Stretching and Shearing cell registration

Program to map before and after stretch/shear images of cell colonies via point cloud registration mapping algorithms.

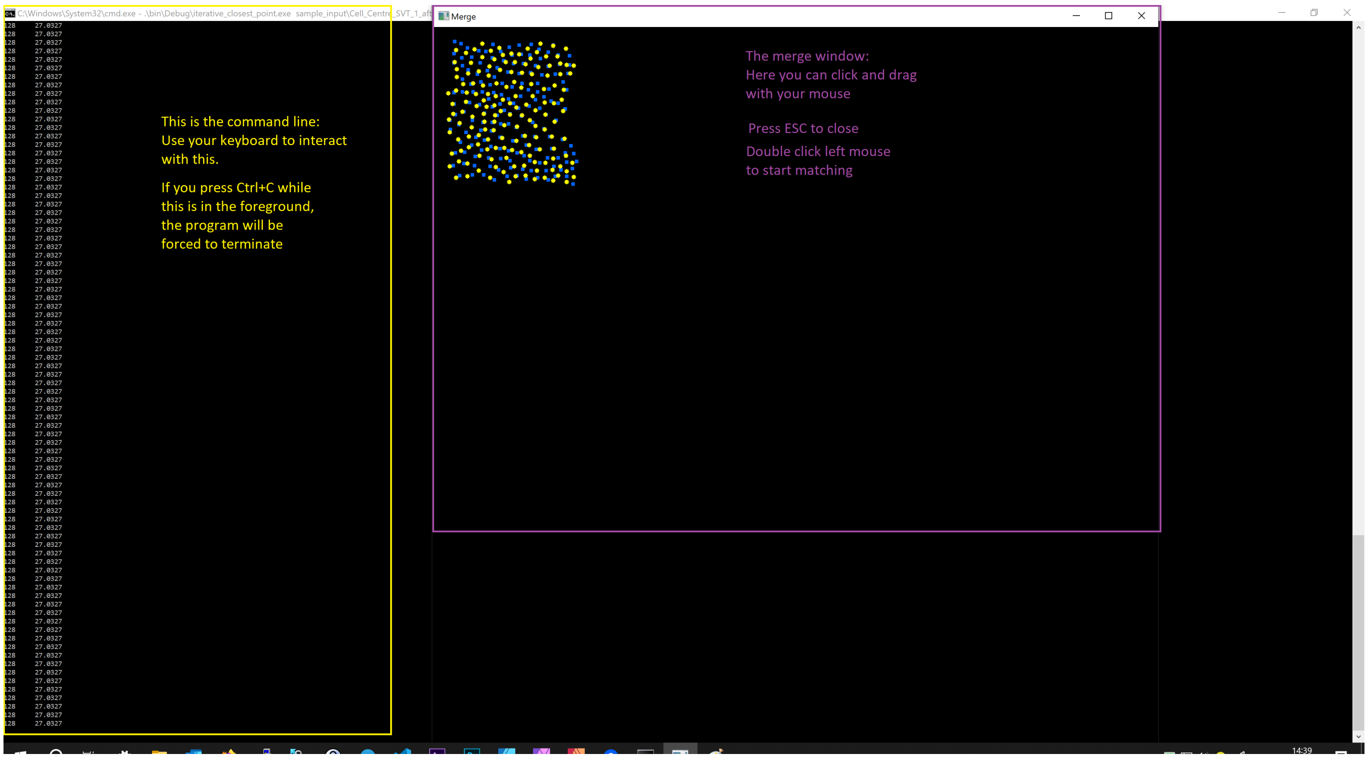
Usage

The compiled code for this project is usually provided as `.\iterative_closest_point.exe` in a folder with certain libraries (i.e. files ending in `.dll`). If the full project is provided, the executable will be in a folder under `./bin`. If you are not changing the code yourself, use the code in `./bin/Release/` under windows. If so, the program should be called as `./bin/Release/iterative_closes_point.exe` instead of `./iterative_closest_point.exe` as written in the following.

As you start the program with a command line invocation like:

```
.\iterative_closest_point.exe <input_file_before> <input_file_after> <output_directory>
```

a window will open.



This is the command line:
Use your keyboard to interact with this.

If you press Ctrl+C while this is in the foreground, the program will be forced to terminate

The merge window:
Here you can click and drag with your mouse

Press ESC to close

Double click left mouse to start matching

Controls

In this window, by clicking the left mouse button and dragging the mouse, you can shift the after distribution spatially. By right-clicking and dragging, you can change the scaling. This is kinda fiddly, I would not rely on it.

Instead, by pressing the keyboard key `m`, you can get into a cell-matching window. There you can click on a cell in the before and then in the after group to tell the Program to match the two during its optimization attempts. Click with the left mouse button on a highlighted (red) cell once to mark the before cell, then click a second time on a highglighted (red) after cell to match them. Afterwards, the matching window will close and you will see the two corresponding cells being put on top of each other and displayed in a different color from the rest.

