# КОМАНДА “ИБРИУМ”

ОТЧЕТ ЗА НЕДЕЛЮ | 18.03

# Проделанная работа

1. Доработан алгоритм
    a. Улучшение определения границ теплопотерь
    b. Уменьшено количество “грязных” зон
    c. Восприятие части изображения, для игнорирования зоны с температурной шкалой
2. Доработана рекомендательная система
    a. Благодаря высокоточному алгоритму разработана новая система рекомендаций
3. Общий рефакторинг кода для дальнейшего переноса алгоритма в отдельную библиотеку.
    a. Вывод части кода в отдельные функции
4. Добавлен кроссплатформенный интерфейс
5. Отладка кода, внесение изменений для более стабильной работы

NumPy

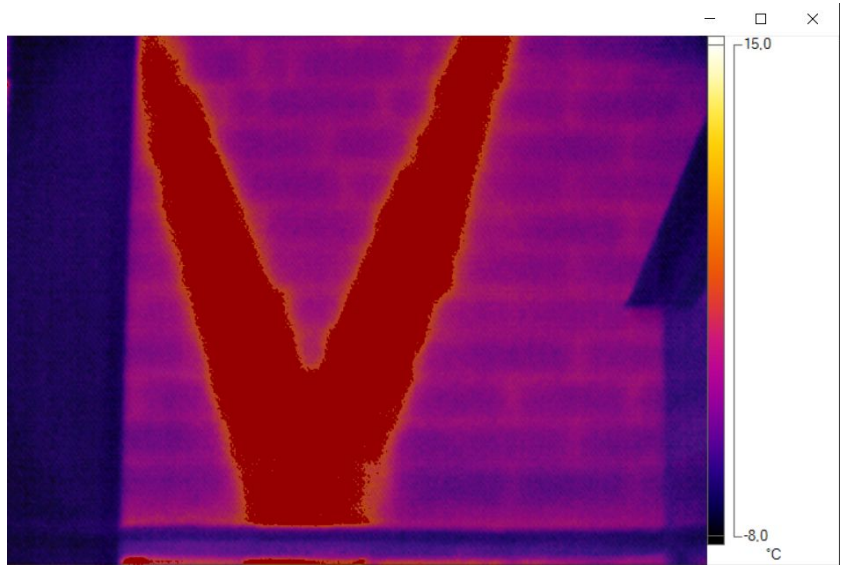
# Доработка алгоритма

Была проблема - алгоритм воспринимал все изображение, в том числе и удобную температурную шкалу.

Это решено путем использования конкатенации изображения с помощью библиотеки NumPY.

Уменьшено количество отдельных зон (пикселей с теплопотерями).

Повышена точность определения границ теплопотерь путем интеграции методов из плюсовых алгоритмов для определения границ.




```
recognize_function(img)
vis = np.concatenate((img, img_copy), axis=1)
```

Пример фото обработанного алгоритмом и блок кода с конкатенацией двух изображений.

# Один из явных примеров доработки алгоритма

Самым явным примером является интеграция оператора Кэнни в код.

Границы отмечаются там, где градиент изображения приобретает максимальное значение. Они могут иметь различное направление, поэтому алгоритм Кэнни использует четыре фильтра для обнаружения горизонтальных, вертикальных и диагональных ребер в размытом изображении.

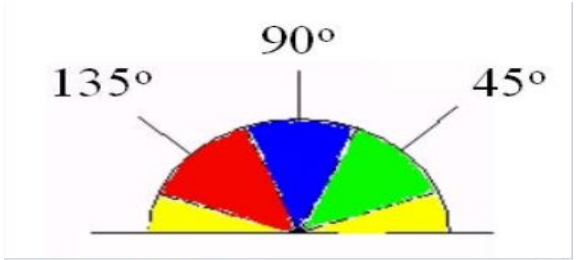
Угол направления вектора градиента округляется и может принимать такие значения: 0, 45, 90, 135.

Благодаря этому точность с 60-70%* повышена до 75-80%* на фото используемых в ходе отладки.

