Детектор объектов

НИР, весна 2016

Артем Лобанов
Даниил Смирнов
Иван Веселов
Илья Шевченко

Руководители: Антон Крыщенко, Богдан Бугаев

Академический университет

31 мая 2016 г.

Задачи

- Обеспечить загрузку входных данных
- Визуализировать результаты поиска объекта
- Склеить написанный код в единый проект

Процесс разработки

Написаны:

- модуль для загрузки ключевых кадров
- интерфейс командной строки для работы с ними
- рендер объекта при помощи OpenGL
- пользовательский интерфейс для удобства работы

Загрузка ключевых кадров

- Сканирует директорию и ищет в ней объект и ключевые кадры в заданном формате
- Считывает их
- Выдает класс, содержащий информацию о ключевом кадре — KeyFrame

Структура KeyFrame

- Положение и поворот камеры
- Внутренние параметры камеры
- Положение и поворот объекта

GUI. Структура

- Панель инструментов с кнопками для запуска алгоритмов поиска объекта
- Виджет с вкладками, содержащими ключевые кадры

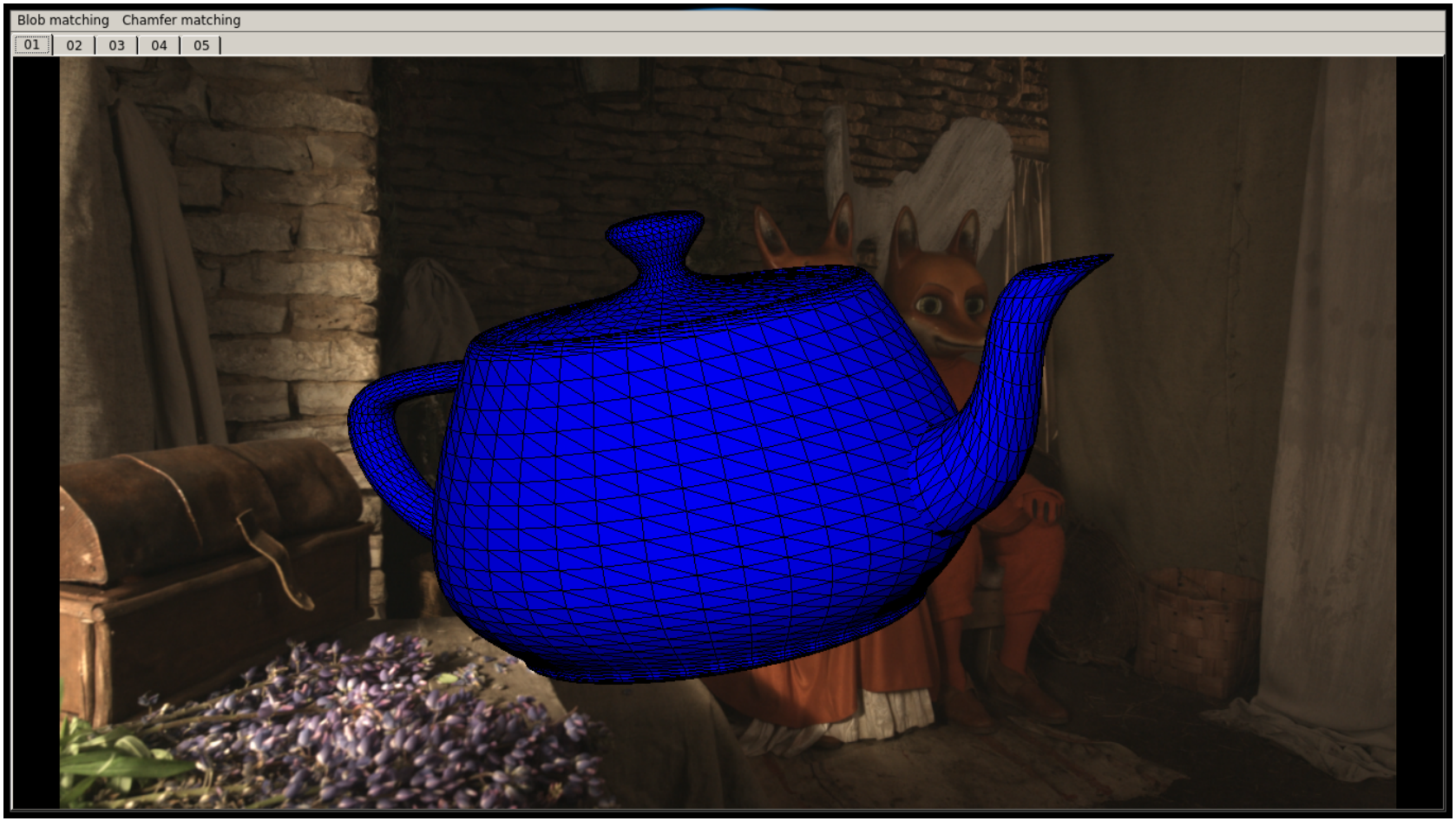
Структура вкладки:

