Deliverables: Instructions

I've put together a detailed guide on how I set up and ran the program that extracts, processes, and counts icons from the PDFs. I'll walk through each part carefully, so even if you're not familiar with coding or this environment, you'll know exactly what to do. This is based on my experience of getting everything working properly in Google Colab, making sure the program runs smoothly and efficiently.

Step 1: Setting Up the Environment in Google Colab

1. **Open Google Colab:**

I use Google Colab because it's a free, online platform that makes it easy to write and run Python code without needing to install anything on my local computer. It's also great because it has many libraries pre-installed, which saves me a lot of setup time.

To get started, I go to Google Colab and create a new notebook. This gives me a blank space to write my code and see the output immediately.

2. **Mounting My Google Drive:**

Since the PDFs are stored in Google Drive, I need to connect my Google Drive account to Colab. This allows the program to access and read files directly from my Drive.

I run this code:

```
from google.colab import drive

drive.mount('/content/drive')
```

When I run this, a link appears. I click on it, and it prompts me to log in to my Google account. Once logged in, it gives me a code that I copy and paste back into Colab. This gives Colab permission to access my Google Drive so the program can use the PDFs directly.

Step 2: Installing the Necessary Libraries

1. **Installing Libraries**:

Although Colab has many libraries pre-installed, I still need to make sure pdf2image and other tools are properly set up. These libraries help convert PDF files into images and process those images.

I run this in a new cell:

```
!apt-get install -y poppler-utils

!pip install pdf2image opencv-python-headless numpy matplotlib
```

This command installs poppler-utils, which is a tool necessary for reading PDF files, and other Python libraries (pdf2image, opencv-python-headless, numpy, and matplotlib) that the program uses for image processing and visualizations.

Step 3: Setting Up the PDF File Paths

1. **Finding the File Paths**:

   I navigate to my Google Drive, find the folder where I've stored the Schedule and Room Plan PDFs, and copy their paths. To do this:

   - I right-click each file, select "Get link," and make sure the sharing settings allow Colab to access it.
   - The file paths look something like
   - **/content/drive/My Drive/DataSet_Intern/DataSet/Schedule.pdf**
   - I note these paths down.

2. **Updating the Code with My File Paths**:

   - I updated the file paths in the code so that it knows where to find the PDFs

     schedule_pdf_path = '/content/drive/My Drive/DataSet_Intern/DataSet/Schedule.pdf'

     room_plan_pdf_path = '/content/drive/My Drive/DataSet_Intern/DataSet/RoomPlan.pdf'

It's important to make sure these paths are exactly correct, otherwise the program won't be able to find the files, and it will give an error.

Step 4: Writing and Running the Program

1. **Writing the Code**:

I carefully write the code that extracts icons, processes them, and counts how many times each icon from the Schedule PDF appears in the Room Plan PDF. This involves using functions like extract_icons_from_pdf to extract icons from each PDF, preprocess_special_icon to process and prepare the icons for analysis, and count_matching_icons to count matches accurately.

2. **Executing the Code**:

I run the code by clicking the "Run" button (the play icon) next to the code cell or pressing Shift + Enter. This initiates the program, where Colab:

- Reads the Schedule and Room Plan PDFs.
- Extracts the icons from each PDF.
- Applies various processing techniques (like thresholding and edge detection) to prepare the icons for comparison.
- Matches icons from the Schedule PDF against those in the Room Plan PDF and counts the occurrences.

The output appears below the cell, showing the count for each icon and whether it found matches in the Room Plan.

Step 5: Viewing and Interpreting the Output

1. **Checking the Icon Counts**:

   The program outputs results like this:

   Icon 0: 2 occurrences in the room plan.

   Icon 1: 3 occurrences in the room plan.

   Each line provides a summary of how many times an icon from the Schedule PDF was found in the Room Plan PDF. This helps me verify if the program is accurately detecting and counting the icons.

2. **Viewing the Visualizations**:

   For specific icons that I'm interested in (or that might be problematic), the program also visualizes:

   - The original version of the icon as extracted from the PDF.
   - The thresholded version, highlighting the edges.
   - The version with edge detection applied.
   - The morphology-processed version showing the icon's shape.

These visualizations help me understand how well the program processes each icon and where I might need to make adjustments.

Step 6: Making Adjustments

1. **Adjusting the Matching Threshold**:

If I notice that the program isn't detecting icons accurately, I adjust the threshold in the count_matching_icons function. The threshold controls the sensitivity of the icon matching.

By default, it's set to 0.8:

if max_val > 0.8:

I can change this value to be more or less strict, depending on the results I'm seeing. If icons are too similar, lowering the threshold might help the program differentiate them better.

2. **Visualizing Additional Icons**:

If I want to explore how other icons are being processed, I update the problematic_icon_ids list in the code. This allows me to visualize any icon I'm curious about, so I can observe and refine the process if needed. By following these steps, I successfully set up the environment, wrote and executed the program, and customized the process to match and visualize icons from the PDFs.