*-метод расчета производился с помощью изображения имеющего 36 окон на основании правильности зоны выделенных теплопотерь.

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$$

$$\mathbf{\Theta} = \operatorname{arctg}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right).$$



Направление градиента

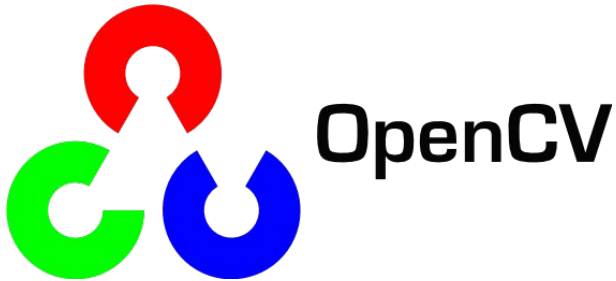
# Создан кроссплатформенный интерфейс

Данный интерфейс работает с любой системой на пк (Linux, Windows, MacOS)

Все благодаря инструментам OpenCV и Tkinter.

Визуальные элементы отображаются через собственные элементы текущей операционной системы, поэтому приложения, созданные с помощью Tkinter и OpenCV, выглядят так, как будто они принадлежат той платформе, на которой они работают.

Работа проверена на Windows 10 и Fedora Linux(w35+gnome)

# Возможности

1. Выбор файла для обработки пользователем
2. Выбор зоны теплопотерь с помощью перекрестия
3. Вывод рекомендации и изменения ее фона в тон цвету выделенной зоны
4. Вывод системных данных для отладки
5. Вывод файла до обработки для сравнения с обработанным изображением
6. Возможность отправки отчета с ошибкой
7. Отображение температурной шкалы
8. Отображение поддерживаемых форматов

Поддерживаемые форматы файлов:

BMP, DIB, JPEG-JPEG, JPG, JPE, JPEG 2000-jp2, PBM, PGM,

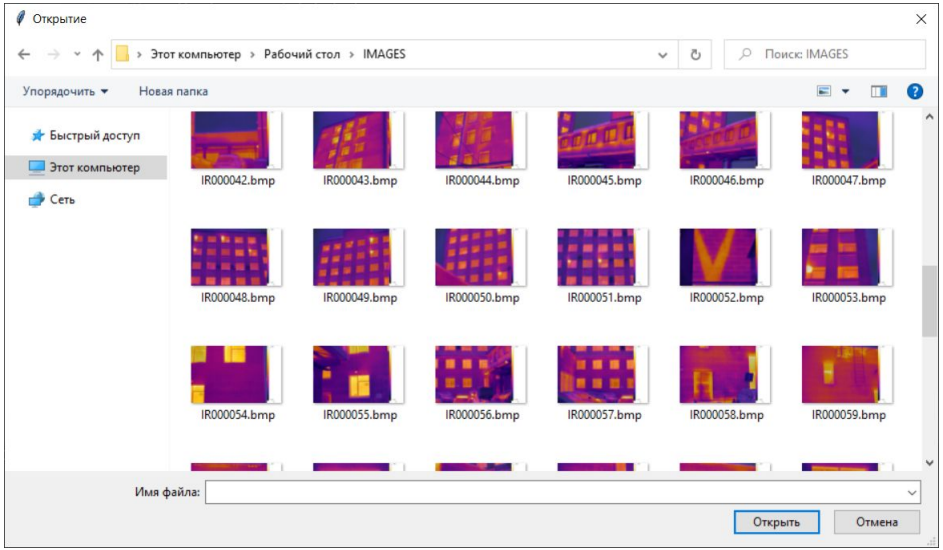
PPM, SR, RAS, TIFF - TIFF, TIF, OpenEXR HDR Picture-EXR

# Как работает?

На данный момент требуется лишь запустить файл программы и она начнет свою работу.

Перед пользователем сразу выйдет окно с открытием файла, который вы хотите выбрать для обработки

После будет выведено два окна.