- Картинка
- Модель, отрисованная поверх нее в виде сетки

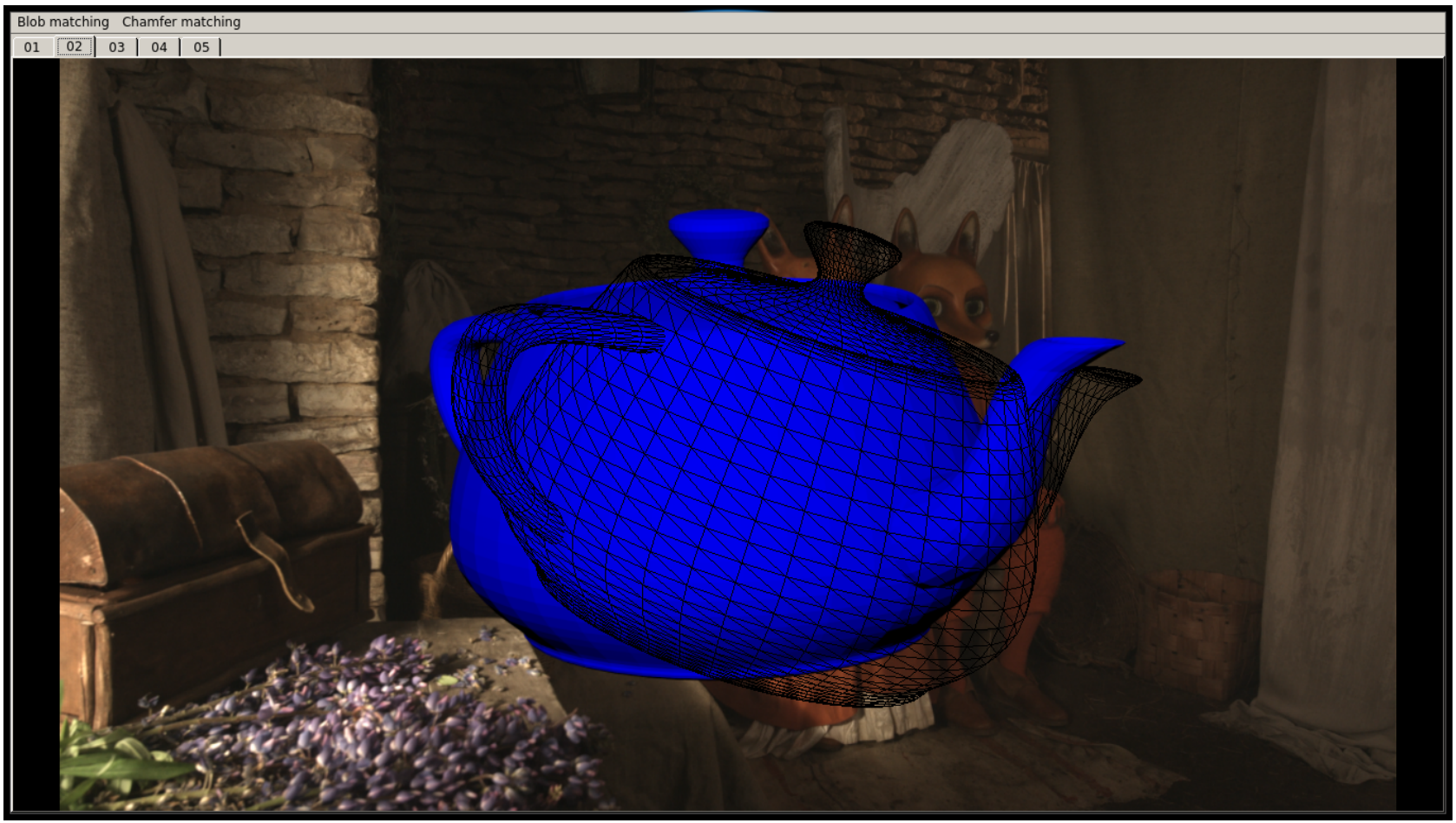
Технологии

- PySide (Qt) - GUI
- OpenGL - Отрисовка модели объекта на ключевом кадре

GUI



GUI



Склейка кода

Структура проекта

- main
- loading
- matching
  - chamfer
  - blob
- gui
  - main window
  - key frame previewer

Результаты

- Получился относительно цельный проект
- Все работает и шуршит
- Я разобрался с OpenGL и его использованием в Qt
- Получил опыт разработки в команде и работы с Github

Геометрия и 3D-объекты

- Придумать внутреннее представление 3D-моделей

Геометрия и 3D-объекты

- Придумать внутреннее представление 3D-моделей
- Научиться читать файлы в формате OBJ
- `http://www.martinreddy.net/gfx/3d/OBJ.spec`

Геометрия и 3D-объекты

- Придумать внутреннее представление 3D-моделей
- Научиться читать файлы в формате OBJ
- `http://www.martinreddy.net/gfx/3d/OBJ.spec`
- Разобраться с моделью камеры с точечной диафрагмой
- Multiple View Geometry in computer vision (Richard Hartley, Andrew Zisserman)

