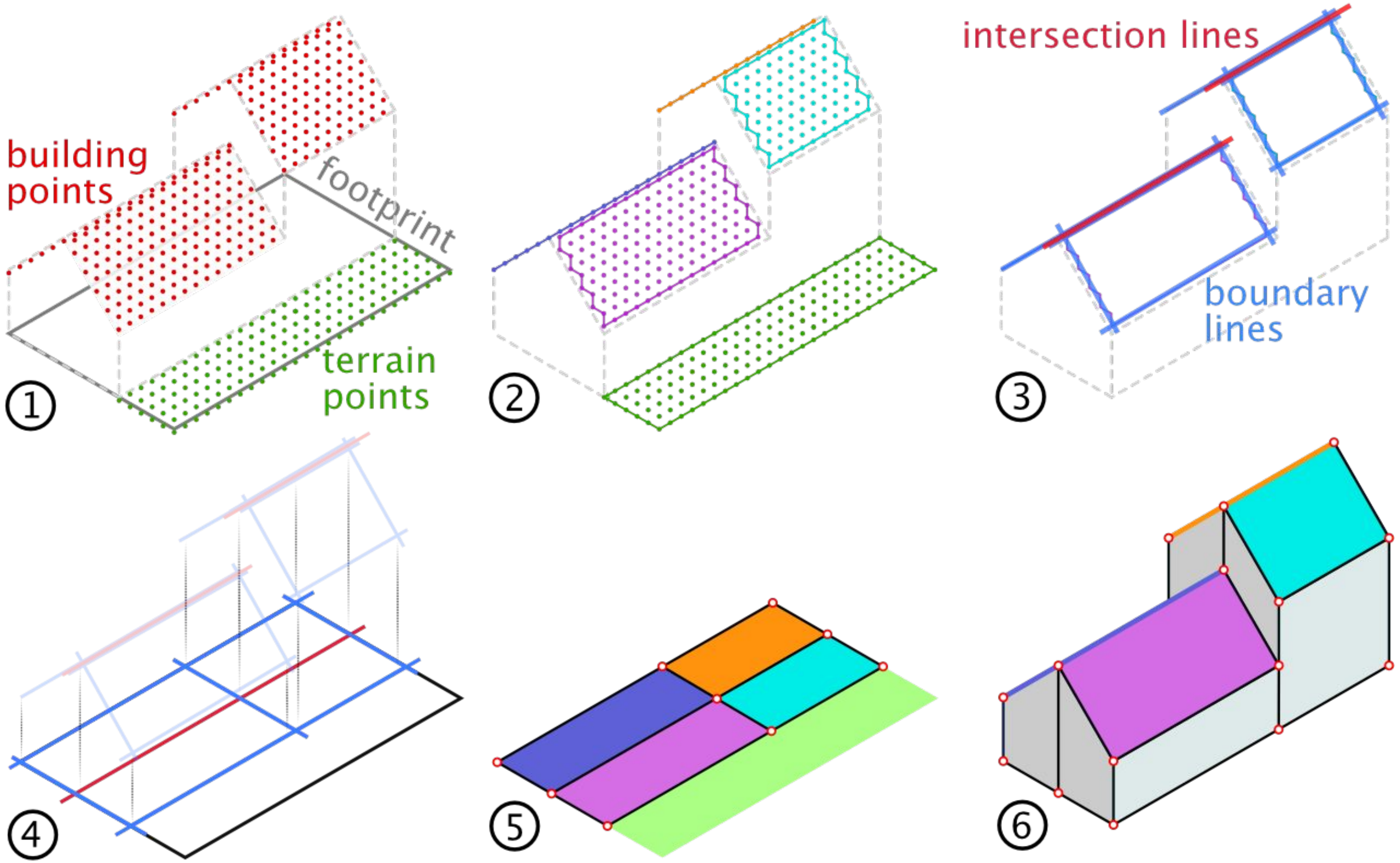

pointcloud


footprint polygon


LoD 2.2
Reconstruction

- Accurate 3D model with low complexity
- In robust and fully automatic way

Overview roofer reconstruction algorithm



Roofer API

3DBAG/roofer

Python example:

```
import rooferpy

# ... get the input pointcloud and footprint polygon for your building

# Set the reconstruction configuration
roofer_config = rooferpy.ReconstructionConfig()
roofer_config.complexity_factor = 0.7 # Change the default values if needed

# Reconstruct
print("Reconstructing building...")
meshes = rooferpy.reconstruct(points_roof, points_terrain, footprint, roofer_config)
```

`pip install roofer` is coming soon

Roofer API

3DBAG/roofer

C++ example:

```
#include <roofer/roofer.h>

// ... get the input pointcloud and footprint polygon for your building

auto meshes =
     roofer::reconstruct(points_roof, points_terrain, footprints.front(),
{.complexity_factor = 0.7});
```

Add to your project via CMake

Roofer CLI application

3DBAG/roofer

Designed to efficiently reconstruct large areas with many buildings

- Capable of handling large input datasets
- Using multithreading for efficient computation
- Outputs 3D models as CityJSONSequence

Usage:

```
roofer -c config.toml
```

```
[input.footprint]
path = "data/wippolder/wippolder.gpkg"
id_attribute = "identificatie"

[[input.pointclouds]]
name = "AHN3"
path = "data/wippolder/wippolder.las"

[reconstruction]
reconstruction_complexity = 0.7

[output]
path = 'output/wippolder'
```

3dbag-pipeline

Overview

- Implemented with the **Dagster framework**
- What it does:
  - downloads source data (AHN, BAG, TOP10NL)
  - preprocesses source data
  - extracts metadata
  - runs building reconstruction
  - runs extension modules
    - party walls calculation
    - nr. floor estimation
  - converts and packages output to various formats
  - deploys output to the webserver

Architecture overview

Mono-repo with ***subpackages:***

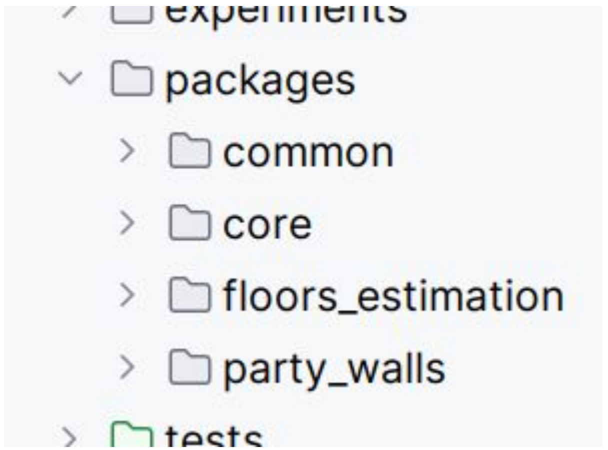
`common` package for functionality that is used by all workflow packages

Workflow packages contain the pipeline logic:

