# Seeds

## Description

The goal of the project is to compute the disparity map associated to a pair of images using normalized cross-correlation (NCC) on local patches. The program first computes a dense disparity map, then finds high-confidence points (seeds) for which NCC ≥ 0.95, and lastly, the program propagates computed seed disparities to the 4-neighbors. Finally (and optionally, only if the setup supports OpenGL), the program shows a point cloud computed from the final disparity map. The output of the program is as follows:

- 0dense.png - dense disparity map for all points.
- 1seeds.png - disparity map restricted to seed pixels only.
- 2final.png - final disparity map after seed propagation.
- (Optional) 3D window - interactive window showing a point cloud reconstructed from the 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\Seeds.exe` (Windows) or `./build/Seeds.exe` (Linux/Mac) command. The program loads the default images *im1.jpg* and *im2.jpg* and uses the default disparity range *dmin=-30* and *dmax=-7.*

*Using the command line (custom params):*
When using custom params, you should provide all four arguments in the command line. Therefore, run `.\build\Release\Seeds.exe <im1> <im2> <dmin> <dmax>` (Windows) or `./build/Seeds.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 *im1.jpg* and *im2.jpg* and use the default disparity range *dmin=-30* and *dmax=-7.*
To use custom params, you should specify all four parameters in the `.vscode/launch.json` file under `"args"` field (example: `"args": ["im1.jpg","im2.jpg","-30","-7"]`). 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 multi-tab window opens (Figure 1). The window contains 5 tabs:
- *image 1:* left image.
- *image 2:* right image.
- *dense:* dense disparity map for all points.
- *seeds*: disparity map restricted to seed pixels only.
- *propagation*: final disparity map after seed propagation.

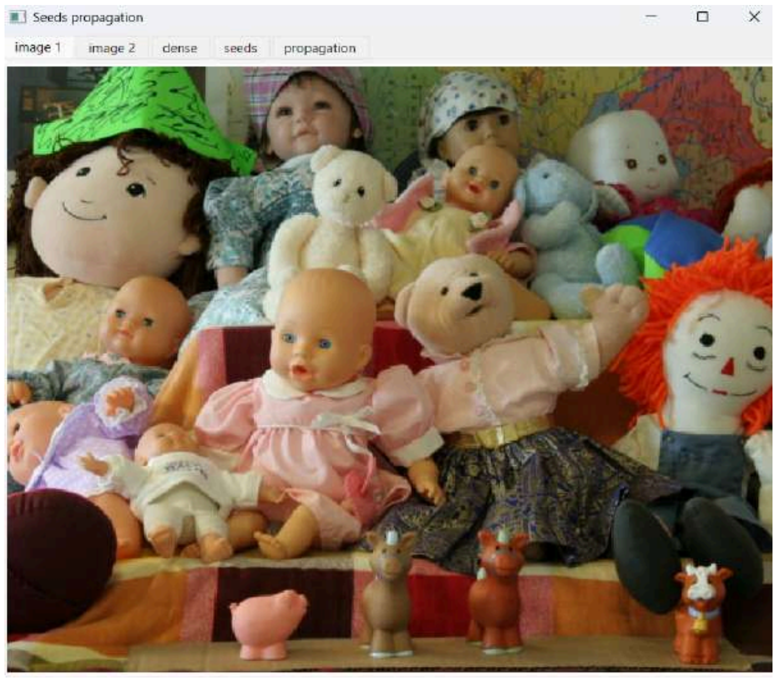

*Figure 1: Window at the start of the application.*

The window opens on the *image 1* tab by default (Figure 1). The user can click on the *image 2* tab to view the right image (Figure 2). The result tabs, i.e., *dense, seeds,* and *propagation* tabs*,* are all initially empty (Figure 3).

