Panorama

Description

The goal of the project is to build a panorama from two images. The user should click 4 or more corresponding points in the left and right images. After the points are obtained, the program computes the homography, warps the left image into the right image, and produces the panorama.

Setup and requirements

The project was developed on Windows 11 using Visual Studio Code with the CMake Tools extension.
To run the program, the Imagine++ library must be installed and accessible.

Compilation

Using the command line
Run the `cmake --build .` command.

Using VS Code

1. Open the project folder in Visual Studio Code.
2. The CMake Tools extension will automatically configure the project.
3. Press the Build button (bottom toolbar) to compile.

Running the Program

Using the command line
Run `.\build\Release\Panorama.exe` (Windows) or `./build/Panorama.exe` (Linux/Mac) command.

Using VS Code
Run the built target by pressing the Run button (bottom toolbar).
The program will load the default images - *image0006.jpg* and *image0007.jpg*. If you want to use your own images:

1. Place them in the same directory as the default images.
2. Change the file paths at the beginning of *main()* in *Panorama.cpp* (variables s1 and s2).
3. Rebuild and run the program.

Use case scenario

In this section, we show how the program works with example screenshots.
1. When the program is run, two windows open showing the input images (you can move the windows left and right to make point selection easiest). In the console, the following message is displayed: *“Click on images w1 and w2. Click at least 4 points on both. Click right click to stop”.*

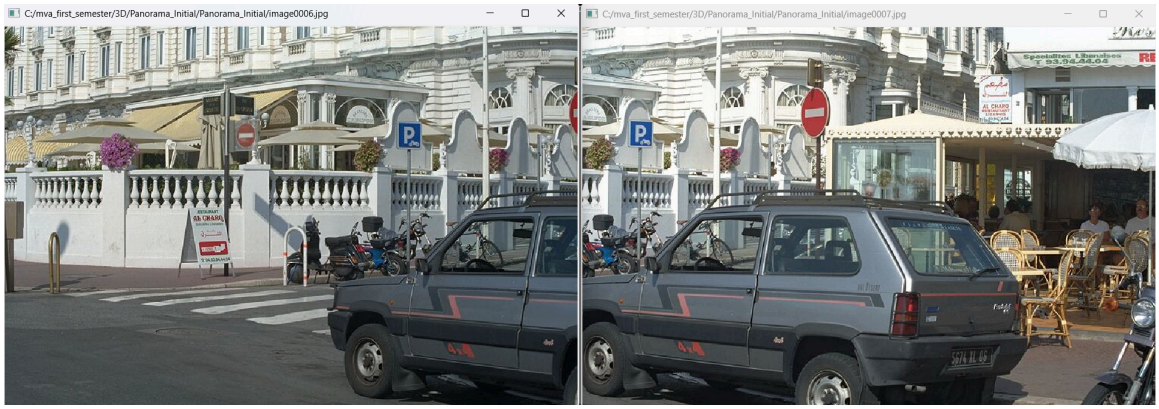

Figure 1: Two windows at the start of the application

2. The user clicks 4 or more corresponding points in both images. Each clicked point is visualized with a red circle. The more points selected, the better the panorama alignment will be.

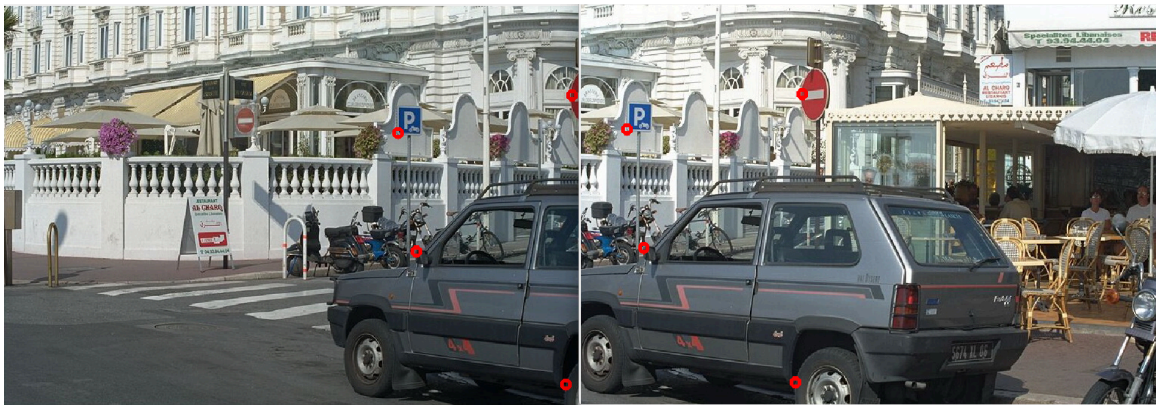

Figure 2: Corresponding point selection

3. By right-clicking the screen, point selection is finished. The program then checks the input:
   - If the two images don't have the same number of points, it prints the warning that the results may be inaccurate. Later, the points will be truncated to the minimum number of points in one image.
   - If one of the images has fewer than 4 points, it uses the identity (3x3) homography and prints a warning in the console.
   - Otherwise, it computes the homography from the correspondences.