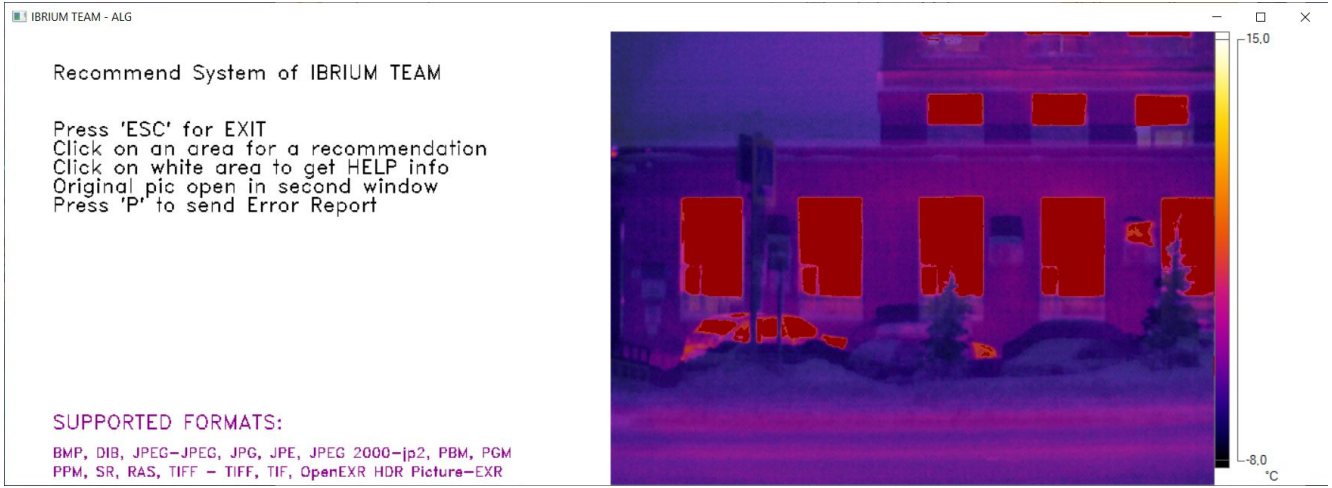


# Первое окно

Отрисовка производится с помощью OpenCV

Выводится обработанное изображение и инструкция по работе с приложением.

Отдельной графой идут поддерживаемые форматы.