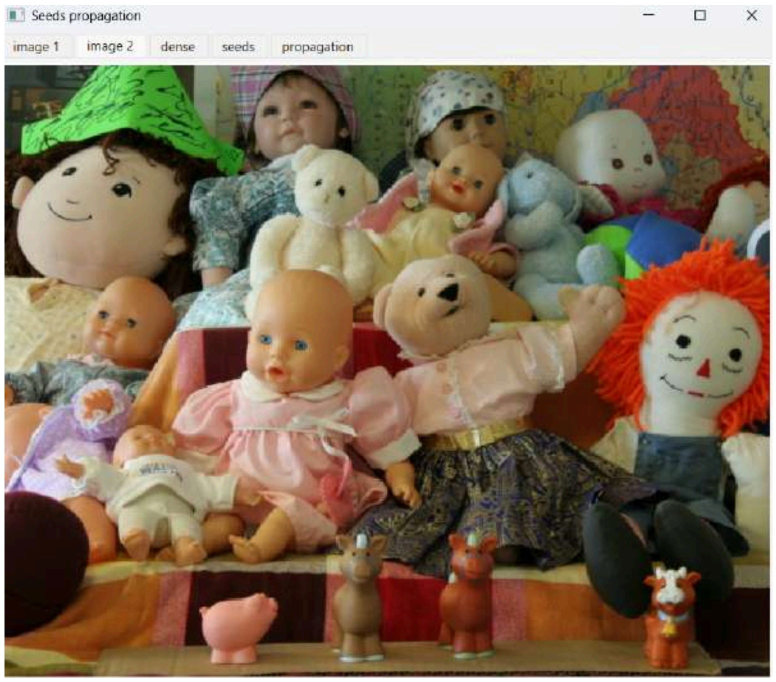

*Figure 2: Tab image 2 showing the right image.*

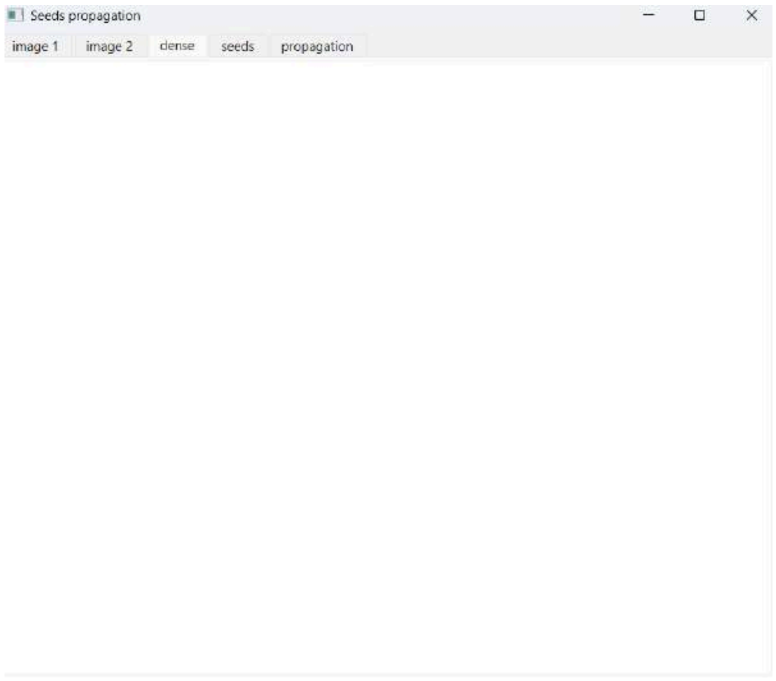

*Figure 3: Tab dense initially empty dense disparity map image.*

2. When the program starts (step 1), it immediately begins computing the dense disparity map between the left and right images by finding the disparity with the highest NCC score for each point. During the computation process, the percentage of progress is printed in the console. Once the process is done, the window automatically switches to the dense tab and displays the dense disparity map. (Figure 4).

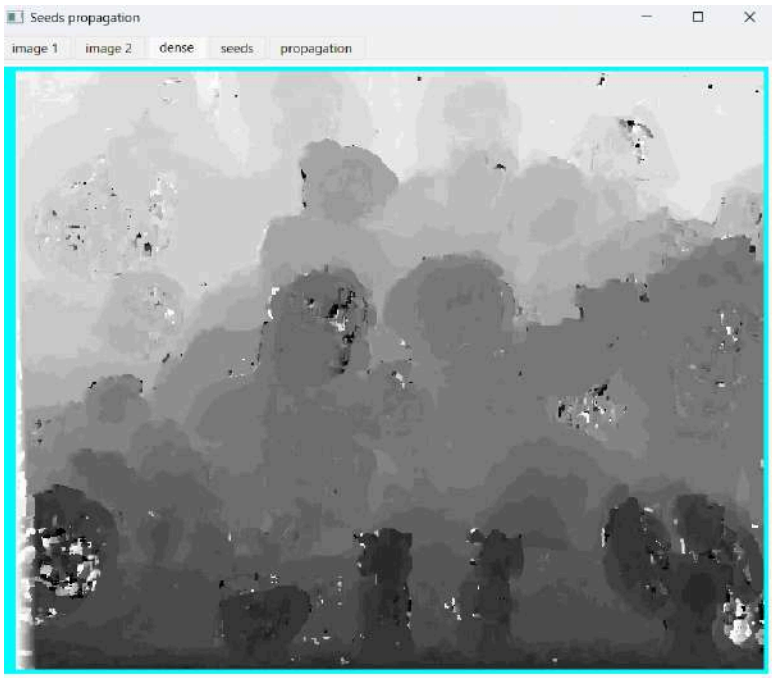

*Figure 4: Tab dense showing dense disparity map. Cyan regions mark pixels where the disparity is either outside the predefined range or has not been assigned.*

3. After the dense disparity map is computed, the program immediately starts to compute the seed disparities. The program keeps only the disparities where the NCC score is sufficiently high (≥ 0.95). The percentage of progress is once again printed in the console.
Once the process is done, the window automatically switches to the seeds tab and displays the seed disparity map (Figure 5).