In the main window, by double clicking the left mouse button, you will trigger the automatic optimization of the colony matching. This may run for a while depending on the number of cells in your sets. However, you should be able to see progress or at least slight changes in the plotted after-distribution

After the fitting is done or generally if you want to close the program, press the `ESC` on the keyboard while in the merge window or press `Ctrl+C` when in the command line window to force a program shutdown. If you do the latter, while the fit is still running, the output files may be empty or not fully there. So please consider when it is appropriate to force a shutdown of the program while it is running. You can also close the cell window. Sometimes it does not immediately recognize the attempt to close the window due to computations being run. Then please try and close the window again.

Results will be written to `<output_directory>_mapping.txt` (a list of before and after indices) and `<output_directory>_all.txt` a file with the before indices, the before x and y position and the after index with its x and y position.

To run, the program will try to load a configuration file (see section `Configuration` for options.) from the path `./conf/config.txt` by default, but any path to the configuration file can be provided as option `-c <config_file_path>`.

In this configuration file, minimum and maximum values for the attempted interval of stretch and shear in x and y direction respectively are provided. If you only want either stretch or shear to be optimized, choose minimum and maximum values very close to 1.0 for the other paramaters, i.e. `stretch_x_min 0.999` and `stretch_x_max 1.0001` if you do not want the stretch in x direction to be optimized.

Examples

If you want to run a match of the files `input1.txt` and `input2.txt` using the configuration at `config.txt` and write the results to `output/`, then the correct command would be:

```
.\iterative_closest_point.exe input1.txt input2.txt output/ -c config.txt
```

or without a custom configuration:

```
.\iterative_closest_point.exe input1.txt input2.txt output/
```

Options are generally written in a style like `-o <value>` or `--long_option <value>`, where `o` and `long_option` are the names of the option to be set, either in short form (i.e. only one character) or in long form (i.e. usually a full word but no whitespaces allowed within that word). To get a list of all available options, use the option `-h`, i.e.

```
.\iterative_closest_point.exe -h
```

To get the version of the program that you are using, use `--version`, i.e.

```
.\iterative_closest_point.exe --version
```

Support: Errors and Bugs

If there is an issue while using the library, do not hesitate to contact me at `kevin.hoellring@fau.de`. To be able to help you, it would be great, if you could provide the following things:

- input before and after files

- your config file or the information that you did not use a config file
- the version of the program you are using, obtainable by invoking `.\iterative_closest_point.exe --version`. The output will provide you with all of the necessary information.
- A description of the issue you are facing, e.g. "The program crashes when I double click", "The program does not seem to be able to load the input files", "The program crashes randomly with no error output", "The output files are empty or not produced", etc. so that I have an idea what I am looking for 😃

Quirks

- Currently, the _all.txt mapping output has questionable positions for some of the cells. Do not rely on that data, instead use the indices to look up the position in your input files.
- If no configuration is provided, certain default settings are applied (no scaling, no minimum scale, no maximum scale)
- I added shearing to the code. If shear or stretching should be disabled, the upper and lower boundary values should be set close to 1.0 so that there is almost no range for a spread in those values.

Dependencies

This code depends on

- git for version control and submodules (needs to be installed)
- CMake for building (needs to be installed)
- OpenCV (version 4.8.0)
- PCL (version PCL-1.13.1)
- Eigen library (version 3.4.0 for PCL if building it yourself)
- cxxopts (for command line argument parsing, provided as a submodule)

For building the program it uses CMake, which needs to be installed. To build, a rather new compiler is required. On Windows, this makes Visual Studio (2022 or later) necessary, on Linux, clang or the gnu c++ compiler should be fine.

The library cxxopts is included as a submodule to this repository. To download the necessary files, use `git submodule init` and then `git submodule update`. If this doesn't work, check the current git submodule documentation online for how to download submodules.

Installing these tools proved to be more complex than I had expected. On Windows, I recommend downloading vcpkg, then installing with .\vcpkg.exe install eigen pcl qt5 opencv. Afterwards, it needs to be integrated using .\vcpkg.exe integrate install. Then the libraries should be available to cmake.

Directories

Name	Purpose
`bin`	Output of the compiled program. Not in version control
`build`	Build directory if present. Not included in version control
`conf`	Default configuration file

Name	Purpose
`doc`	Documentation of the code, mostly images
`include`	headers
`sample_input`	Input files to test the point cloud matching algorithm
`src`	source code for program

Configuration

The program expects a configuration file at the path provided by the option `-c` or alternatively `--config_file` (default: `./conf/Config.ini`). The configuration file should be a text-format file consisting of multiple rows, with each row containing a key (i.e. a non-whitespace separated string) then a whitespace and then the value associated with that key again with no whitespace.