`core` The core 3DBAG data

`party_walls` Party walls calculation

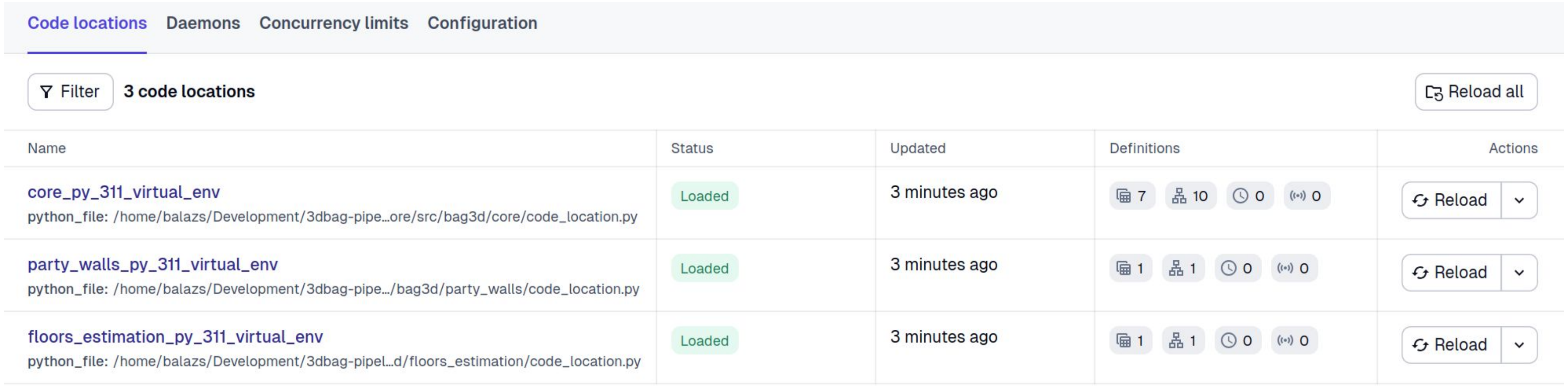
`floors_estimation` Nr. of floors estimation



Architecture overview

Each workflow package in an isolated virtual environment ("Code Location").

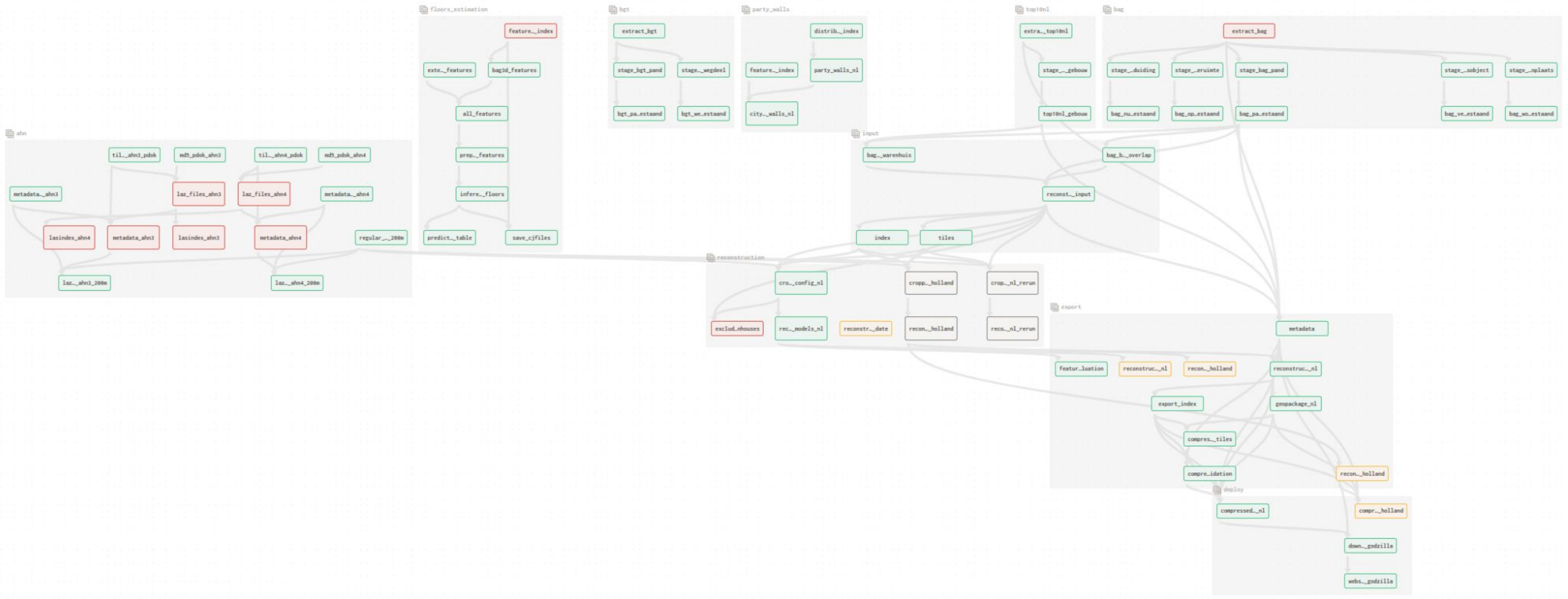
- + **Helps us avoid dependency hell**
- + **Dagster uses gRPC for communicating with the code locations**



