RASTER FLOOR PLAN → SEGMENTATION → VECTOR MODEL → EDITABLE 3D

Floor Plan *Editor*
From architectural images to editable 3D geometry.

A two-part system that turns any floor-plan image into a structured 3D model. A Python backend runs a pretrained CubiCasa5k hourglass network and a custom vectorizer; a React + Three.js editor lets the user refine the result and export it as **GLB** or **JSON**.

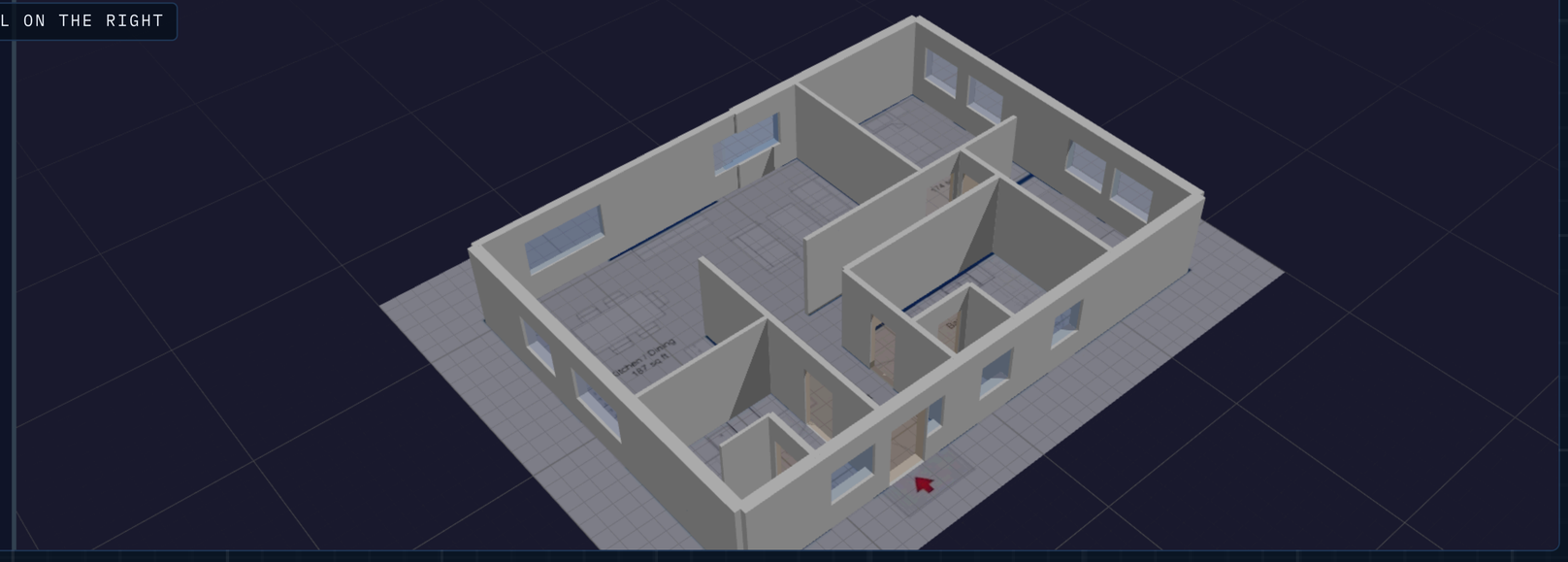
AUTHOR
Andre Ahuna - andre.ahuna@ut.ee

COURSE
MTAT.03.328, Computer Graphics Project

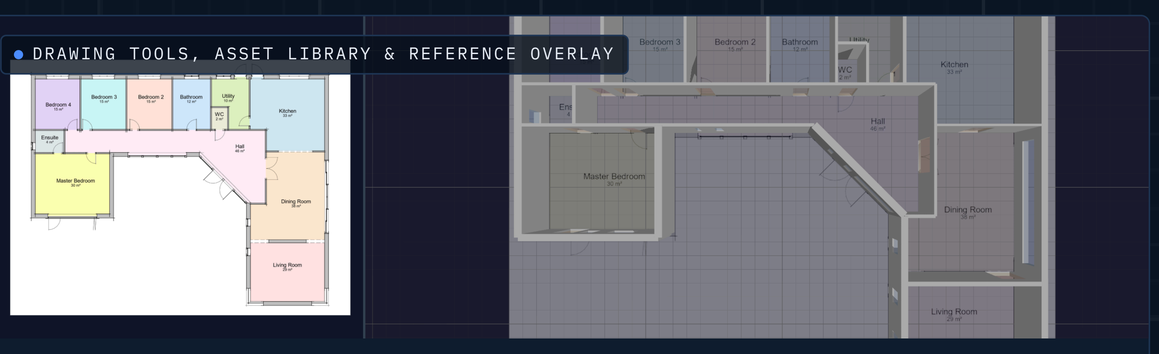
BACKEND STACK
Python / Flask / PyTorch / OpenCV