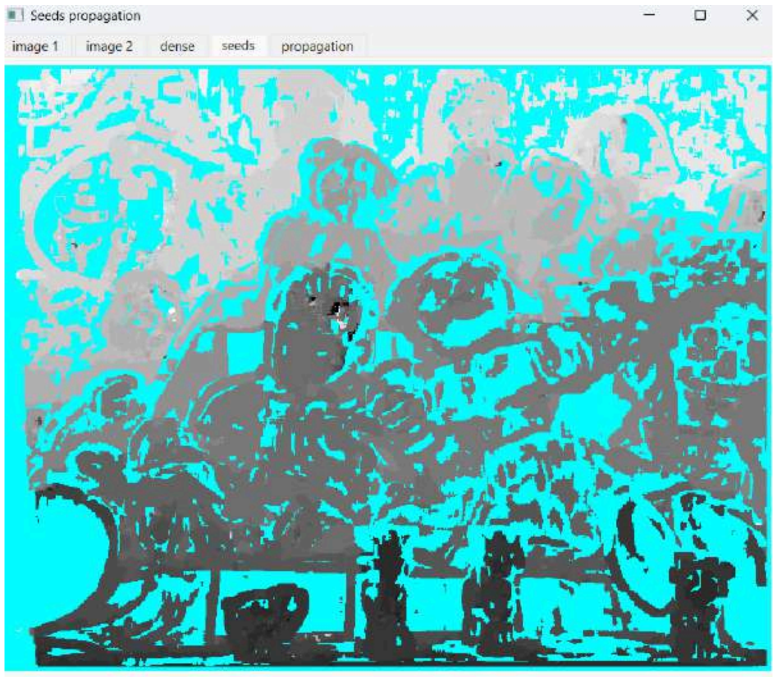

*Figure 5: Tab seeds showing the seed disparity map. There are a lot more cyan regions since they now also include the pixels where the NCC score is not sufficiently high, hence, their disparity is not kept.*

4. Once the previous processes are finished, the program finally starts the propagation step during which it expands the seed disparities computed in the previous step to four neighboring pixels. For every neighbor pixel Q of seed P that has no valid disparity, the disparity of Q is set by the highest NCC score among $d_p - 1$, $d_p$, and $d_p + 1$, where $d_p$ is the disparity at P.
Once the propagation is done, the window automatically switches to the propagation tab and displays the final disparity map (Figure 6).

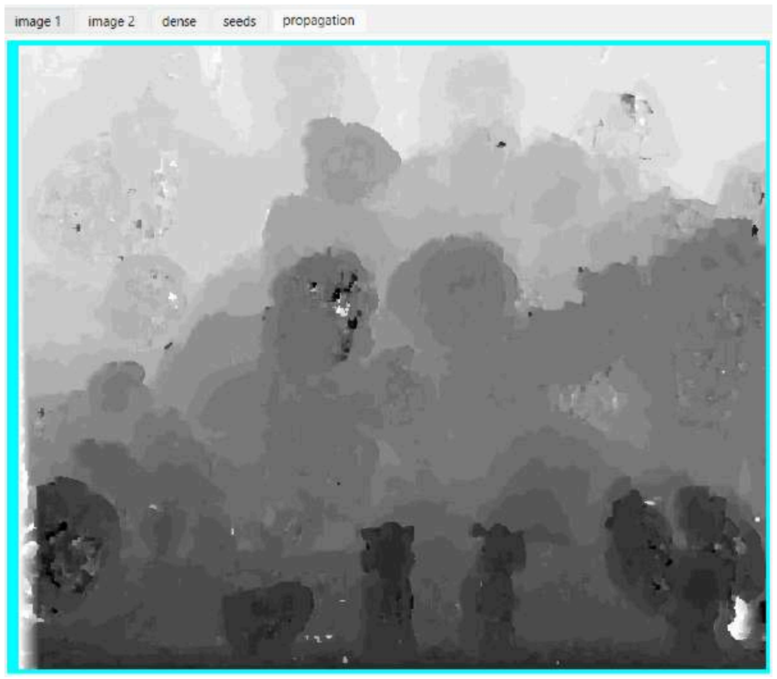

*Figure 6: Tab propagation showing the final (propagated) disparity map. Cyan regions mark pixels where the disparity is either outside the predefined range or has not been assigned.*

5. If Imagine++ was built with OpenGL, an additional window opens and displays a point cloud reconstructed from the final disparity map (Figure 7). To convert from 2D to 3D space, the program uses intrinsic parameters given by the Middlebury website and computes the depth of each pixel as a scaled inverse of its disparity value.
The user can interact with the window by holding the Shift button and dragging the mouse to rotate the point cloud in 3D space (Figure 8). The user can also zoom in and out.
To finish the program and shut both the windows down, the user should click anywhere in the 3D window without holding the Shift button.

