GraphCut

Description

The goal of the project is to compute the disparity map associated to a pair of images using graph cuts. The energy is optimized with zero-mean normalized cross-correlation (ZNCC; data term) and a smoothness constraint (regularization term). The program first constructs a linear multi-label graph with disparities as labels, and then minimizes the energy using the max-flow algorithm. Finally, the program extracts the disparity map from the min-cut and displays it. If the setup supports OpenGL, it also shows a 3D mesh reconstructed from the final disparity map.

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.
For the 3D view at the end of the program, Imagine++ must be built with OpenGL support. If OpenGL is not available, the program simply skips the 3D view. All the previous steps won't be affected.

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 (defaults):
Run `.\build\Release\GCDisparity.exe` (Windows) or `./build/GCDisparity.exe` (Linux/Mac) command. The program loads the default images *face0.png* and *face1.png* and uses the default disparity range *dmin=10* and *dmax=55.*

Using the command line (custom params):
When using custom params, you should provide all four arguments in the command line. Therefore, run `.\build\Release\GCDisparity.exe <im1> <im2> <dmin> <dmax>` (Windows) or `./build/GCDisparity.exe <im1> <im2> <dmin> <dmax>` (Linux/Mac) command. Custom images should be placed in the same directory as the default images.

Using VS Code

Run the built target by pressing the Run button (bottom toolbar). The program will load the default images *face0.png* and *face1.png* and use the default disparity range *dmin=10* and *dmax=55.*
To use custom params, you should specify all four parameters in the `.vscode/launch.json` file under `"args"` field (example: `"args": ["im1.png","im2.png","10","55"]`). Custom images should be placed in the same directory as the default images.

Use case scenario

In this section, we show how the program works with example screenshots.
1. When the program is run, a window opens showing left and right images (Figure 1). In this example, the left image is shown on the right side of the window. The parameters (*disparity range, lambda, patch size, sigma, zoom*) are printed in the console.