4. The program generates and displays the panorama in a new window.

Figure 3: Constructed panorama

Other results

Figure 4 shows another selection of points. Compared to the first case, the points are less well distributed, so the resulting panorama in Figure 5 is of lower quality.

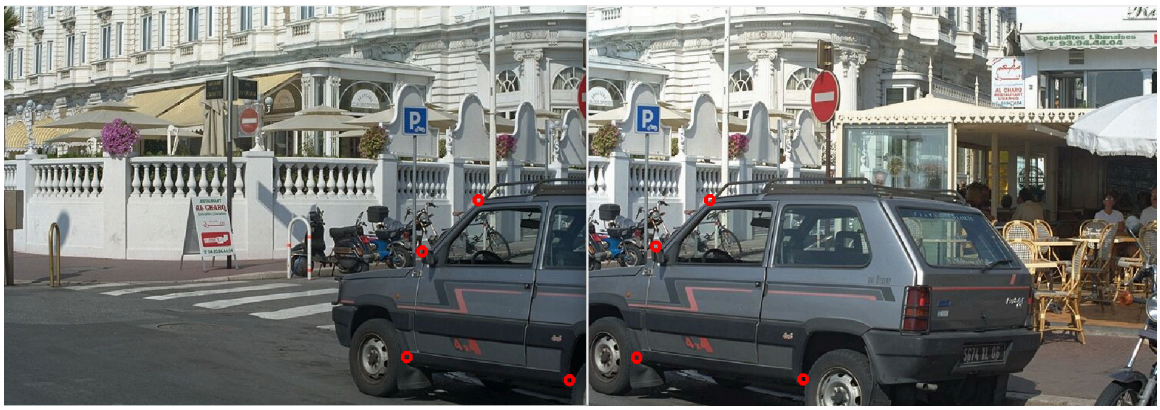

Figure 4: Poor point selection

Figure 5: Panorama constructed from points in Figure 4

Figure 6 shows the selection of 23 points. The panorama in Figure 7 shows the best geometric alignment because of the larger and better-distributed set of points.

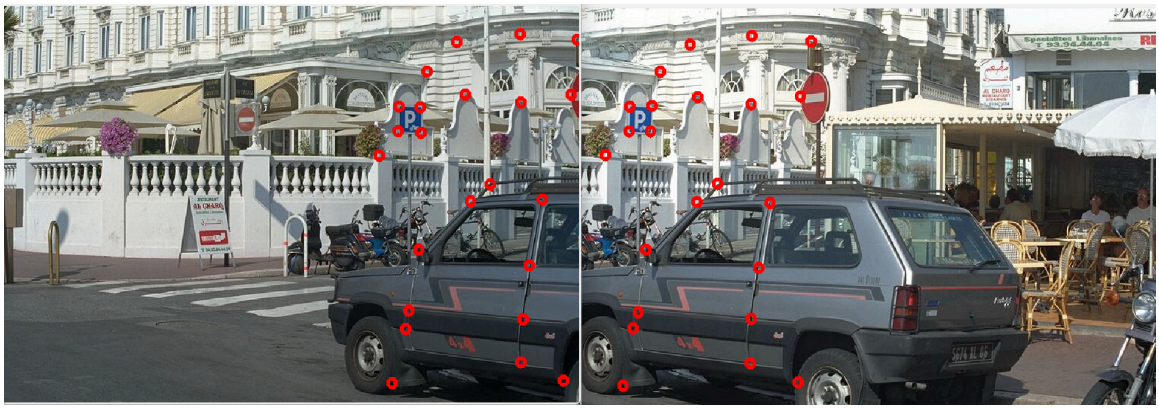

Figure 6: Greater point selection

Figure 7: Panorama constructed from points in Figure 6

Observed limitations

The panorama constructed with *image0006.jpg* and *image0007.jpg* has a visible color seam in blended regions. This happens because the two input images have different brightness/exposure.