Геометрия и 3D-объекты

- Придумать внутреннее представление 3D-моделей
- Научиться читать файлы в формате OBJ
- `http://www.martinreddy.net/gfx/3d/OBJ.spec`
- Разобраться с моделью камеры с точечной диафрагмой
- Multiple View Geometry in computer vision (Richard Hartley, Andrew Zisserman)
- Научиться по информации об объекте и камере находить прообраз точки на картинке

Геометрия и 3D-объекты

- Придумать внутреннее представление 3D-моделей
- Научиться читать файлы в формате OBJ
- `http://www.martinreddy.net/gfx/3d/OBJ.spec`
- Разобраться с моделью камеры с точечной диафрагмой
- Multiple View Geometry in computer vision (Richard Hartley, Andrew Zisserman)
- Научиться по информации об объекте и камере находить прообраз точки на картинке
- Научиться по информации об объекте и камере находить контур проекции, видимые рёбра

Формат OBJ

- Это довольно простой, но популярный формат описания геометрических объектов

Формат OBJ

- Это довольно простой, но популярный формат описания геометрических объектов
- `v 0.123 0.234 0.345 1.0 # так описывается точка`
- `f 6/4/1 3/5/3 7/6/5 # так может описываться грань`
- `f 6 3 7 # а это то, что нам в действительности от неё нужно`

Внутренее представление

- Камера задаётся матрицей перехода из мировой системы координат в её систему координат и матрице проекции

Внутренее представление

- Камера задаётся матрицей перехода из мировой системы координат в её систему координат и матрице проекции
- Объект задаётся набором точек и граней в собственной системе координат и матрицей перехода в мировую систему координат
- Все геометрические расчёты с участием объекта (поиск пересечений объекта с лучами, поиск видимых рёбер и т.д.) производятся в системе отсчёта объекта. Это удобно т.к., например, позволяет предподсчитать нормали граней.

Честная геометрия и эвристики

- Для поиска прообраза точки луч, соответствующий всем точкам, которые проектируются в данную точку, пересекают с моделью объекта. Среди всех пересечений находится ближайшее к камере

Честная геометрия и эвристики

- Для поиска прообраза точки луч, соответствующий всем точкам, которые проектируются в данную точку, пересекают с моделью объекта. Среди всех пересечений находится ближайшее к камере
- "Видимыми"гранями считаются те, которые повёрнуты к камере лицевой стороной

Честная геометрия и эвристики

- Для поиска прообраза точки луч, соответствующий всем точкам, которые проектируются в данную точку, пересекают с моделью объекта. Среди всех пересечений находится ближайшее к камере
- "Видимыми"гранями считаются те, которые повёрнуты к камере лицевой стороной
- "Видимыми"рёбрами считаются те, соседние грани которых видимы

Честная геометрия и эвристики

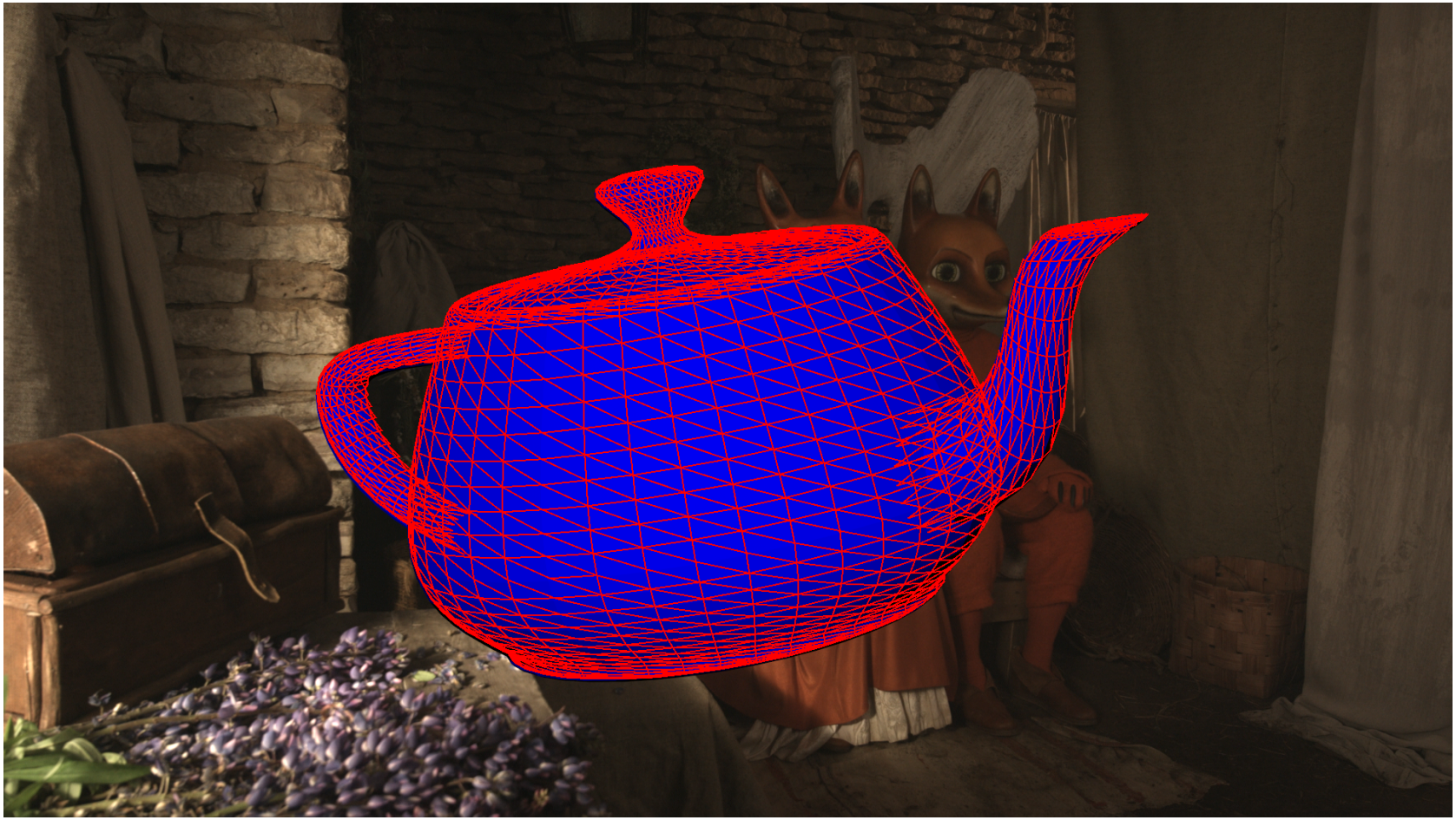
- Для поиска прообраза точки луч, соответствующий всем точкам, которые проектируются в данную точку, пересекают с моделью объекта. Среди всех пересечений находится ближайшее к камере
- "Видимыми"гранями считаются те, которые повёрнуты к камере лицевой стороной
- "Видимыми"рёбрами считаются те, соседние грани которых видимы
- Ребро считается лежащим на границе проекции, если видна ровно одна из соседних граней