FRONTEND STACK
React / TypeScript / Three.js / Vite

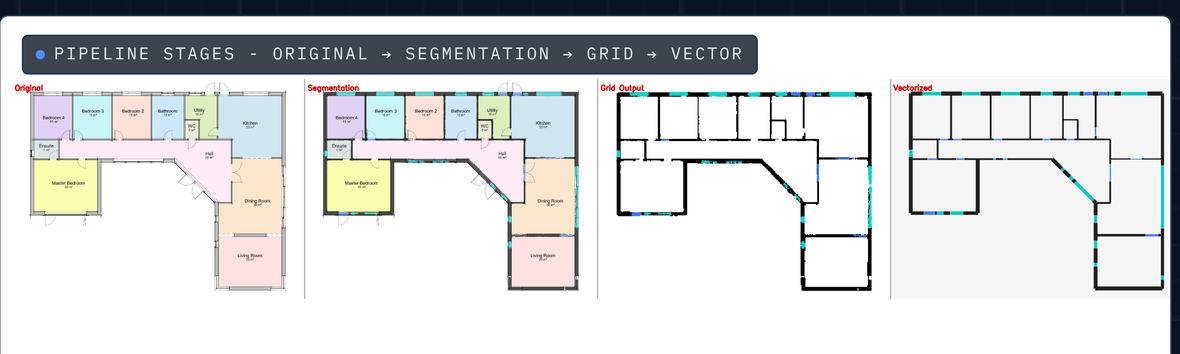
FLOOR PLAN EDITOR - 2D SOURCE ON THE LEFT, EXTRUDED 3D MODEL ON THE RIGHT



DRAWING TOOLS, ASSET LIBRARY & REFERENCE OVERLAY



PIPELINE STAGES - ORIGINAL → SEGMENTATION → GRID → VECTOR



5 classes
WALL / DOOR / WINDOW / STAIRS / BG

<10 sec
TYPICAL PROCESSING PER FLOORPLAN

Live edit
WALLS, DOORS, WINDOWS, STAIRS IN 3D

GLB / JSON
CLIENT-SIDE EXPORT FROM THE EDITOR

01 PROBLEM

Architects publish images. Renderers need structure.

Floor plans live as **raster images** - scans, listings, IKEA Home Planner exports. Any downstream visualization needs **structured geometry**: walls as segments, openings tied to walls, classes per pixel. Building that by hand is slow.

Pure rule-based vision struggles: hatching, dimensions, furniture and door arcs share the same low-level signal as walls. CubiCasa5K treats the problem as semantic segmentation followed by junction-based vectorization.

02 METHOD

CubiCasa5k inference, then junction vectorization.

The input is white-padded to a square and resized to a multiple of 32 (capped at 1024 px), then passed in a **single forward pass** through a CubiCasa5k stacked-hourglass network. The model outputs **21 junction heatmaps**, a 12-channel room map, and an 11-channel icon map; we keep five semantic classes - wall, door, window, stairs, background.

The **vectorizer** reads the junction heatmaps directly. Each junction (I / L / T / X) is matched against its neighbours along the predicted directions to recover axis-aligned wall segments; openings are then attached to the wall they sit on, with all coordinates remapped back into the original image.

03 EDITOR

Refine in the browser, export anywhere.

- → Draw / move / delete walls
- → Stair tool with run / rise gizmo
- → Live 2D / 3D preview
- → Undo / redo history
- → Doors & windows snap to walls
- → Reference image overlay
- → Grid snap & keyboard shortcuts
- → Export to **GLB** or **JSON**

The ML output is a **starting point, not a black box**. Every wall, opening and stair is editable in 3D; the final export runs client-side, so nothing leaves the browser after refinement.

04 VECTOR OUTPUT

Structured JSON, wall-anchored openings.

The vectorizer emits a structured floor-plan record: image-size header, a list of walls (each with start, end, thickness, polygon) and a list of openings keyed by wall_id. The editor extrudes each wall polygon into a 3D mesh and slices openings as boolean cuts. The final scene is exported client-side as a binary GLB or as the same structured JSON with user edits applied - round-trippable back into the editor.

```
"walls": [{
  "id": 0,    "start": [107.4, 78.8],
  "end": [1487.8, 78.8],    "thickness": 20.3
}],
"openings": [{
  "type": "door",    "wall_id": 0,
  "position": [412.0, 78.8],    "width": 85.0
}]
```

05 BACKEND

Flask service, one round-trip.

The Python backend wraps the model and vectorizer behind a single extraction endpoint. The frontend uploads an image and receives parsed vector data plus a rendered preview in one response.

```
POST /api/extract/ml/full
  file=@floorplan.png

# Response (abbrev.)
stats:        { walls: 42 }
vector_data:  { walls[], openings[] }
preview_url / json_url
```

→ SCAN FOR PROJECT INFO & LIVE DEMO

REFERENCES

[1] Kalervo, A. et al. *CubiCasa5K: A Dataset and an Improved Multi-Task Model for Floorplan Image Analysis.* SCIA, 2019.
[2] Liu, C., Wu, J., Furukawa, Y. *Raster-to-Vector: Revisiting Floorplan Transformation.* ICCV, 2017.
[3] Newell, A., Yang, K., Deng, J. *Stacked Hourglass Networks for Human Pose Estimation.* ECCV, 2016.
[4] Liu, C., Wu, J., Furukawa, Y. *FloorNet: A Unified Framework for Floorplan Reconstruction from 3D Scans.* ECCV, 2018.
[5]

PROJECT & CONTACT

COURSE	MTAT.03.328, Computer Graphics Project
INSTITUTE	University of Tartu, Institute of Computer Science
AUTHOR	Andre Ahuna - andre.ahuna@ut.ee
STACK	Flask / PyTorch / CubiCasa5k / React / Three.js