(Fullstack) Realtime Image Streaming using Flask

In below we provide you with several tasks. You will be evaluated by the quality of your work and the clarity of your code.

You will have 2 weeks to finish your tasks. In case you want more time, please contact us for further discussion.

▼ Initial considerations

- This challenging project is designed to measure your technical and problem-solving skills;
- Maintain your code clean, readable, and easy to understand;
- Everything that is not obvious must be explained;
    - The format is up to you: You can write a report that summarizes all your challenges, record videos, take screenshots, etc…
    - **Everything that can improve the evaluator's understanding of your work the better.**
- Provide an environment for running your code:
    - **The evaluator should have as less as work as possible to run your code;**
    - Document your development environment as much as possible: OS version, kernel version (Linux), library versions, etc..
    - If you use exclusively Python, provide a requirements.txt so the evaluator can create a virtual environment;
    - If you use any other language, please create and provide a docker file for running your code.
- Your code should be generic and reusable:
    - Avoid hard-coding things;
    - Provide ways for configuring your code behavior: configuration files, command line arguments, etc…

    - Make your code as much generic as possible and can accept different inputs from the ones you used by default (different URDF files, different robots etc…).
- Read all the challenge steps before starting:
    - These challenges were designed so you reuse the functions you make in one step in the following steps.
    - So, read all the requirements before starting coding and doing your research.

Task1: Capture Webcam Image Data by OpenCV

Please use the Python OpenCV library to stream and capture images from a webcam, which can be the webcam of your laptop. Your program should be able to call the webcam and capture images.

Task2: Stream Image Data In Web Browser GUI using Flask

Please reuse the Python code from Task 1 and design a web GUI together with the Flask framework to continuously visualize the image data in the web browser. Use a build management system to automate the compilation of your source code and write a script to launch your application. Your application needs to stream data from a webcam and continuously visualize the image data in an image viewer that you design in the browser. The web browser need to be able to accessed through chrome or firework.

Task3: Improve the UI/UX

Improve the web GUI design that you made in Task 2, and add buttons to start and stop image data streaming. If possible, add a button and a mode to trigger single-image capture and visualization. For UX, think of reducing the delay when visualizing the image data in the web browser continuously. Also, if possible, improve the CSS to adaptively enlarge or shrink the components in the web GUI when the browser is resized.

Task4: Add Camera Parameters Adjustment DOM

Modify the HTML, JavaScript, and Python code to add dragging sliders and buttons in the web GUI for adjusting webcam parameters such as exposure and gain. The

Python backend logic needs to receive the real-time updated parameters from the frontend and update the webcam parameters. Then stream the latest camera images and visualize them in the web GUI. It would be nice to have an information box in the GUI to show the real-time camera parameters (exposure, image size, capturing timing stamp, etc.).

Task5: Add Image Operation Functionalities

Modify the HTML, JavaScript, and CSS to add dragging sliders and buttons in the web GUI for adjusting real-time operation of the viewer of the streaming image, such as resizing, zooming in/out, and even drag and moving the location of the viewer of the streaming windows.

Task6: Simple Image Processing on Captured Images

Add a button with a simple image processing utility in the designed web GUI. When users click that button, an image-processing Python function will be executed on the captured images in the backend, and all the captured images will be operated with the indicated image processing. Here are the several image processing operations you can choose: image white balance, image color-to-gray transfer, and canny edge map computation. When users click the same button again, the image processing operation will be canceled and the captured images will be shown as the original captured RGB image.

Task7: Add Another View for Image Data Streaming In Parallel

Add another view in the same browser web-GUI to proceed with the same image data streaming in parallel. Please take care of the appearance also and the CSS style when you arrange the two streaming windows, as well as the dragging buttons.

Task8: Wrap Up the Program Inside a Docker

Please wrap up the program and prepare a docker to make/build from the docker file to generate a docker that can be executed to launch the web GUI.

Expected Output

1. your ideas for handling each task

2. source code with readme instruction
3. documentation that explains the code (which can comment inside the source code files)
4. some simple slides or docs to explain each task

Additional Information

If you have any questions about the tasks, please contact jinze@connected-robotics.com for more information