3 code locations

Name	Status	Updated	Definitions	Actions
core_py_311_virtual_env python_file: /home/balazs/Development/3dbag-pipe…ore/src/bag3d/core/code_location.py	Loaded	3 minutes ago	7 10 0 0	Reload
party_walls_py_311_virtual_env python_file: /home/balazs/Development/3dbag-pipe…/bag3d/party_walls/code_location.py	Loaded	3 minutes ago	1 1 0 0	Reload
floors_estimation_py_311_virtual_env python_file: /home/balazs/Development/3dbag-pipel…d/floors_estimation/code_location.py	Loaded	3 minutes ago	1 1 0 0	Reload

Architecture overview

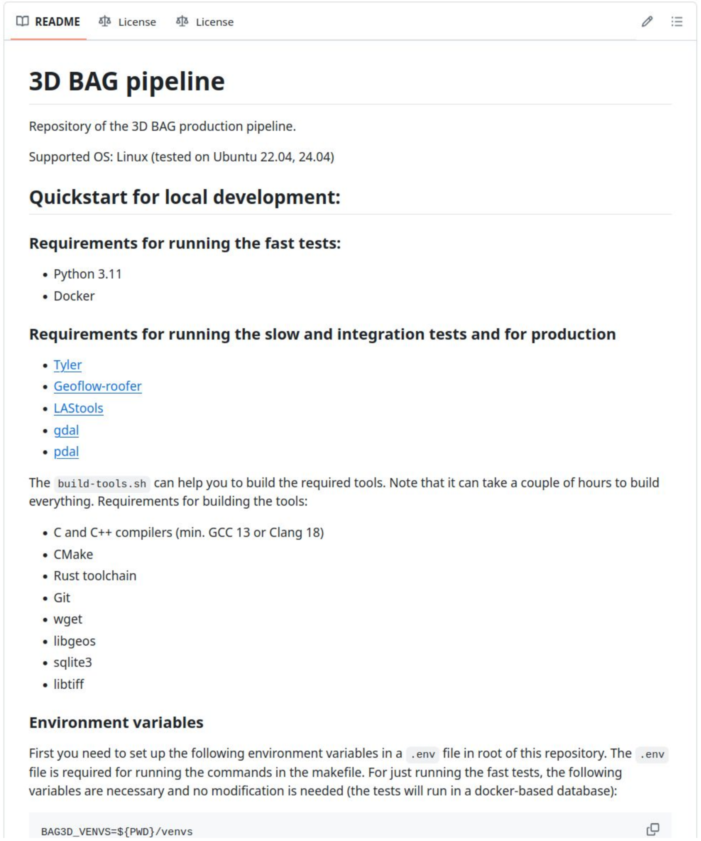


Requirements

3DBAG/3dbag-pipeline

1. Linux
2. Python 3.11
3. Docker
4. C & C++ compilers
5. (Well-configured) PostgreSQL:

shared_buffers = 24GB
max_parallel_workers = 24
max_connections = 150
effective_cache_size = 4GB
effective_io_concurrency = 100
maintenance_work_mem = 2GB



3D BAG pipeline

Repository of the 3D BAG production pipeline.

