Поиск правильного положения оси и перерасчет проекций

Никитина Полина Владимировна
МФТИ ФПМИ Б05-924
Октябрь 2022

Содержание

Теоретическая часть

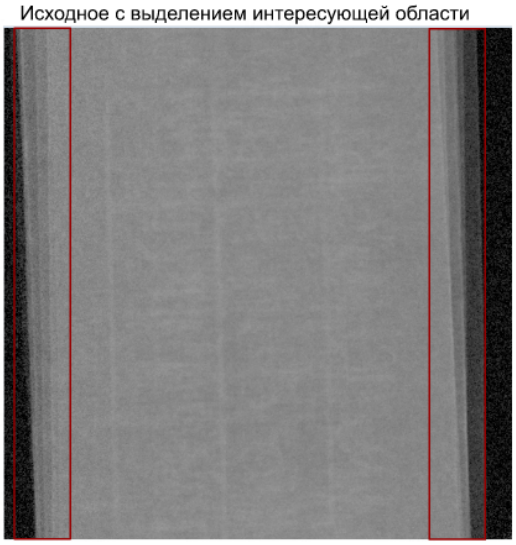
Введение.
При создании рентгеновских снимков объектов объект помещается на некоторую ось, после чего выполняется серия рентгентовских снимков при повороте объекта.
Таким образом при симметрии изображений сделанных при повороте объекта на 180° контуры объектов должны совпасть, однако на практике это не так, из-за смещения положения оси.
Таким образом, если найти угол смещения изображений друг относительно друга, то можно найти и угол смещения оси и выполнить корректировку изображений.

Постановка задачи.
По двум изображениям необходимо находить угол смещения оси вращения, а также корректировать изображения на найденный угол, если это требуется.

Описание алгоритма.
Для поиска угла смещения будем использовать гистограммы по направляениям изображений.
Для нахождения гистограммы по направлениям необходимо вычислить градиент изображения, он указывает направление изменения интенсивности изображения.
Для нахождения градиента изображения воспользуемся оператором Собеля (cv::Sobel) - это дискретный дифференциальный оператор, вычисляющий приближение градиента интенсивности изображения.
Оператор Собеля основан на свертке изображения в вертикальном и горизонтальном направлении.

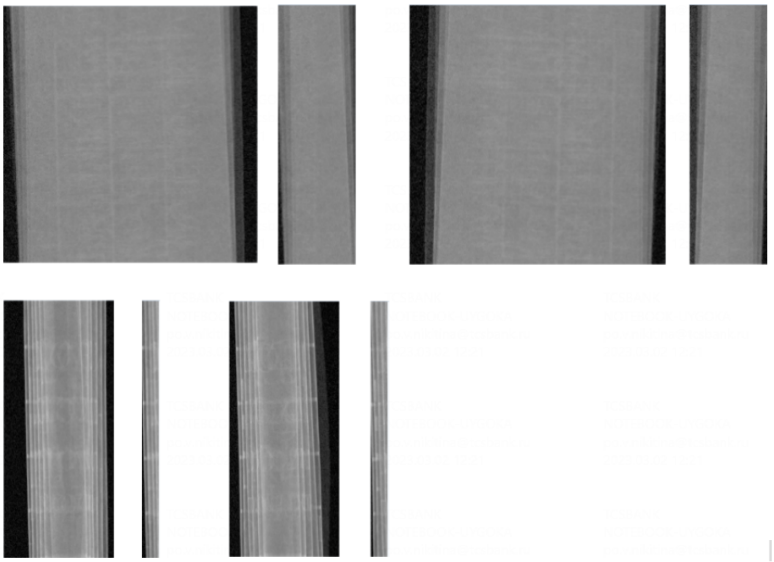
После вычисления градиента по осям x и y перейдем к углам с помощью следующей формулы: atan2(gr_x/gr_y)). Причем при посторении гистограммы хотим учитывать только углы, которые указывают на существенное изменение интенсивности изображения.
Не будем учитывать в гистограмме углы, для которы $\sqrt{x^2 + y^2} < threshold$, $threshold$ - некоторое пороговое значение. За $threshold$ выберем $\sqrt{x^2 + y^2}$, которое отделяет некоторую часть значений. В общем случае это 20%, но этот параметр можно изменять.

Если изучить выборку, на которой данный алгоритм будет использоваться, то можно заметить, что важно не все изображение, а лишь его часть.



В общем случае выделяем 30% изображения, но этот параметр можно изменять.
Размер рамки рассчитываю по формуле $bound = width * 0.3 / 4$. Тогда $bound*2$ - размер интервала, который задает ширину рамки в каждой половине изображения.

Для выделения важной части изображения алгоритм рассчитывает сумму $\sqrt{x^2 + y^2}$ в рамке размера $bound*2$ для всего изображения. В итоге получает два индекса, которые указывают на центр рамки в каждой половине изображения.