Честная геометрия и эвристики

- Для поиска прообраза точки луч, соответствующий всем точкам, которые проектируются в данную точку, пересекают с моделью объекта. Среди всех пересечений находится ближайшее к камере
- "Видимыми"гранями считаются те, которые повёрнуты к камере лицевой стороной
- "Видимыми"рёбрами считаются те, соседние грани которых видимы
- Ребро считается лежащим на границе проекции, если видна ровно одна из соседних граней
- Находить видимые грани можно одним проходом по списку граней с $O(1)$ вычислений для каждой грани
- Эти эвристики абсолютно точно работают на выпуклых объектах и довльно неплохо для близких к выпуклым

Работа эвристик

Работа эвристик



numpy

- Библиотека numpy позволяет создавать массивы примитивных типов в python
- В нашем проекте используется для компактного хранения точек и для ускорения операций, применяемых к большому количеству точек

Цель

- Цель: определить положение объекта на изображении

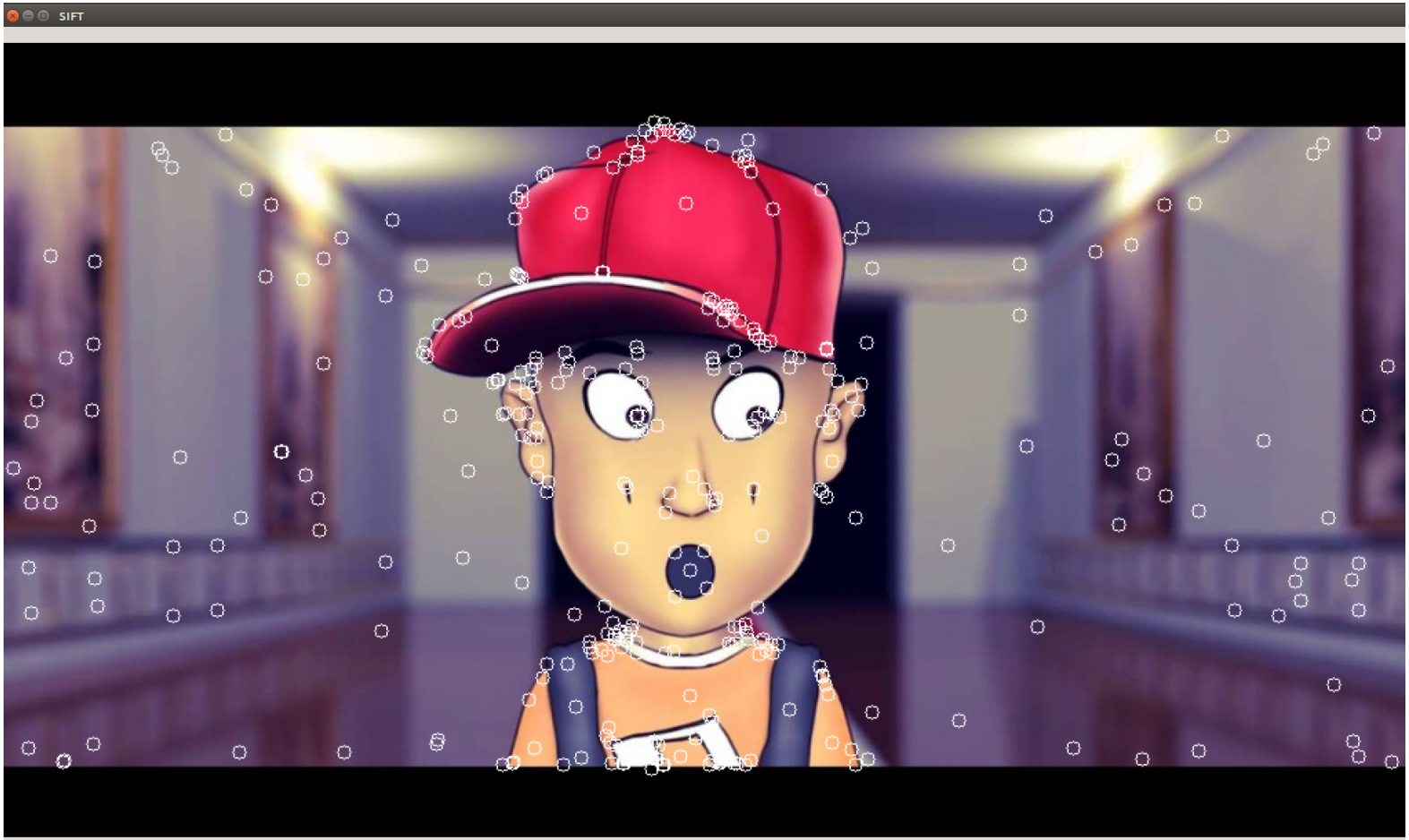
Задачи

- Определить ключевые точки на ключевом кадре и изображении.
- Репроецировать точки ключевого кадра на объект.
- Сопоставить ключевые точки. Получить соотвествие точек на изображении и точек на объекте.
- Определить положение объекта по соответствию 2D и 3D точек

Определение ключевых точек

- Ключевые точки - это точки, которые обладают некоторыми важными свойствами
- Важнейшие свойства: устойчивость к преобразованиям объекта, точность локализации
- Для данной задачи были выбраны ключевые точки SIFT (scale-invariant feature transform)