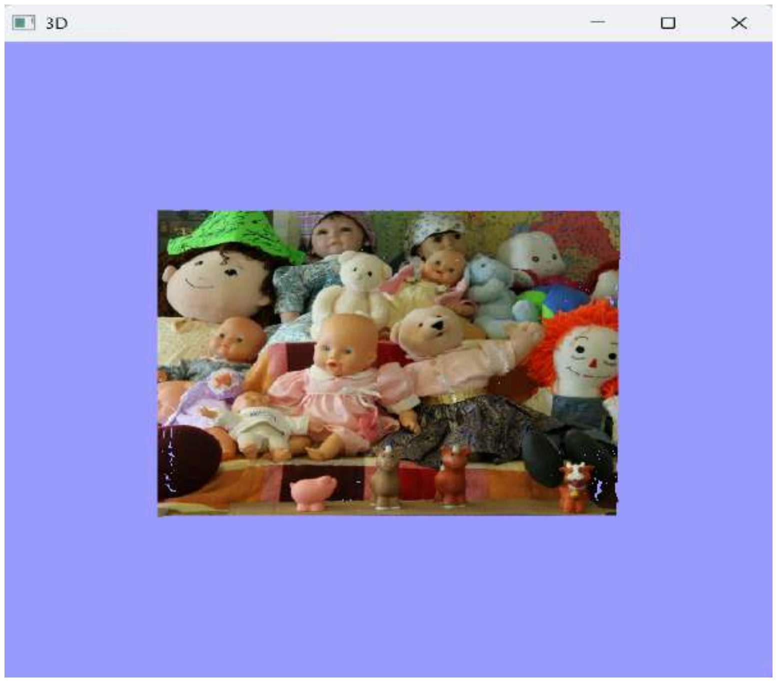

*Figure 7: 3D window showing the point cloud reconstruction from the final disparity map.*

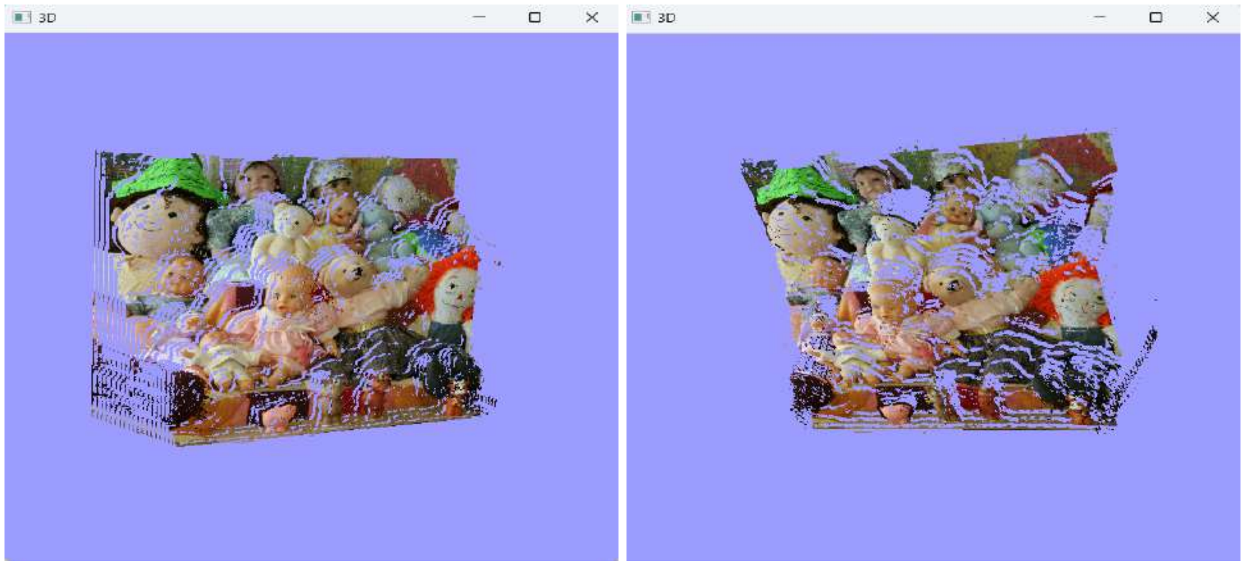

*Figure 8: Two views showing the same scene rotated from different angles. Each point corresponds to a pixel in the original image positioned in 3D according to its computed disparity.*

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.