Supported OS: Linux (tested on Ubuntu 22.04, 24.04)

Quickstart for local development:

Requirements for running the fast tests:

- Python 3.11
- Docker

Requirements for running the slow and integration tests and for production

- Tyler
- Geoflow-roofer
- LAStools
- gdal
- pdal

The `build-tools.sh` can help you to build the required tools. Note that it can take a couple of hours to build everything. Requirements for building the tools:

- C and C++ compilers (min. GCC 13 or Clang 18)
- CMake
- Rust toolchain
- Git
- wget
- libgeos
- sqlite3
- libtiff

Environment variables

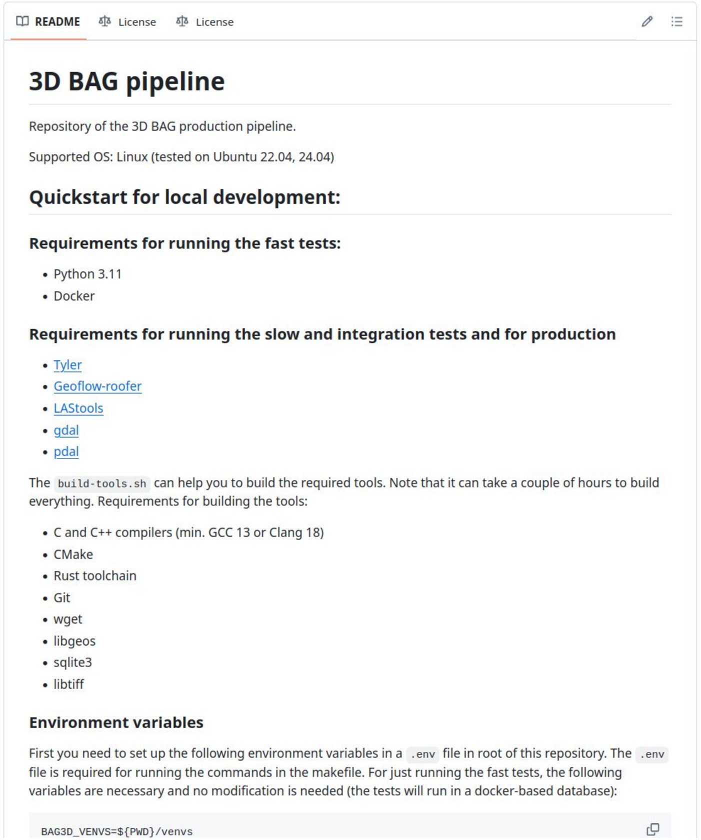
First you need to set up the following environment variables in a `.env` file in root of this repository. The `.env` file is required for running the commands in the makefile. For just running the fast tests, the following variables are necessary and no modification is needed (the tests will run in a docker-based database):

```
BAG3D_VENVS=${PWD}/venvs
```

Getting started

3DBAG/3dbag-pipeline

1. Pull the repo
2. Have Linux, Python 3.11, Docker
3. Set environment variables in `.env`
4. Run the unit tests
   a. `make venvs`
   b. `make download`
   c. `make build`
   d. `make run`
   e. `make test`
5. Try the integration tests
   a. `build-tools.sh`
   b. `make integration`



3D BAG pipeline

Repository of the 3D BAG production pipeline.

Supported OS: Linux (tested on Ubuntu 22.04, 24.04)

Quickstart for local development:

Requirements for running the fast tests:

- Python 3.11
- Docker

Requirements for running the slow and integration tests and for production

- Tyler
- Geoflow-roofer
- LAStools
- gdal
- pdal

The `build-tools.sh` can help you to build the required tools. Note that it can take a couple of hours to build everything. Requirements for building the tools:

- C and C++ compilers (min. GCC 13 or Clang 18)
- CMake
- Rust toolchain
- Git
- wget
- libgeos
- sqlite3
- libtiff

Environment variables

First you need to set up the following environment variables in a `.env` file in root of this repository. The `.env` file is required for running the commands in the makefile. For just running the fast tests, the following variables are necessary and no modification is needed (the tests will run in a docker-based database):

```
BAG3D_VENVS=${PWD}/venvs
```

Refactoring

Why?

