Machine Learning Engineer Test:
Computer Vision and Object Detection

Objective

This test aims to assess your skills in computer vision and object detection, with a specific focus on detecting room walls and identifying rooms in architectural blueprints or pre-construction plans.

This test evaluates your practical skills in applying advanced computer vision techniques to a specialized domain and your ability to integrate machine learning models into a simple API server for real-world applications.

Choose one of the visual tasks, one of the text extraction tasks, and the API Server task. We encourage you to submit your tests even if you can't complete all tasks.

Good luck!

Duration

2 hours

Tasks

Visual tasks

1) Room Wall Detection

You will receive a dataset containing images of architectural blueprints or pre-construction plans. Develop a model or algorithm to identify walls within the blueprints, annotating each wall and ensuring that the boundaries are clearly defined and each wall is distinctly marked.

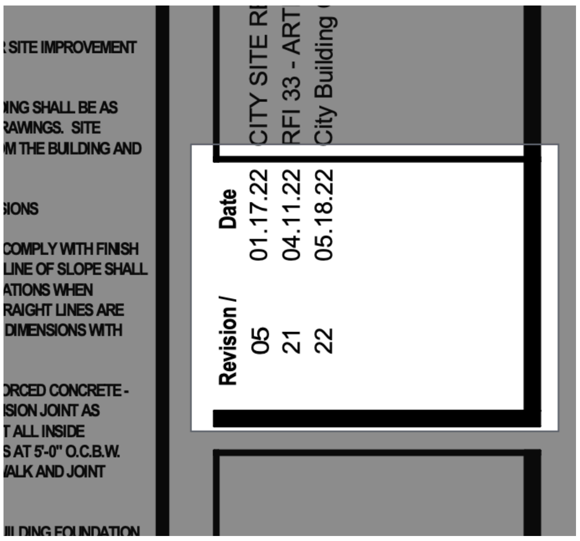
2) Room Detection

You will receive a dataset containing images of architectural blueprints or pre-construction plans. Develop a model or algorithm to identify distinct rooms within the blueprints, annotating each room and ensuring that the boundaries are clearly defined and each room is distinctly marked.

Text extraction tasks

1) Extract sheet number, page name and revision

All document pages have a sheet number, a page name and optionally a revision that should be extracted and presented in a JSON or similar format. See examples for more information.



SITE IMPROVEMENT

ING SHALL BE AS
RAWINGS. SITE
M THE BUILDING AND

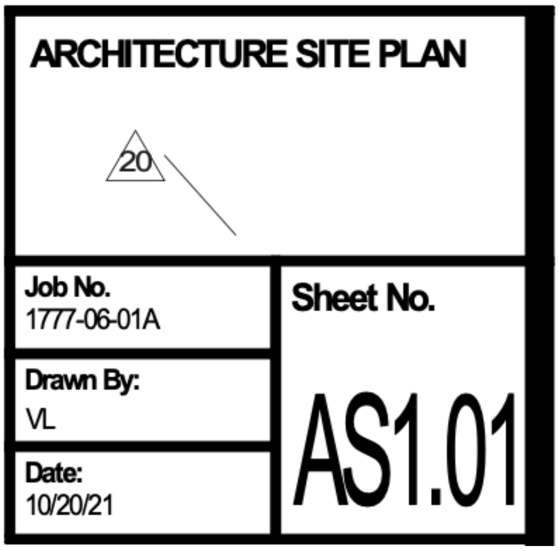
IONS

COMPLY WITH FINISH
LINE OF SLOPE SHALL
ATIONS WHEN
RAIGHT LINES ARE
DIMENSIONS WITH

ORCED CONCRETE -
ISION JOINT AS
T ALL INSIDE
S AT 5'-0" O.C.B.W.
ALK AND JOINT

ILDING FOUNDATION

Revision /	Date	
05	01.17.22	CITY SITE R
21	04.11.22	RFI 33 - ART
22	05.18.22	City Building