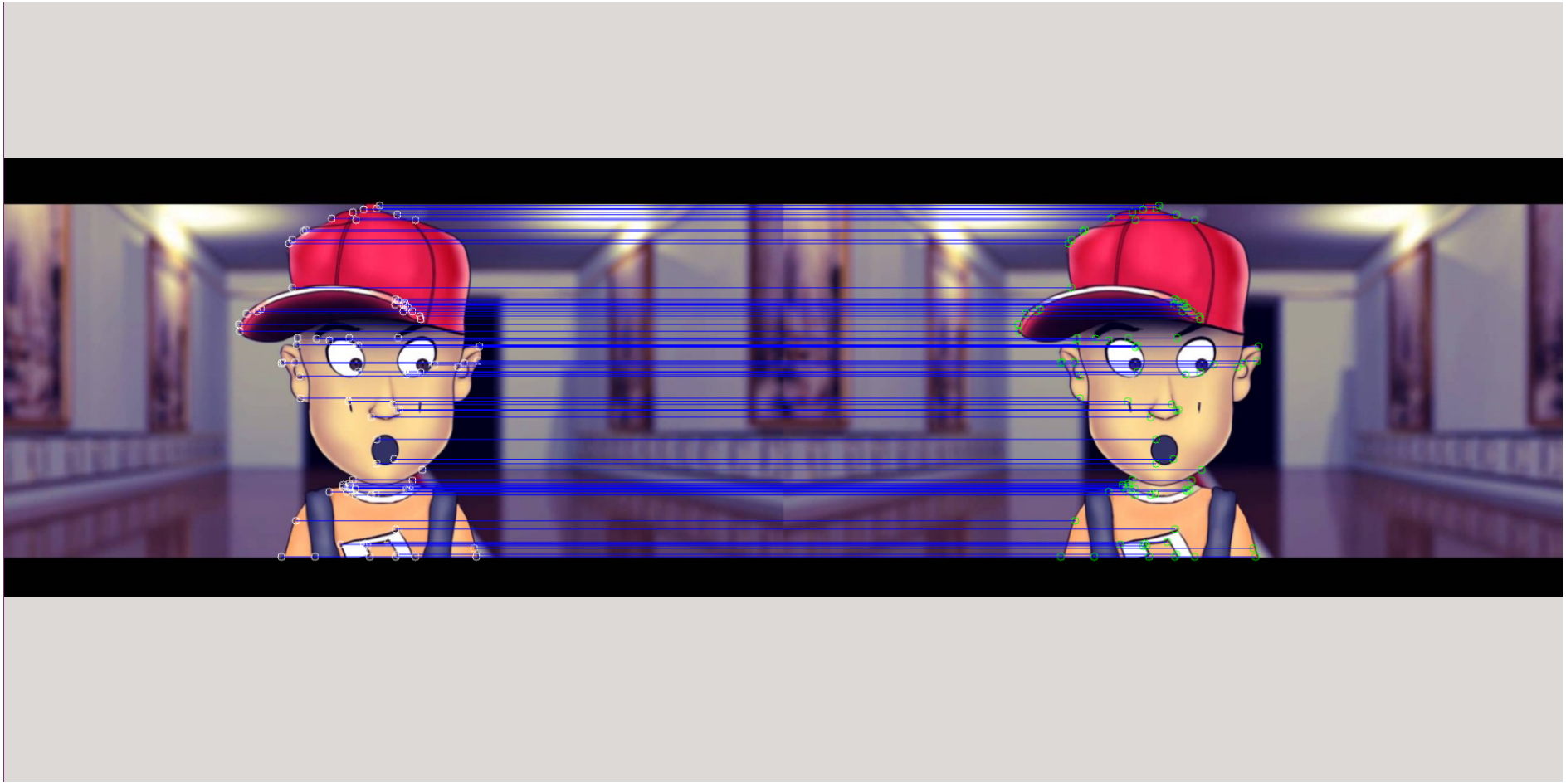
Ключевые точки: пример



Сопоставление ключевых точек

- Вычисление дескриптора, то есть объекта, который описывает ключевую точку.
- Дескриптором для ключевых точек SIFT является вектор из градиентов в ближайших точках.
- Сопоставление точек расстояние между дескрипторами которых минимально.
- Наивный метод.

