CS205 C/CPP Final Project

Contributor:

- Wenjin Li 12012514
- Fanbo Zhou 12012519
- Guoyi Dai 12011211

Overall introduction:

Our code realize the `Matrix` class using `template` definition. So we can change the type we passing to the `class` and construct the needed type of matrix. We can using the `Matrix<T>` to realize the function of the numeric types.

The storage of our data of the matrix using a pointer to point to the start address of the data and we provide the get and set function for the user to get the needed data in the `(row,clo)`. Hence, we add some functions and exceptions for users to test the data validation.

Constructor:

We provide 5 constructors for our users to use.

```
Matrix() : data(nullptr), size(0), m_col(0), m_row(0) {}

Matrix(size_t col, size_t row);

Matrix(const cv::Mat &mat); // 构造函数，将cv::Mat转换为Matrix

Matrix(const Matrix<T> &a); // copy constructor

Matrix(Matrix<T> &&a) noexcept; // move constructor
```

The first constructor is the default constructor returns an empty `Matrix` with no data and rows.

The second constructor is the most used constructor which needed two parameters: cols and rows.

The third constructor realized the function of transformation from `opencv` matrix to our `Matrix`, it receives a `cv::Mat` as a parameter and returns our defined `Matrix`.

The forth and fifth constructor is the copy and move constructor, which is very important in `assignment statements` which is largely used in `operator override`.

Basic arithmetic

The basic arithmetic is realized by the

Eigenvalues and Eigenvectors

We set a new class named `SMatrix`, which represents a square matrix that only the square matrix supports the operations, so we will throw an exception which represent the matrix is not a square matrix.

The realization of this part is mainly depend on the `unique_ptr` and the overriding of `operator[]` which enables us to use the `SMatrix` the same as a `2-d array`, for example, we can use `SMatrix[1] [1]` to get the data at the point (1,1). So we can get the needed data in a much easier way. Also, the `unique_ptr` enables us to use it in the smarter way that it will automatically release the memory pointed by it when we finish the use of it.

The calculation in this part follows the principle of linear algebra. We just do what we have done in our freshman year. We first augment it to a `augmented matrix` and then we do `Gram-Schimidt orthogonalization` to it to get the left part of the matrix to an `eye matrix` so we get the `eigenvalues and eigenvectors` in the right part.

Traces, Inverse and Determinant

They are used in the above part, so it maybe much easy for us to get them down. We also takes what we learned in linear algebra and written them down on the paper, the rest work is translate the process into the `cpp` language that can do the job for computers.

Reshape

When storing the data of matrix, we use a pointer to allocate a connected space to store all the data so that the reshape is easy for us. We first do a check on whether the reshape size is equal to the previous one, If unmatched, then throw a exception. Else, we change the col and row definition.

Convolution

As for convolution, we first extend the matrix the matrix size by kernel size. Then use the up-left center of kernel as the convolution kernel to do the calculation.

Transformation with opencv