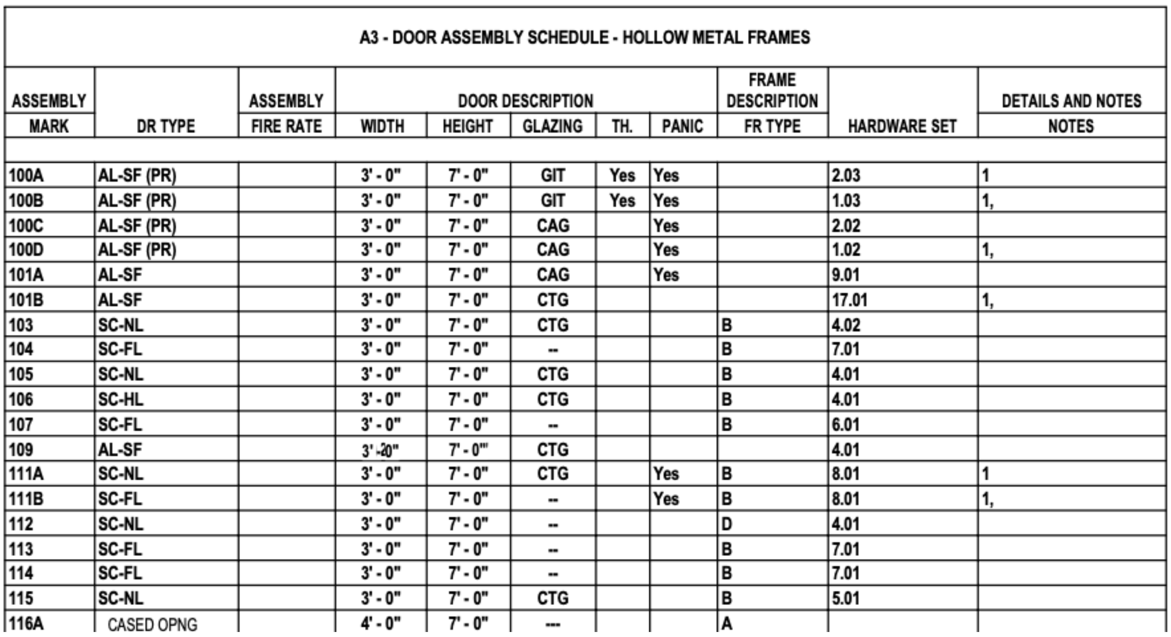
ARCHITECTURE SITE PLAN

20

Job No. 1777-06-01A	Sheet No. AS1.01
Drawn By: VL	
Date: 10/20/21	

2) Extract tables (for example the following schedule of materials)

Some pages have tables, which describe materials, summaries of the content, or even instructions. Extract the table information and present it in a reasonable format. See examples for more information.



A3 - DOOR ASSEMBLY SCHEDULE - HOLLOW METAL FRAMES

ASSEMBLY		ASSEMBLY	DOOR DESCRIPTION					FRAME DESCRIPTION		DETAILS AND NOTES
MARK	DR TYPE	FIRE RATE	WIDTH	HEIGHT	GLAZING	TH.	PANIC	FR TYPE	HARDWARE SET	NOTES
100A	AL-SF (PR)		3' - 0"	7' - 0"	GIT	Yes	Yes		2.03	1
100B	AL-SF (PR)		3' - 0"	7' - 0"	GIT	Yes	Yes		1.03	1,
100C	AL-SF (PR)		3' - 0"	7' - 0"	CAG		Yes		2.02	
100D	AL-SF (PR)		3' - 0"	7' - 0"	CAG		Yes		1.02	1,
101A	AL-SF		3' - 0"	7' - 0"	CAG		Yes		9.01	
101B	AL-SF		3' - 0"	7' - 0"	CTG				17.01	1,
103	SC-NL		3' - 0"	7' - 0"	CTG			B	4.02	
104	SC-FL		3' - 0"	7' - 0"	--			B	7.01	
105	SC-NL		3' - 0"	7' - 0"	CTG			B	4.01	
106	SC-HL		3' - 0"	7' - 0"	CTG			B	4.01	
107	SC-FL		3' - 0"	7' - 0"	--			B	6.01	
109	AL-SF		3' - 0"	7' - 0"	CTG				4.01	
111A	SC-NL		3' - 0"	7' - 0"	CTG		Yes	B	8.01	1
111B	SC-FL		3' - 0"	7' - 0"	--		Yes	B	8.01	1,
112	SC-NL		3' - 0"	7' - 0"	--			D	4.01	
113	SC-FL		3' - 0"	7' - 0"	--			B	7.01	
114	SC-FL		3' - 0"	7' - 0"	--			B	7.01	
115	SC-NL		3' - 0"	7' - 0"	CTG			B	5.01	
116A	CASED OPNG		4' - 0"	7' - 0"	---			A		

Inference API Server task

Develop a basic API server that can run inference using your models or algorithms. The API should accept an image as input and return the identified walls, rooms or extracted text.

