Task description:

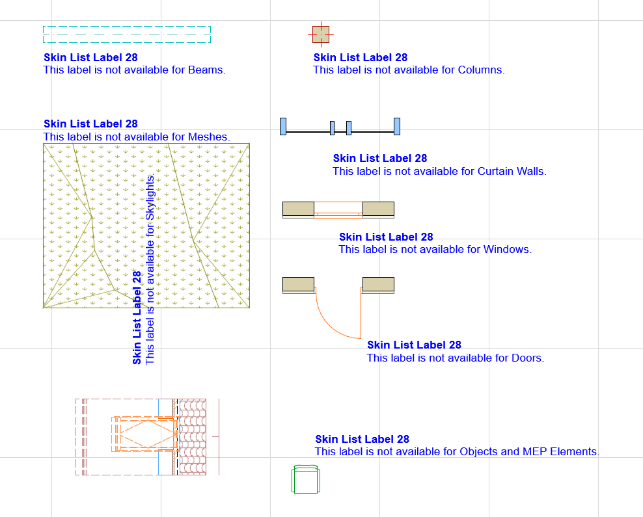
Description of the implemented component:

The core algorithm of my implementation is the latest version of Tesseract OCR engine. It consists of a line finding algorithm and an LSTM based text classifier. I chose this algorithm since it is considered one of the best performing OCR model with a relatively low computational complexity, furthermore it is easy to use in the implementation. Tesseract generally performs well on images containing printed texts, but its efficiency significantly drops when there are texts with various orientation on one image or the text-background color contrast is not high enough or the image quality is bad (The provided test set contains plenty of these cases). Therefore simply applying Tesseract would not bring the desired results, I had to extend it with further steps. A high-level description of the steps carried out by the implemented component can be seen on the flowchart below:

A képen szöveg, képernyőkép, diagram, Betűtípus látható

Automatikusan generált leírásFor each input image the component perform preprocessing steps. I chose these steps based on the following Tesseract documentation: <https://tesseract-ocr.github.io/tessdoc/ImproveQuality>. Since Tesseract prefers images with high resolution, an upscaling operation is performed based on a parameter given by the user. This parameter should be determined based on the resolution of the avaliable images.

In case of images with bright background upscaling is followed by denoising (with OpenCV’s **fastNlMeansDenoisingColored** algorithm), converting to grayscale and OpenCV thresholding (either OTSU or Adaptive). The effect of these steps can be seen on the example below:

 A képen szöveg, képernyőkép, diagram látható

Automatikusan generált leírás Before preprocessing After preprocessing

Note that in case of images with low text-background contrast Adaptive threshold performs better than OTSU, so I recommend using that one.

Since Tesseract is not able to efficiently process images containing texts with various orientation (and the test set contains many of these), I added the following steps:

1. The user can define a range of angles in degrees in the following format: (from, to, step\_size).
2. For each angle in range the component performs the follwing operations:
   1. Rotates the image by the current angle
   2. Applies Tesseract ont he rotated image
   3. Collects the prediction results (above a given confidence threshold) for all angles
   4. Transforms the results back to the original (not rotated) image coordinate system
3. Non-Maximal Supression is performed on all rectangles in order to delete double detections caused by processing the image (and detecting the same text) in different angles

Evaluation

Since the test set is unlabeled

Output

* Visualization
* Text extraction

**Setup guide:**

Required python modules:

* pytesseract==0.3.10
* tqdm==4.62.2
* numpy==1.23.5
* opencv-python==4.7.0.68
* matplotlib==3.4.3
* nms==0.1.6

The required packages are also included in the attached **requirements.txt.**

Additional steps when using the component on **Windows**:

1. the Tesseract engine should be installed. It can be downloaded from the following link: <https://digi.bib.uni-mannheim.de/tesseract/tesseract-ocr-w64-setup-5.3.3.20231005.exe>
2. The path to tesseract.exe should be added to --tesseract\_exe\_path argument when running ocr\_script.py

After setting up the environment the component can be used by running **ocr\_script.py**

**User manual:**

The ocr\_script.py has the following arguments:

--image\_folder\_path: (string) Path to the folder containing the input images. It is a required argument

--tesseract\_exe\_path: (string) Path to the tesseract engine. It is required when using i ton Windows.

--image\_upscale\_factor: (int) If input images have low resolution, this should be set to an integer value higher than 1 in order to upscale the images. Default 2

--apply\_denoising: (boolean) When True, OpenCV denoising algorithm will be applied on the images

--thresholding\_method: (string) Defines which thresholding algorithm will be applied on the images. Default None

--confidence\_threshold: (int) Predictions above this threshold (in percentage) will be saved and forwarded to NMS. Default 80

--rotation\_range: (list) Range of image rotation angles in format of: from\_angle to\_angle step\_size

--text\_output\_folder\_path: (string) Path to the folder where extracted text will be saved in form of .txt files. Default

--save\_visualizations: (string) When added, visualizations containing the highlighted, recognized text will be saved into visualization\_folder\_path

--visualization\_folder\_path: (string) Path to the folder where visualizations (if desired) will be saved

Example command:

python ocr\_script.py --image\_folder\_path testfiles --tesseract\_exe\_path "C:\\Program Files\\Tesseract-OCR\\tesseract.exe" --confidence\_threshold 80 --rotation\_range -45 45 5 --apply\_denoising --thresholding\_method adaptive --image\_upscale\_factor 2 --save\_visualizations