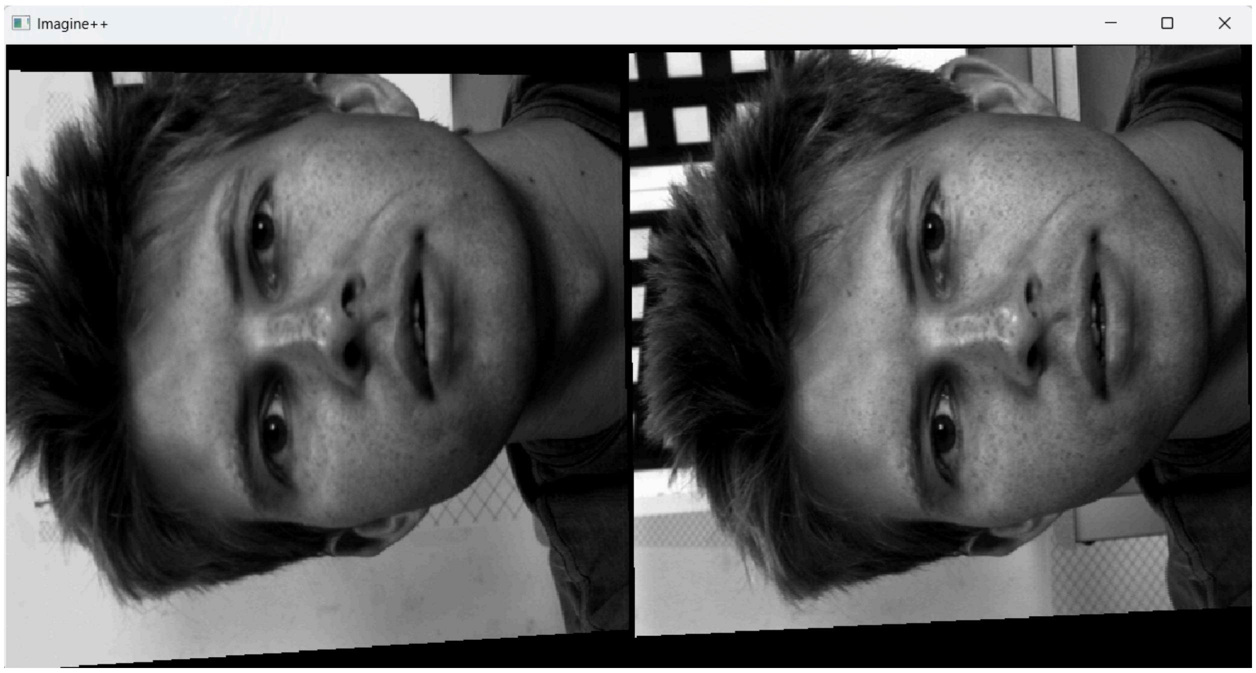

Figure 1: Window at the start of the application.

2. When the program starts (step 1), it immediately begins constructing the multi-label graph. Once the graph is built, the program computes the min-cut and extracts the disparity map from it. During computation, the percentage of progress is printed in the console. When the process is finished, the window automatically replaces the image on the left side of the window with the computed disparity map (Figure 2).

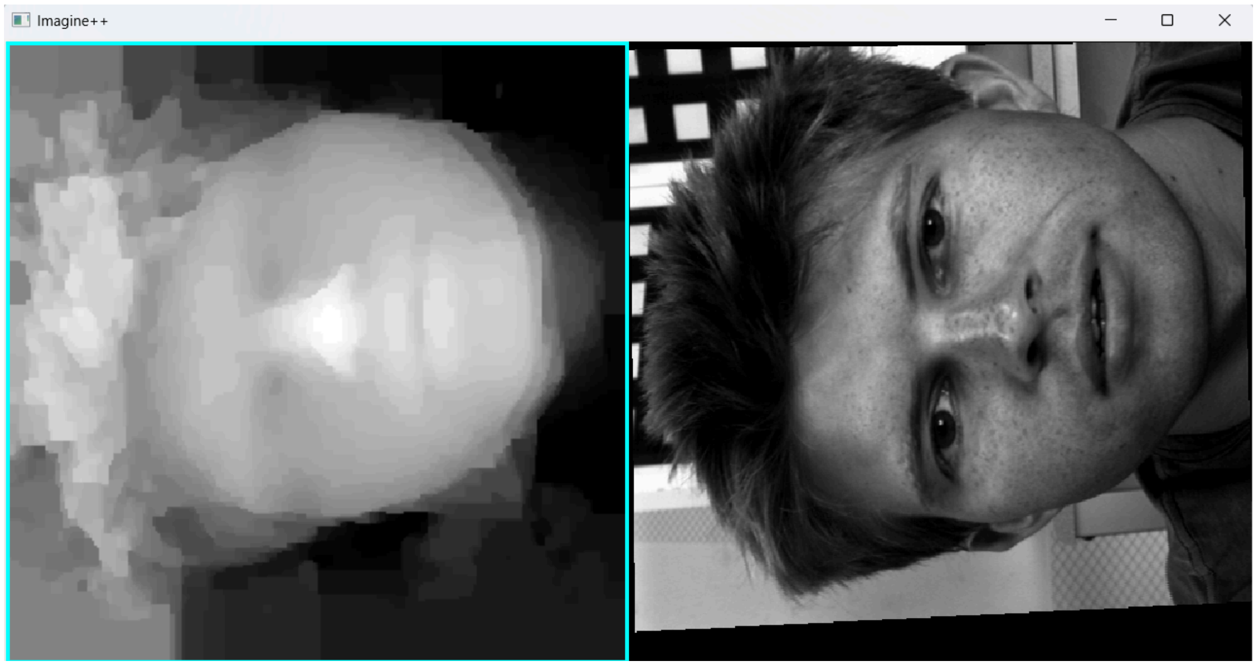

Figure 2: Disparity map (left) computed via graph-cut.

3. The user should now click on the window to smooth the disparity map using blur. The disparity map will automatically be redisplayed (Figure 3).