Сопоставление ключевых точки: пример



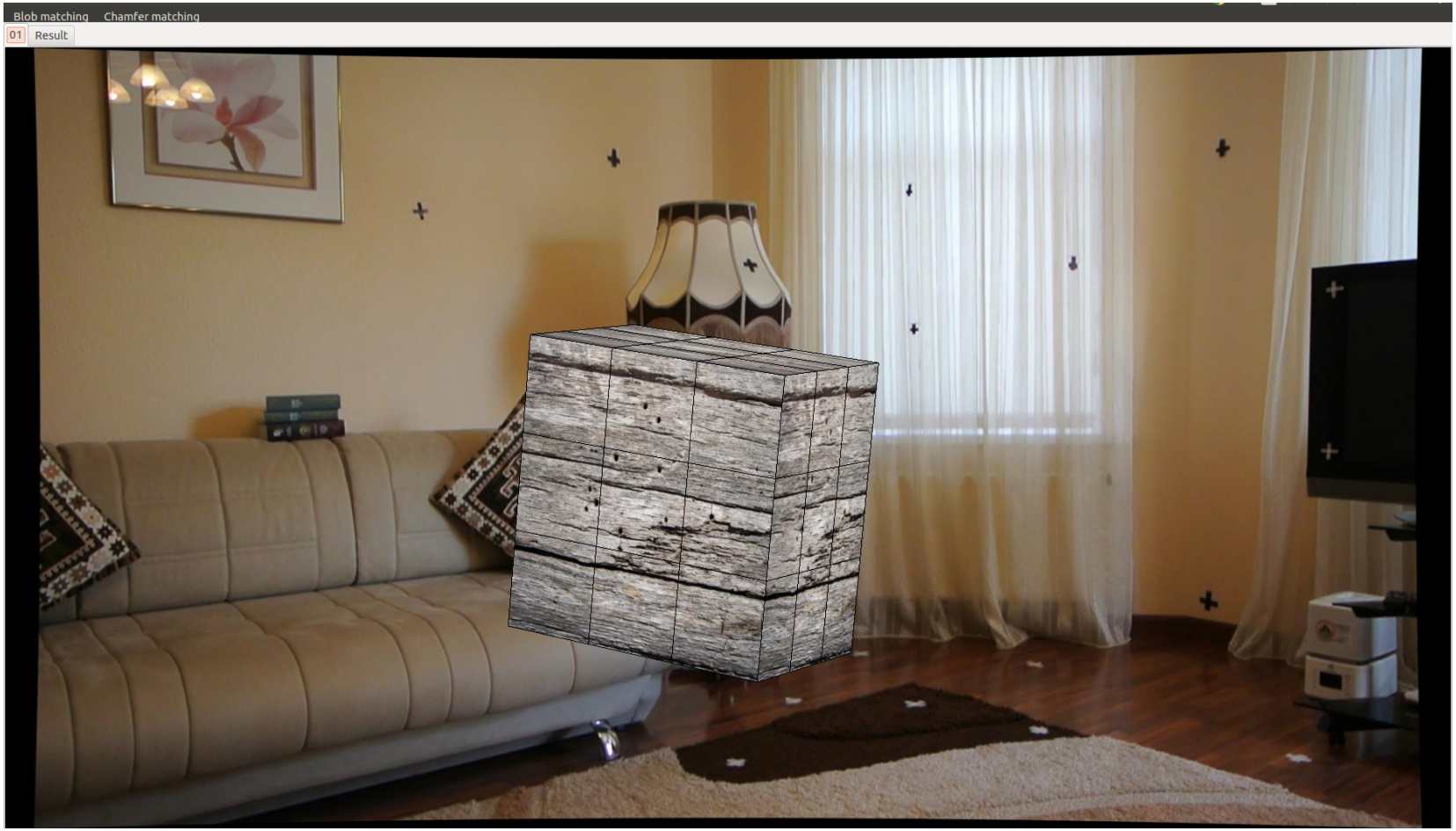
PnP

- Задача PnP состоит в определении матрицы трансформации по соотвествию двухмерных и трёхмерных точек
- Проблема в том, что из-за неточности соответствия ключевых точек есть точки-выбросы. Для наборов с такими точками методы точного решения PnP дают очень плохие результаты
- Решение проблемы - RANSAC