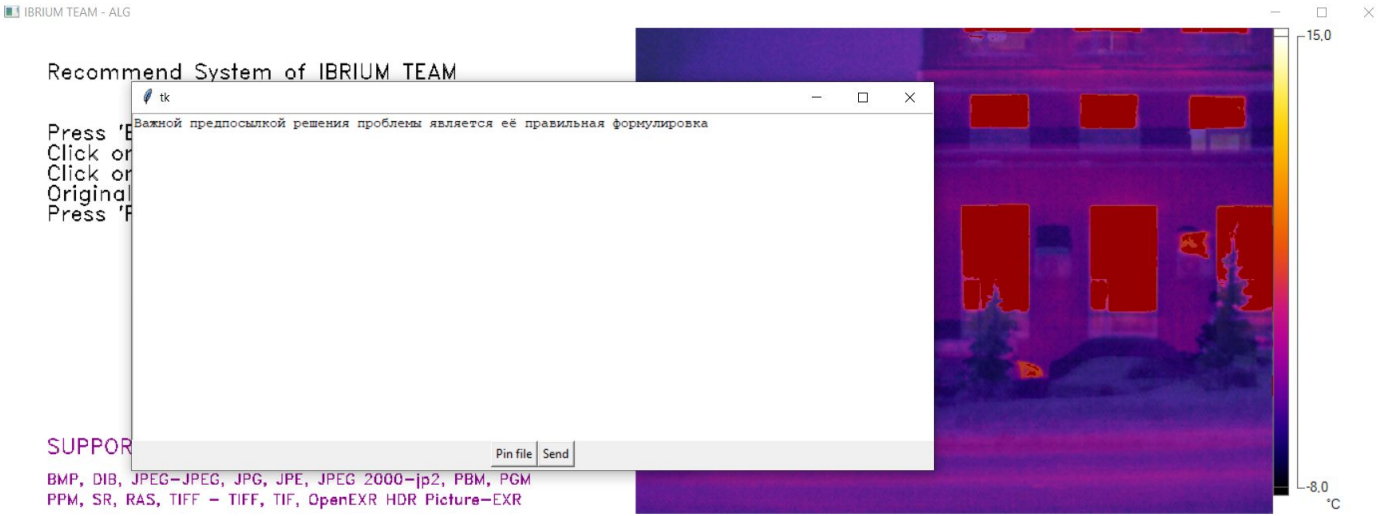


# Второе окно.

Содержит файл до обработки и выводит его для прямого сравнения.

# У вас проблема?  -СООБЩИТЕ НАМ ОБ ЭТОМ!

Если возникли проблемы связанные с работой алгоритма - вы можете написать нам и прикрепить скриншот в сообщении.



# Блок кода с окном Error Report.

Отрисовка происходит с помощью Tkinter

На данный момент реализовано прикрепление файла и возможность набора текста.

Метода для отправки куда либо нет, рассматриваются варианты реализации этого вопроса.

```
def insert_file():
    file_name = filedialog.askopenfilename()
    f = open(file_name)
    s = f.read()
    text.insert(1.0, s)
    f.close()

if key == ord('P') or key == ord('p'):
    root = Tk()
    text = Text(width=100, height=20)
    text.grid(columnspan=2)
    b1 = Button(text="Pin file", command=insert_file)
    b1.grid(row=1, sticky=E)
    b2 = Button(text="Send")
    b2.grid(row=1, column=1, sticky=W)
    root.mainloop()
```

# Тестирование и отладка.

За время отладки были устранены ошибки работы, но в случае непредвиденной ошибки из рекомендательной системы будет выдаваться “UNEXPECTED ERROR-4” (сопровождается системными данными).

```
case 4:
    cv2.putText(vis, "UNEXPECTED ERROR-4:", (50, 440), 2, 0.7, (255, 255, 255), font_face)
    cv2.putText(vis, "R:" + str(r) + "G:" + str(g) + "B:" + str(b),
        (50, 460), 2, 0.7, (255, 255, 255), font_face)
```

# Состояние рекомендательной системы

В данный момент рекомендательная система представляет компиляцию инструкции по использованию, рекомендательных статусов, системных данных и непредвиденной ошибки.

```
def recommend_sys(rec_status):
    font_face = cv2.FONT_HERSHEY_SIMPLEX
    match rec_status:
        case 0:
            cv2.putText(vis, "Recommend System of IBRIUM TEAM", (50, 50), 2, 0.7, (0, 0, 0), font_face)
            cv2.putText(vis, "Press 'ESC' for EXIT",
                        (50, 110), 2, 0.7, (0, 0, 0))
            cv2.putText(vis, "Click on an area for a recommendation",
                        (50, 130), 2, 0.7, (0, 0, 0))
            cv2.putText(vis, "Click on white area to get HELP info",
                        (50, 150), 2, 0.7, (0, 0, 0))
            cv2.putText(vis, "Original pic open in second window",
                        (50, 170), 2, 0.7, (0, 0, 0))
            cv2.putText(vis, "Press 'P' to send Error Report",
                        (50, 190), 2, 0.7, (0, 0, 0))
            cv2.putText(vis, "SUPPORTED FORMATS:",
                        (50, 420), 2, 0.7, (139, 0, 139))
            cv2.putText(vis, "BMP, DIB, JPEG-JPEG, JPG, JPE, JPEG 2000-jp2, PBM, PGM",
                        (50, 450), 2, 0.5, (139, 0, 139))
            cv2.putText(vis, "PPM, SR, RAS, TIFF - TIFF, TIF, OpenEXR HDR Picture-EXR",
                        (50, 470), 2, 0.5, (139, 0, 139))
        case 1:
            cv2.putText(vis, "Heat loss.",
```

# Статусы в системе

Основных статуса два

1. О наличии теплопотери и  методе решении ее.
2. Об отсутствии теплопотери.