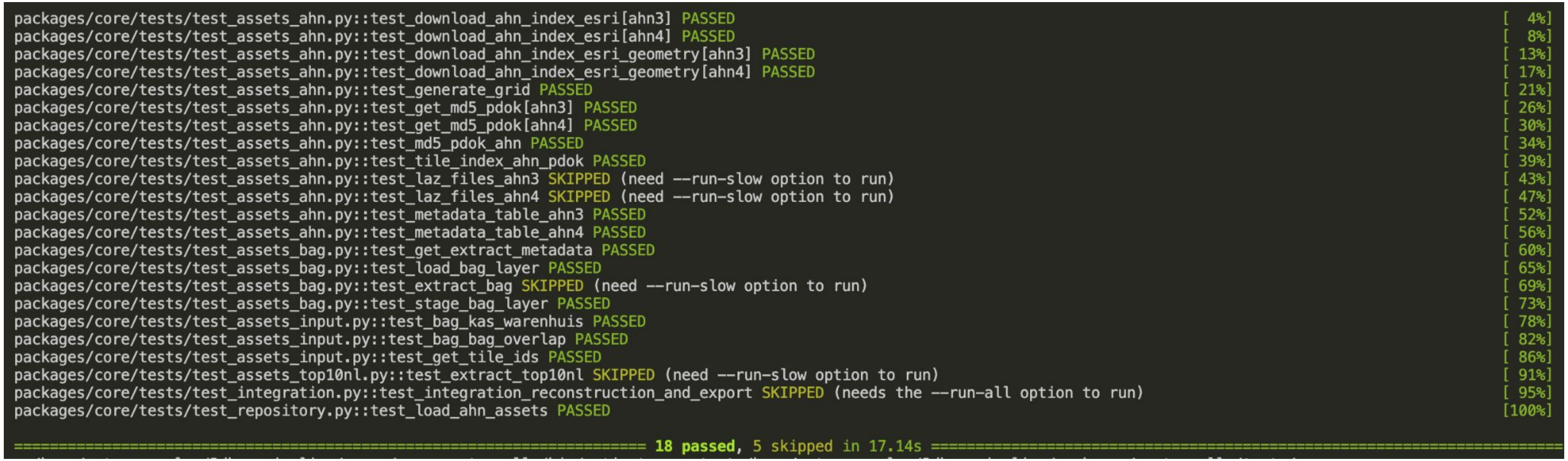
1. Improve readability & reusability for future users
2. Enhance maintainability & testability
3. Reduce complexity
4. Improve performance
5. Remove redundancy

What?

1. Added unit tests, created integration tests
2. Removed dead code
3. Rename variables and removed hardcoded paths
4. Formatted the code base
5. Clear docs
6. Adopted better practices