Each request should receive the inference type to run accordingly a query parameter named “type”, which can assume any of the following values:

- walls
- rooms
- page_info
- tables

Recommended Libraries and Tools

PyTorch	Useful for developing machine learning models.
OpenCV	Essential for image processing tasks.
Scikit-image	For image processing and computer vision tasks.
Scikit-learn	For implementing machine learning algorithms.
Matplotlib/Seaborn	For visualizing the results.
Pandas/Numpy	For data manipulation and numerical operations.
Flask/FastAPI	For creating the API server to run inference.

Project Submission

Fork the provided base repo with test data and example requests, develop your project, and give access to the following GitHub users:

- vhaine-tb
- omasri-tb
- alexwine36

Sample project to Fork

https://github.com/TrueBuiltSoftware/ml-eng-test

Format of Submission

- Include a README file in your repository with detailed instructions on how to set up and run the API server.
- Use docker to containerize your application, ensuring consistency across different environments.
- Make sure to add the trained model and any additional files/libraries required to run the code.
- Provide a sample CURL/fetch/script to test the API inference
- Ensure your code is well-commented and documented to explain your approach and methodology.
- Include any jupyter notebooks with the necessary files inside your GitHub repository submission.

Evaluation Criteria

- **Accuracy of Wall Detection:** The effectiveness of your algorithm in accurately detecting walls within the blueprints.
- **Accuracy of Room Detection:** The effectiveness of your algorithm in accurately detecting rooms within the blueprints.
- **API Server Functionality:** The API server should be functional, easy to run, and capable of performing inference as expected.
- **Code Quality:** Clarity, organization, documentation, and adherence to best practices in coding and machine learning.
- **Result Presentation:** Clear visualization and annotation of detected walls, rooms, and a concise report of your findings.

Examples

- Wall detection API response

JavaScript
```javascript
{
  "type": "walls",
  "imageId": "some_image_id",
  "detectionResults": {
    "walls": [
      {
        "wallId": "wall_1",
        "position": {
          "start": { "x": 10, "y": 15 },
          "end": { "x": 110, "y": 15 }
        },
        "confidence": 0.95
      },
      {
        "wallId": "wall_2",
        "position": {
          "start": { "x": 20, "y": 30 },
          "end": { "x": 120, "y": 30 }
        },
        "confidence": 0.9
      },
      // Additional walls...
    ]
  }
}
```

- Room Detection API response

JavaScript
```
{
  "type": "rooms",
  "imageId": "some_image_id",
  "detectionResults": {
    "rooms": [
      {
        "roomId": "room_1",
        "vertices": [
          { "x": 10, "y": 10 },
          { "x": 20, "y": 10 },
          { "x": 20, "y": 20 },
          { "x": 10, "y": 20 }
        ],
        "confidence": 0.95
      },
      {
        "roomId": "room_2",
        "vertices": [
          { "x": 30, "y": 30 },
          { "x": 40, "y": 30 },
          { "x": 40, "y": 40 },
          { "x": 30, "y": 40 },
          { "x": 25, "y": 35 }
        ],
        "confidence": 0.9
      },
      // Additional rooms...
    ]
  }
}
```

- Extract sheet number, page name and revision

```javascript
{
  "type": "page_info",
  "imageId": "some_image_id",
  "detectionResults": {
    "sheet_number": "A3.01",
    "sheet_name": "DOOR SCHEDULES, DOOR AND FRAME TYPES",
    "revision": { // If present
      "number": 4,
      "date": "01/28/22"
    }
  }
}
```

- Extract tables

```javascript
{
  "type": "schedule_of_materials",
  "imageId": "some_image_id",
  "detectionResults": [
    {
      "tableName": "A3 - DOOR ASSEMBLY SCHEDULE [...]",
      "columns": [
        {
          "header": ["ASSEMBLY", "MARK"],
          "rows": ["100A", "100B", "100C"] // rest of the rows
        },
        // rest of the columns
      ]
    },
    // other tables
  ]
}
```