Вывод сопутствующих системных данных.

```
case 1:
    cv2.putText(vis, "Heat loss.",
                (50, 50), 2, 0.7, (255, 255, 255), font_face)
    cv2.putText(vis, "Fill gaps and repair cladding defects",
                (50, 75), 2, 0.7, (255, 255, 255), font_face)
    cv2.putText(vis, "Carry out high-quality installation of windows",
                (50, 100), 2, 0.7, (255, 255, 255), font_face)
    cv2.putText(vis, "SYSTEM-INFO:", (50, 440), 2, 0.7, (255, 255, 255), font_face)
    cv2.putText(vis, "R:" + str(r) + "G:" + str(g) + "B:" + str(b),
                (50, 460), 2, 0.7, (255, 255, 255), font_face)
case 2:
    cv2.putText(vis, "No heat loss.", (50, 50), 2, 0.7, (255, 255, 255), font_face)
    cv2.putText(vis, "SYSTEM-INFO:", (50, 440), 2, 0.7, (255, 255, 255), font_face)
    cv2.putText(vis, "R:" + str(r) + "G:" + str(g) + "B:" + str(b),
                (50, 460), 2, 0.7, (255, 255, 255), font_face)
```

# Выдача статусов

Отрисовка окна и выдача статусов производится в одном блоке кода.

Статусы выдаются на основании цвета выделенной зоны, также здесь можно увидеть часть кода ответственную за вывод инструкции и блок для отлова ошибок.

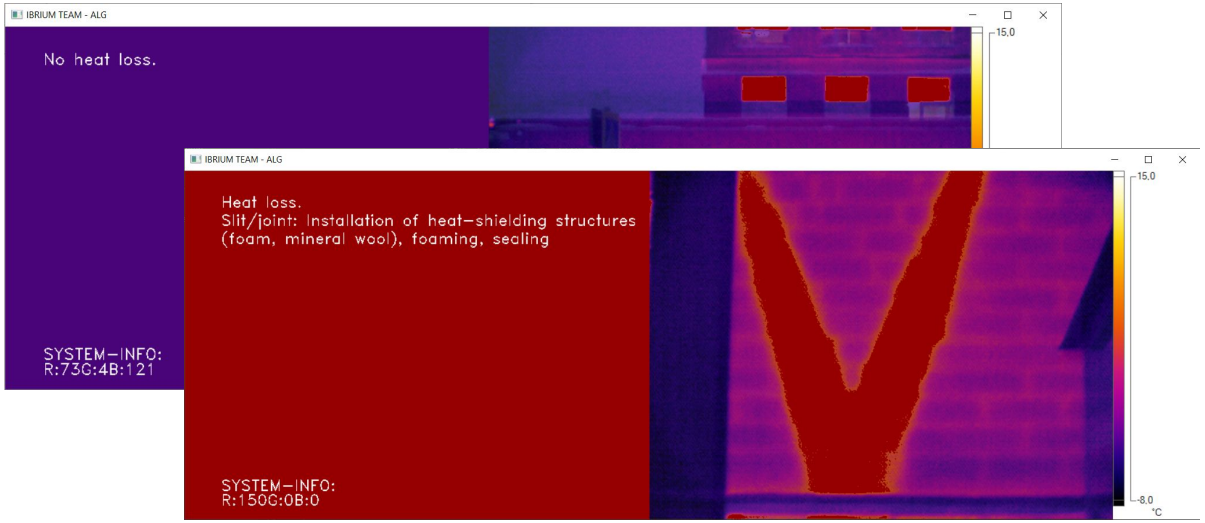
```
while True:
    cv2.imshow("IBRIUM TEAM - ALG", vis)
    cv2.namedWindow("IBRIUM TEAM - ALG")
    cv2.setMouseCallback("IBRIUM TEAM - ALG", color_function)
    if clicked:
        cv2.rectangle(vis, (0, 0), (640, 640), (b, g, r), -1)
        recommend_sys(1) if (r > 220 or r == 150) else recommend_sys(2)
        if r > 254 and b > 254 and g > 254:
            recommend_sys(0)
    else:
        recommend_sys(4)

    key = cv2.waitKey(1)
```

# Демонстрация работы

Вывод происходит в левом меню, которое окрашивается в тон выбранной зоны.