Testing

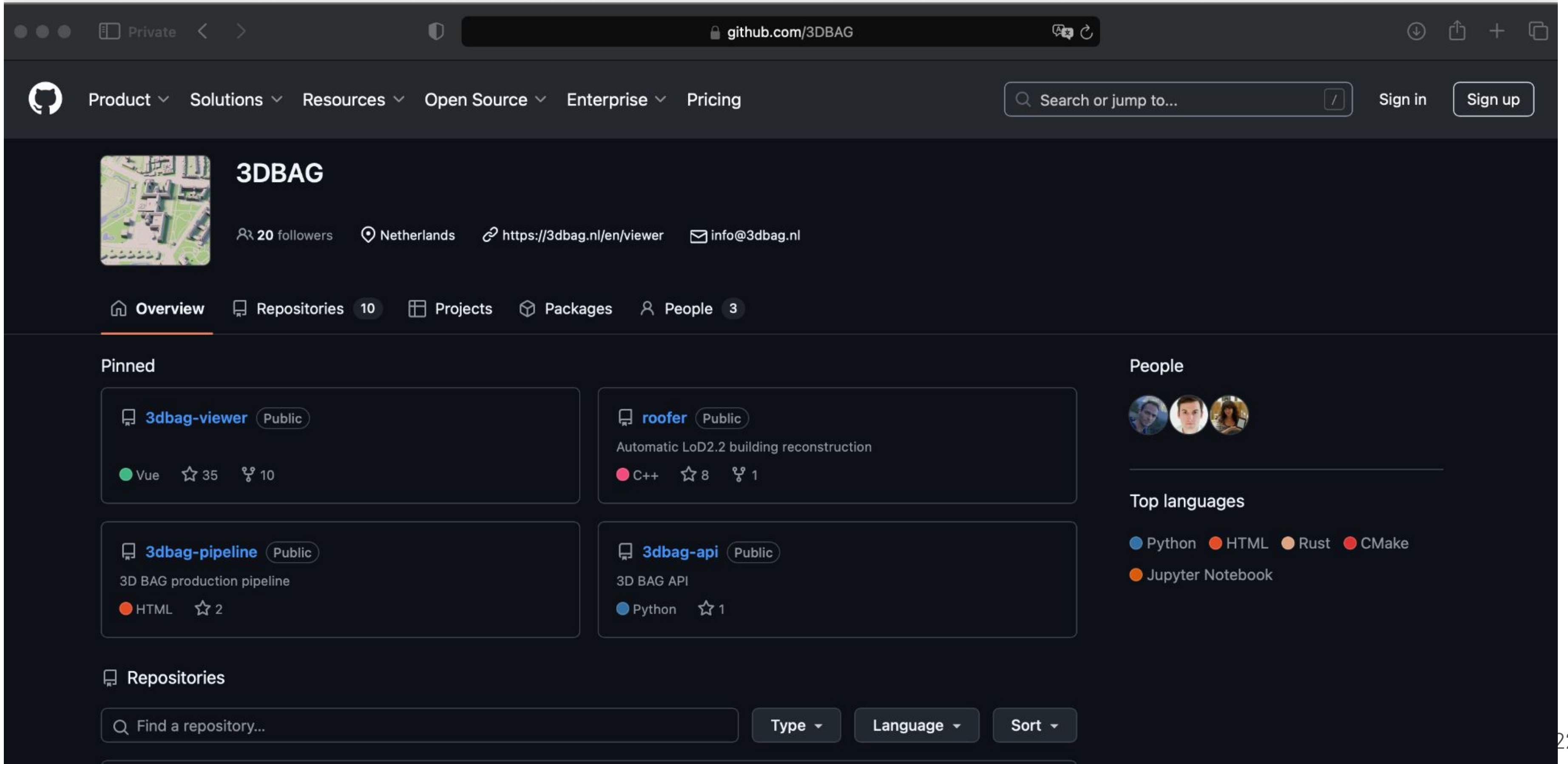
- ➢ Unit tests for individual functions
- ➢ Integration tests to test the Dagster pipeline (for a small region)
- ➢ Coverage Evaluation



```
packages/core/tests/test_assets_ahn.py::test_download_ahn_index_esri[ahn3] PASSED                                    [  4%]
packages/core/tests/test_assets_ahn.py::test_download_ahn_index_esri[ahn4] PASSED                                    [  8%]
packages/core/tests/test_assets_ahn.py::test_download_ahn_index_esri_geometry[ahn3] PASSED                           [ 13%]
packages/core/tests/test_assets_ahn.py::test_download_ahn_index_esri_geometry[ahn4] PASSED                           [ 17%]
packages/core/tests/test_assets_ahn.py::test_generate_grid PASSED                                                    [ 21%]
packages/core/tests/test_assets_ahn.py::test_get_md5_pdok[ahn3] PASSED                                               [ 26%]
packages/core/tests/test_assets_ahn.py::test_get_md5_pdok[ahn4] PASSED                                               [ 30%]
packages/core/tests/test_assets_ahn.py::test_md5_pdok_ahn PASSED                                                     [ 34%]
packages/core/tests/test_assets_ahn.py::test_tile_index_ahn_pdok PASSED                                              [ 39%]
packages/core/tests/test_assets_ahn.py::test_laz_files_ahn3 SKIPPED (need --run-slow option to run)                  [ 43%]
packages/core/tests/test_assets_ahn.py::test_laz_files_ahn4 SKIPPED (need --run-slow option to run)                  [ 47%]