RANSAC

- Random sample consensus
- Метод выбирает несколько случайных наборов точек размера пять. Вероятность того, что ни одна из этих точек не является выбросом достаточна велика если количество выбросов не более половины.
- По пяти точкам можно построить решение задачи PnP.
- С помощью полученной матрицы трансформации трёхмерные точки проецируются на плоскость.
- Точки-проекции, лежащие в пределах допустимой погрешности от исходных точек изображения считаются невыбросами
- Среди всех наборов выбирается тот, который дал наибольшее количество невыбросов.

Финальный результат: ключевой кадр



Финальный результат: изображение

Результаты

- Изучил различные типы ключевых точек и методы их выделения.
- Изучил метод RANSAC.
- Научился пользоваться библиотекой OpenCV в Python.
- Попробовал себе в командной разработке с использованием git.

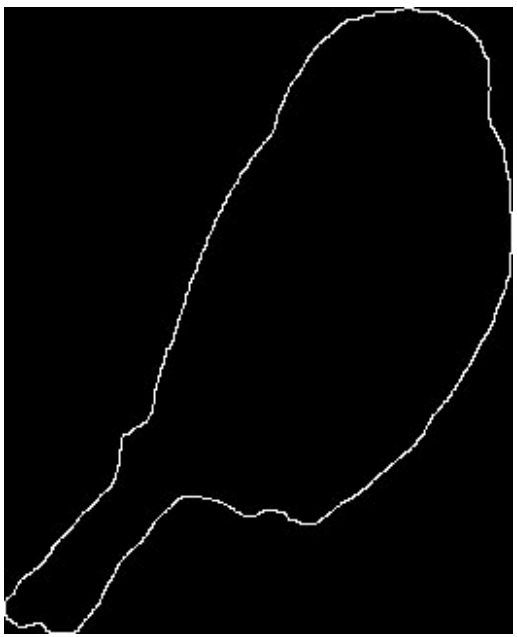
Chamfer matching

- Цель: научиться по контуру предмета находить этот предмет на изображении.

Chamfer matching

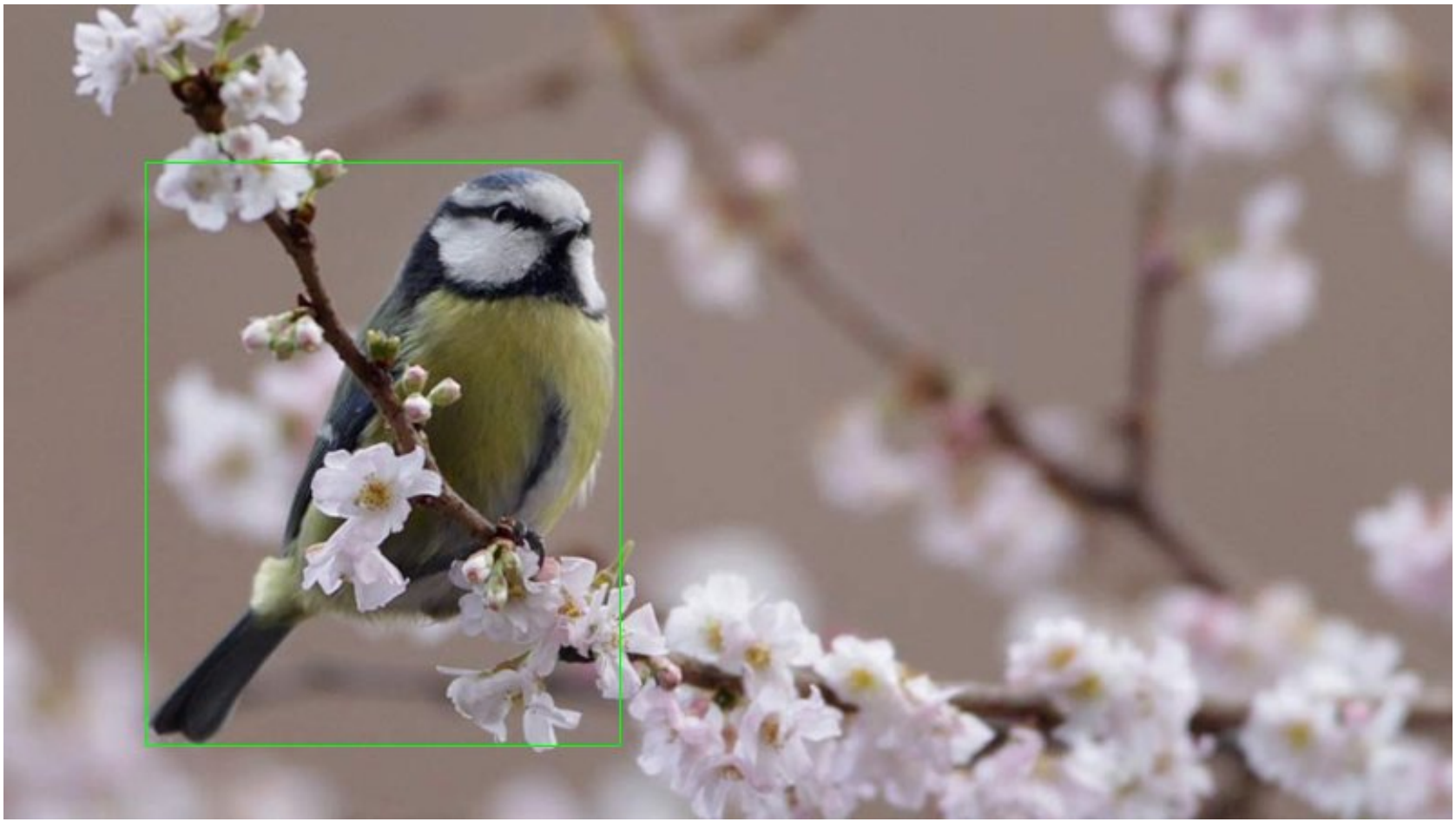
- Цель: научиться по контуру предмета находить этот предмет на изображении.
- Рассмотренный подход к решению описан в статье Ming-Yu Liu, Oncel Tuzel, Ashok Veeraraghavan, Rama Chellappa. Fast Directional Chamfer Matching