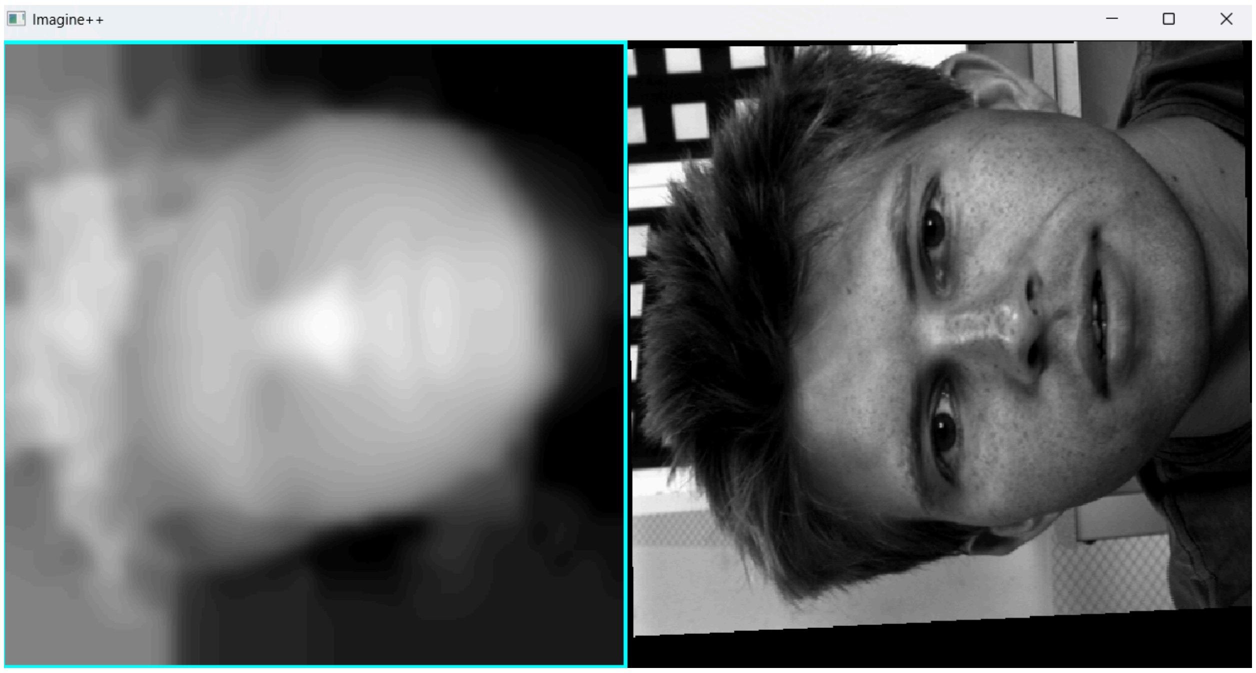

Figure 3: Smoothed disparity map after applying blur.

4. If Imagine++ was built with OpenGL, an additional window opens upon the user’s click and displays a 3D mesh reconstructed from the final disparity map (Figure 4).

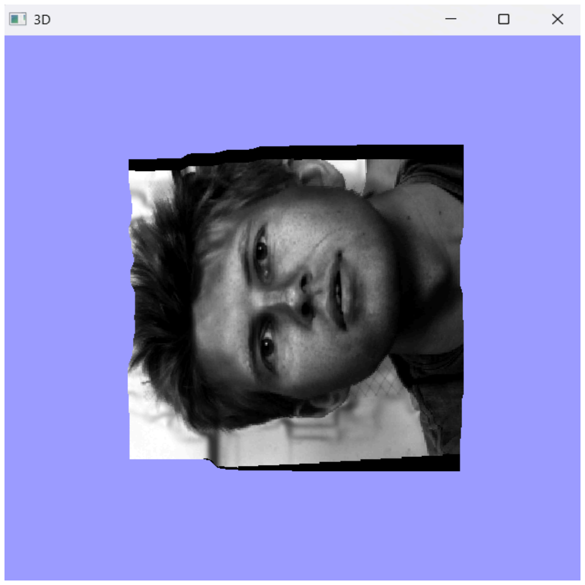

Figure 4: 3D window showing the mesh reconstruction from the final disparity map.