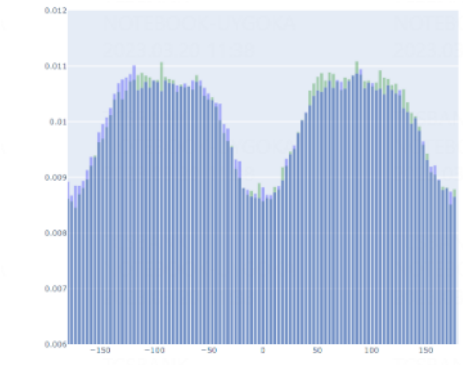


После всех преобразований алгоритм строит гистограмму по направлениям или гистограмму направленных градиентов.
Гистограмма направленных градиентов основана на подсчете количества направлений

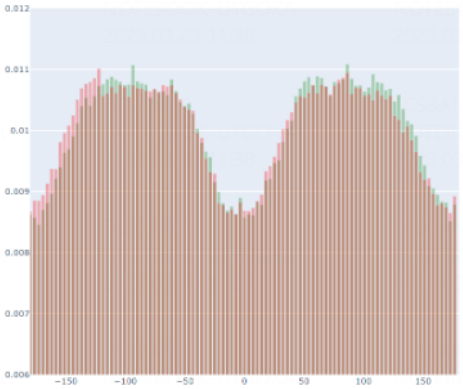
градиента в локальных областях изображения.
Заметим, что при поиске правильного положения оси нам необходимо найти угол, при повороте на который контуры изображений совпадут.
Как изменяется гистограмма по направлениям при повороте изображения на заданный угол:

То есть при повороте изображения гистограмма смещается на тот же угол.
Таким образом, задачу можно решать с помощью смещения гистограммы по направлениям до тех пор пока не будет достигнута минимальная разница между гистограммами.
Разницу между гистограммами будем оценивать с помощью квадратичной ошибки.

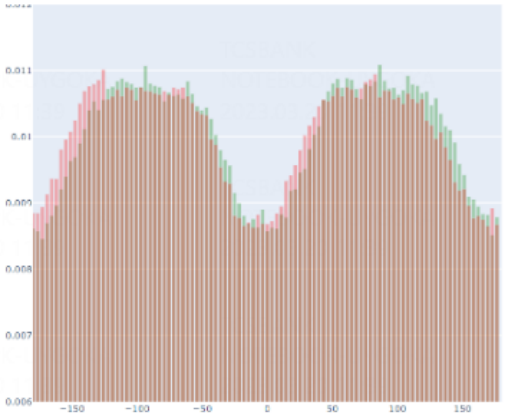
Пример построения и применения гистограмм по направлению:



Наложение двух изображений, одно из которых было отзеркалено
diff=5.324356350769988e-07



angle = 3.6
diff = 3.4594038437595716e-07



angle=7.2
diff=4.5846420533784714e-07

Пользовательская документация

Библиотека и установка.

Ссылка на библиотеку: https://github.com/apollinaria-sleep/Histograms.git

Чтобы использовать библиотеку необходимо:

1. Скачать себе репозиторий с кодом

```
git clone https://github.com/apollinaria-sleep/Histograms
```

2. Собрать библиотеку:

```
mkdir build
cd build
cmake ./path // здесь необходимо указать путь к директории
make
```

Описание работы библиотеки.

Основным классом в библиотеке является класс Histogram, который позволяет находить угол смещения одного изображения относительно другого с различной точностью, а также корректировать изобажения.

1. Инициализация
   В конструктор необходимо передать пути к изображениям. Изображения должны иметь одинаковую ширину и высоту.
2. Изменение и просмотр гиперпараметров
   В алгоритме есть два гиперпараметра: *threshold*, *bound*.
   - чтобы посмотреть *threshold* необходимо вызвать *CheckPartThreshold*()
   - чтобы изменить *threshold* необходимо вызвать *SetPartThreshold*(*new_threshold*)
   Аналогично для Bound.
3. *FindAngle*() - возвращает угол с точностью до 0.5 смещения оси, который был найден алгоритмом.
4. *CorrectImages*(*first_name, second_name, angle*) - корректирует изображения и сохраняет их по заданному пути. Аргумент *angle* опционален, если его не указывать, то функция скорректирует изображение на угол найденный алгоритмом.
5. *VisualizeHistograms*(*name*) - создает текстовый файл с заданным именем, в который сохраняет гистограмму, которую можно визуализировать с помощью ноутбука Отрисовка гистограммы.ipynb

Тестирование.

Тесты покрывают все описанные выше функции библиотеки, кроме пунктов 6 и 7.

Чтобы запустить тесты из директории *build* запустите следующую команду: ./histogram_tests

Техническая документация

Класс Histogram

1.1 Введение

Класс Histograms позволяет находить угол отклонения оси по двум изображениям и корректировать его.