packages/core/tests/test_assets_ahn.py::test_metadata_table_ahn3 PASSED                                              [ 52%]
packages/core/tests/test_assets_ahn.py::test_metadata_table_ahn4 PASSED                                              [ 56%]
packages/core/tests/test_assets_bag.py::test_get_extract_metadata PASSED                                             [ 60%]
packages/core/tests/test_assets_bag.py::test_load_bag_layer PASSED                                                   [ 65%]
packages/core/tests/test_assets_bag.py::test_extract_bag SKIPPED (need --run-slow option to run)                     [ 69%]
packages/core/tests/test_assets_bag.py::test_stage_bag_layer PASSED                                                  [ 73%]
packages/core/tests/test_assets_input.py::test_bag_kas_warenhuis PASSED                                              [ 78%]
packages/core/tests/test_assets_input.py::test_bag_bag_overlap PASSED                                                [ 82%]
packages/core/tests/test_assets_input.py::test_get_tile_ids PASSED                                                   [ 86%]
packages/core/tests/test_assets_top10nl.py::test_extract_top10nl SKIPPED (need --run-slow option to run)             [ 91%]
packages/core/tests/test_integration.py::test_integration_reconstruction_and_export SKIPPED (needs the --run-all option to run) [ 95%]
packages/core/tests/test_repository.py::test_load_ahn_assets PASSED                                                  [100%]

====================================================================== 18 passed, 5 skipped in 17.14s ======================================================================
```