The user can toggle between textured and shaded views by clicking on the mesh and can rotate it using Shift and mouse dragging (Figure 5). The user can also zoom in and out. All of the interaction options and commands are printed in the console.

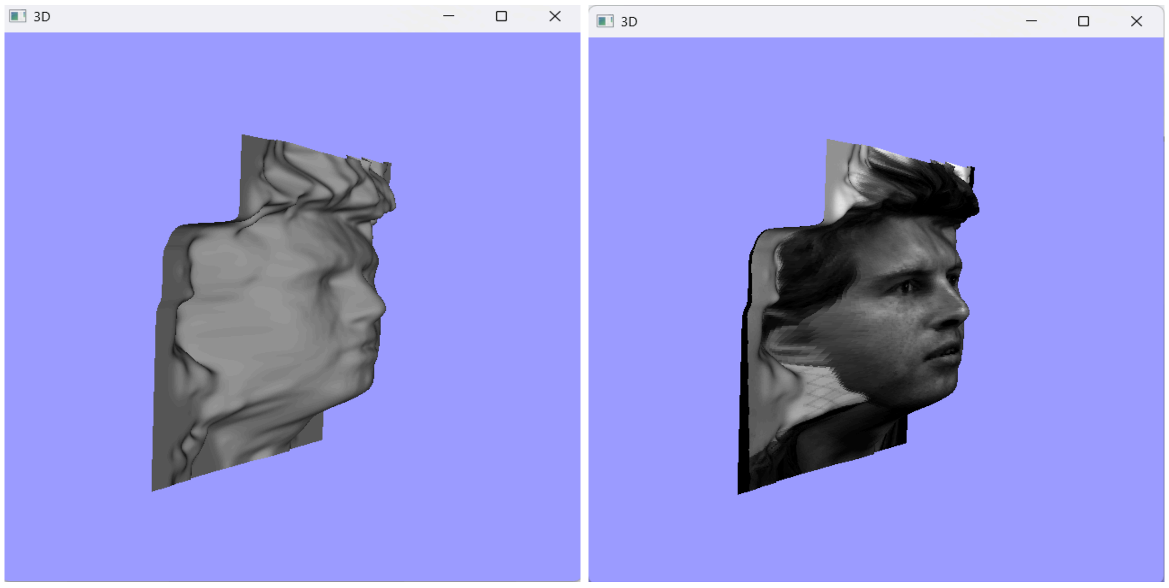

Figure 5: Shaded (left) and textured (right) rotated meshes.

The user can also toggle between solid, wire, and points mode using Shift+M buttons (Figure 6).

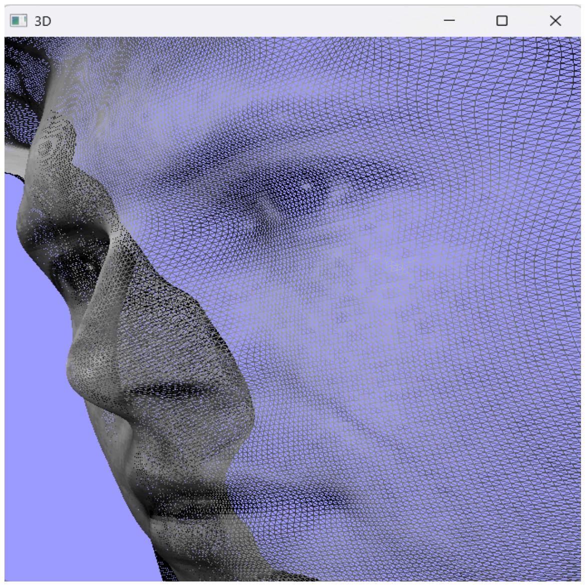

Figure 6: 3D mesh displayed in wire mode showing triangular meshes.

To finish the program and shut both the windows down, the user should click the right-click on mouse.
If OpenGL isn’t available, the program prints a message to the console and skips the 3D view. The first window stays open until the user decides to close it.