2.1 Class List

Здесь перечислены классы, структуры, объединения и интерфейсы с описанием класса:

3 Class Documentation

3.1 Image Class Reference

Класс Image реализует обработку изображения типа tif.

#include <histograms.h>

Public Member Functions

- Image(const std::string name)
- void Flip()
- void CorrectImage()
- void WriteImage(char* name, float angle = 0)

Public Attributes

- size_t height
- size_t width
- float** mag = nullptr
- float** angles = nullptr

3.1.1 Detailed Description

Класс Image реализует обработку изображения типа tif.

Каждый объект класса Image хранит в себе следующую информацию:

- height - size_t высота изображения после преобразований внутри Image
- width - size_t ширина изображения после преобразований внутри Image

- mag - float** указатель на массив размера (height, width) с sqrt(grad_x^2 + grad_y^2)
- angles - float** указатель на массив с углами atan2(grad_x, grad_y)

3.2 Histograms Class Reference

Класс Histograms основной класс, позволяющий находить угол отклонения оси по двум изображениям и корректировать его.
#include <histograms.h>

Public Member Functions

- Histograms(const std::string image_name_0, const std::string image_name_1)
- void SetPartThreshold(float part_threshold)
- void SetPartBound(float part_bound)
- float CheckPartThreshold()
- float CheckPartBound()
- float FindAngle()
- void CorrectImages(char* first_name, char* second_name, float angle=-365)
- void VisualizeHistograms(char* name)

3.2.1 Detailed Description

Класс Histograms основной класс, позволяющий находить угол отклонения оси по двум изображениям и корректировать его.
Каждый объект класса Histograms хранит в себе следующую информацию:

- diff_angle - float угол отклонения оси, найденный по алгоритму
- part_threshold - float доля пикселей изображения, которую отсечет алгоритм
- threshold - float[2] значения, по которому происходит отсечение для каждого изображения
- part_bound - float доля пикселей изображения, которые попадут в рамку
- bound - int размер рамки
- first - Image первое обработанное изображение
- second - Image второе обработанное изображение
- indexes - size_t[4] индексы, которые показывают положение середины рамки для каждого изображения
- first_image - std::vector<float> углы изменения интенсивности первого изображения в рамке с учетом threshold
- second_image - std::vector<float> углы изменения интенсивности второго изображения в рамке с учетом threshold
- first_hist - std::deque<float> гистограмма по напрвлениям для первого изображения
- second_hist - std::deque<float> гистограмма по нраправлениям для второго изображения

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Histograms()

```
Histograms::Histograms(const std::string image_name_0, const
std::string image_name_1)
```

Создает объект класса Histograms.

Parameters

image_name_0	путь к первому изображению
image_name_1	путь ко второму изображению

Exceptions

std::exception	изображения должны иметь одинаковый размер

3.2.3 Member Function Documentation

3.2.3.1 SetPartThreshold()

```
void Histograms::SetPartThreshold(float part_threshold)
```

Позволяет изменить долю пикселей изображения, которую отсечет алгоритм

Parameters

part_threshold	новая доля

Exceptions

std::exception	0 < part_threshold < 1

3.2.3.2 SetPartBound()

```
void Histograms::SetPartBound(float part_bound)
```

Позволяет изменить долю пикселей изображения, которые попадут в рамку

Parameters

part_bound	новая доля

Exceptions

std::exception	0 < part_bound < 1

3.2.3.3 CheckPartThreshold()

```
float Histograms::CheckPartThreshold()
```

Позволяет узнать долю пикселей изображения, которую отсечет алгоритм

Returns Текущая доля

3.2.3.4 CheckPartBound()

```
float Histograms::CheckPartBound()
```

Позволяет узнать долю пикселей изображения, которые попадут в рамку
Returns Текущая доля

3.2.3.5 FindAngle()

```
float Histograms::FindAngle()
```

Вычисляет, если требуется, и показывает угол смещения оси с точностью до 0.5 градуса
Retuns найденный угол смещения оси

3.2.3.6 CorrectImage()

```
void Histograms::CorrectImages()
```

Сохраняет по заданному пути скорректированные изображения. Также позволяет самостоятельно задать угол корректировки
Parameters

first_name	путь, по которому надо сохранить первое скорректированное изображение
second_name	путь, по которому надо сохранить второе скорректированное изображение
angle	угол, на который необходимо выполнить поворот

Note

angle - необязательный параметр, если его не указать, то изображения будут скорректированы по углы, найденному алгоритмом

3.2.3.7 VisualizeHistograms()

```
void Histograms::VisualizeHistograms(char* name)
```

Создает файл с заданным именем, который затем можно визуализировать с помощью файла Отрисовка гистограммы.ipynb
Parameters

name	путь к файлу, в который надо будет сохранить гистограммы