The following key and value pairs are supported:

Key	Value	Purpose
Stretch settings:		
`stretch_x_min`	floating point number (e.g. 0.8)	Minimum scale/stretch in x direction to be considered
`stretch_x_max`	floating point number (e.g. 1.2)	Maximxum scale/stretch in x direction to be considered
`stretch_y_min`	floating point number (e.g. 0.8)	Minimum scale/stretch in y direction to be considered
`stretch_y_max`	floating point number (e.g. 1.2)	Maximxum scale/stretch in y direction to be considered
Shear settings:		
`shear_x_min`	floating point number (e.g. -0.1)	Minimum shear in x direction to be considered (negative for opposite shear direction)
`shear_x_max`	floating point number (e.g. 0.1)	Maximxum shear in x direction to be considered
`shear_y_min`	floating point number (e.g. -0.1)	Minimum shear in y direction to be considered (negative for opposite shear direction)
`shear_y_max`	floating point number (e.g. 0.1)	Maximxum shear in y direction to be considered
Load time rescale:		
`loadtime_scaling`	floating point number (e.g. 1.0)	Scale factor with which input coordinates are multiplied at time of loading

Roadmap

A few aspects of this tool are still being worked on. Here is a list of things having been done

DONE

- Choose two cells in before and after to set as corresponding to perform shearing and stretching relative to
- Fixed the cost function not actually depending on the transformation. It will now converge and give actual correspondences
- Fixed issues with security programs detecting configuration as virus
- Fixed the weird evolutional optimization very much diverging from the correct parameters over time

BUGS

- Issues with the _all.txt output positions.

PLANNED

- Come up with a better control concept

MAYBE

- Draw before and after correspondence with different colors?
- Draw lines between correspondences
- Add indices to plot in UI in a debug mode

Below this is just some internal stuff, no need to go through it 😃

Integrate with your tools

- [ ] Set up project integrations

Collaborate with your team

- [ ] Invite team members and collaborators
- [ ] Create a new merge request
- [ ] Automatically close issues from merge requests
- [ ] Enable merge request approvals
- [ ] Set auto-merge

Test and Deploy

Use the built-in continuous integration in GitLab.

- [ ] Get started with GitLab CI/CD
- [ ] Analyze your code for known vulnerabilities with Static Application Security Testing(SAST)
- [ ] Deploy to Kubernetes, Amazon EC2, or Amazon ECS using Auto Deploy
- [ ] Use pull-based deployments for improved Kubernetes management
- [ ] Set up protected environments

Editing this README

When you're ready to make this README your own, just edit this file and use the handy template below (or feel free to structure it however you want - this is just a starting point!). Thank you to makeareadme.com for this template.

Suggestions for a good README

Every project is different, so consider which of these sections apply to yours. The sections used in the template are suggestions for most open source projects. Also keep in mind that while a README can be too long and detailed, too long is better than too short. If you think your README is too long, consider utilizing another form of documentation rather than cutting out information.

Name

Choose a self-explaining name for your project.

Description

Let people know what your project can do specifically. Provide context and add a link to any reference visitors might be unfamiliar with. A list of Features or a Background subsection can also be added here. If there are alternatives to your project, this is a good place to list differentiating factors.

Badges

On some READMEs, you may see small images that convey metadata, such as whether or not all the tests are passing for the project. You can use Shields to add some to your README. Many services also have instructions for adding a badge.

Visuals

Depending on what you are making, it can be a good idea to include screenshots or even a video (you'll frequently see GIFs rather than actual videos). Tools like ttygif can help, but check out Asciinema for a more sophisticated method.

Installation

Within a particular ecosystem, there may be a common way of installing things, such as using Yarn, NuGet, or Homebrew. However, consider the possibility that whoever is reading your README is a novice and would like more guidance. Listing specific steps helps remove ambiguity and gets people to using your project as quickly as possible. If it only runs in a specific context like a particular programming language version or operating system or has dependencies that have to be installed manually, also add a Requirements subsection.

Usage

Use examples liberally, and show the expected output if you can. It's helpful to have inline the smallest example of usage that you can demonstrate, while providing links to more sophisticated examples if they are too long to reasonably include in the README.

Support

Tell people where they can go to for help. It can be any combination of an issue tracker, a chat room, an email address, etc.

Contributing

State if you are open to contributions and what your requirements are for accepting them.

For people who want to make changes to your project, it's helpful to have some documentation on how to get started. Perhaps there is a script that they should run or some environment variables that they need to set. Make these steps explicit. These instructions could also be useful to your future self.

You can also document commands to lint the code or run tests. These steps help to ensure high code quality and reduce the likelihood that the changes inadvertently break something. Having instructions for running tests is especially helpful if it requires external setup, such as starting a Selenium server for testing in a browser.

Authors and acknowledgment

Show your appreciation to those who have contributed to the project.

License

For open source projects, say how it is licensed.

Project status

If you have run out of energy or time for your project, put a note at the top of the README saying that development has slowed down or stopped completely. Someone may choose to fork your project or volunteer to step in as a maintainer or owner, allowing your project to keep going. You can also make an explicit request for maintainers.