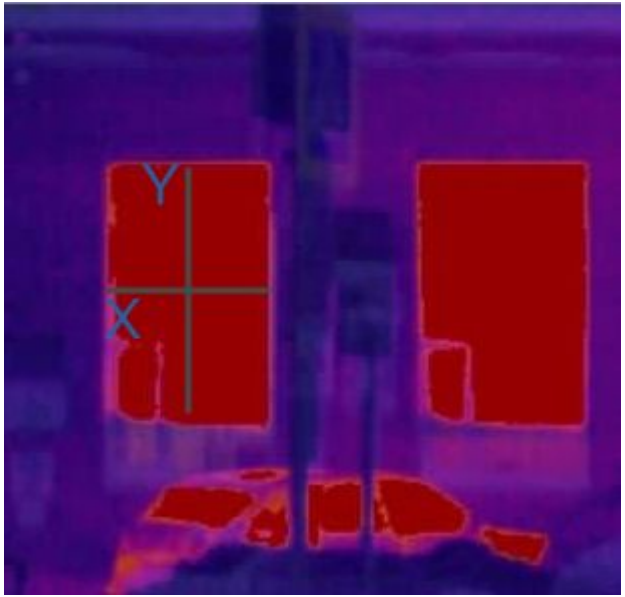
В нижней части можно увидеть системные данные.



# Концепция решения

Концепция решения  проблемы с определением окон и дверей как источника теплопотерей, конечно возможны как ложные срабатывание, так и не правильная фильтрация.

Будут высчитываться длина x и y по пикселям и сравниваться с усредненными данными для игнорирования их как источника теплопотерь.



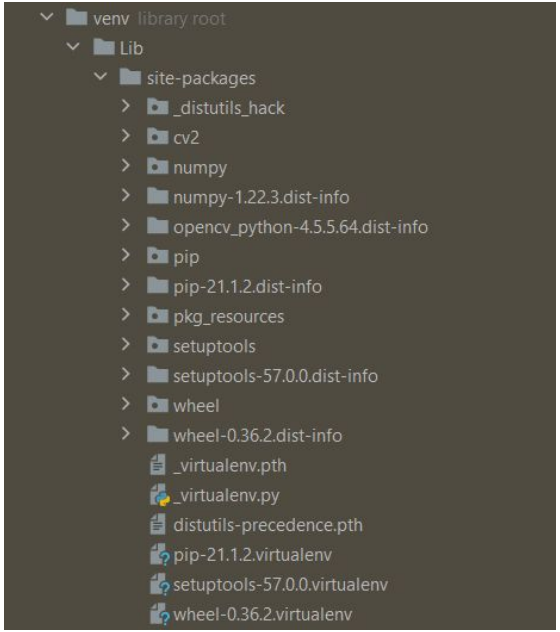
# Проблемы с портативностью

Такие системы, как MacOS, Linux по умолчанию уже содержат Python. Но к примеру в Windows нет.

Так же в проекте используется 3 библиотеки, только Tkinter входит в Python. Но библиотеки OpenCV и NumPy сторонние.

На данный момент используется отдельный venv, созданный мной, в который установлено все необходимое для работы алгоритма.

Конечно, можно использовать и его для запуска, но пользователю будет удобнее приложение, которое будет ставится в систему, либо работать портативно(Windows reference)




venv проекта

# Дальнейшие шаги

Дальнейшие шаги :

1. Добавление портативности программе
2. Улучшение существующего интерфейса
3. Доработка алгоритма для игнорирования окон как источника теплопотерь
4. Вынесение алгоритма в отдельную библиотеку для удобной работы с его функциям, возможной публикации на GitHub, PyPI.

# Спасибо за внимание.