Get our software at github.com/3DBAG



github.com/3DBAG

3DBAG

20 followers · Netherlands · https://3dbag.nl/en/viewer · info@3dbag.nl

Pinned

3dbag-viewer Public
● Vue ☆ 35 Forks 10

roofer Public
Automatic LoD2.2 building reconstruction
● C++ ☆ 8 Forks 1

3dbag-pipeline Public
3D BAG production pipeline
● HTML ☆ 2

3dbag-api Public
3D BAG API
● Python ☆ 1

People

Top languages
● Python ● HTML ● Rust ● CMake ● Jupyter Notebook

Repositories

3DBAG future developments

Organised through the *3DBAG innovation platform*

- 3DBAG **User meeting** in Amsterdam next week!
  - October 2 from 12:00 to 17:30
  - See banner on 3dbag.nl to sign up
- 3DBAG **Developers' meeting**
  - afternoon of November 13 (preliminary date)
  - for advanced 3DBAG users and developers
  - Sign up for newsletter for more information
- **New 3DBAG data release** this fall
  - With AHN5!
  - Using our new and updated software 🤞

Sign up for 3DBAG newsletter:

Thank you for your attention!

Balázs Dukai
Balazs.Dukai@3dgi.nl

Jantien Stoter
J.E.Stoter@tudelft.nl

Hugo Ledoux
H.Ledoux@tudelft.nl

Ravi Peters
Ravi.Peters@3dgi.nl

Gina Stavropoulou
G.Stavropoulou@tudelft.nl

Ivan Pađen
I.Paden@tudelft.nl

3DGI
3dgi.nl

TU Delft 3Dgeoinfo
3d.bk.tudelft.nl

This project has received funding from the European Research Council (ERC) under the Horizon Europe Research & Innovation Programme (grant agreement no. 101068452 3DBAG: detailed 3D Building models Automatically Generated for very large areas)