Chamfer matching



Chamfer matching

Chamfer matching



Алгоритм

- С помощью алгоритма Кэнни найти на изображении границы между объектами.

Алгоритм

- С помощью алгоритма Кэнни найти на изображении границы между объектами.
- Представить границы на изображении и контур предмета в виде набора отрезков. Такое представление удобно для дальнейших оптимизаций.

Алгоритм

- С помощью алгоритма Кэнни найти на изображении границы между объектами.
- Представить границы на изображении и контур предмета в виде набора отрезков. Такое представление удобно для дальнейших оптимизаций.
- Для каждого возможного наложения контура на изображение посчитать эвристическую функцию отклонения.

Алгоритм

- С помощью алгоритма Кэнни найти на изображении границы между объектами.
- Представить границы на изображении и контур предмета в виде набора отрезков. Такое представление удобно для дальнейших оптимизаций.
- Для каждого возможного наложения контура на изображение посчитать эвристическую функцию отклонения.
- Среди всех наложений выбрать одно с минимальным значением функции.

Алгоритм Кэнни

- Алгоритм Кэнни получает на вход изображение, как функцию интенсивности от двух переменных.

Алгоритм Кэнни

- Алгоритм Кэнни получает на вход изображение, как функцию интенсивности от двух переменных.
- Для каждого пикселя вычисляется приближенное значение градиента этой функции в данном пикселе.

Алгоритм Кэнни

- Алгоритм Кэнни получает на вход изображение, как функцию интенсивности от двух переменных.
- Для каждого пикселя вычисляется приближенное значение градиента этой функции в данном пикселе.
- Пиксели с высоким значением модуля градиента объявляются граничными.

Алгоритм Кэнни

- Алгоритм Кэнни получает на вход изображение, как функцию интенсивности от двух переменных.
- Для каждого пикселя вычисляется приближенное значение градиента этой функции в данном пикселе.
- Пиксели с высоким значением модуля градиента объявляются граничными.
- В работе был использован готовый алгоритм из библиотеки OpenCV.

Алгоритм Кэнни, пример работы

Алгоритм Кэнни, пример работы

Линеаризация границ

- Алгоритм Кэнни находит набор пикселей, которые являются граничными.

Линеаризация границ

- Алгоритм Кэнни находит набор пикселей, которые являются граничными.
- Необходимо приблизить этот набор пикселей отрезками. Дополнительно необходимо, чтобы все отрезки имели с осью координат один из q равномерно расположенных на $[0, \pi)$ углов. Число q должно быть небольшое.

Линеаризация границ

- Алгоритм Кэнни находит набор пикселей, которые являются граничными.
- Необходимо приблизить этот набор пикселей отрезками. Дополнительно необходимо, чтобы все отрезки имели с осью координат один из q равномерно расположенных на $[0, \pi)$ углов. Число q должно быть небольшое.
- Авторы статьи предлагают воспользоваться вариацией алгоритма RANSAC (Random sample consensus).

RANSAC

- Алгоритм Кэнни для каждого пикселя находит градиент интенсивности в этом пикселе. Если пиксель граничный, то градиент является нормалью к кривой границы.

RANSAC

- Алгоритм Кэнни для каждого пикселя находит градиент интенсивности в этом пикселе. Если пиксель граничный, то градиент является нормалью к кривой границы.
- Таким образом каждый пиксель p задает прямую l, проходящую через него перпендикулярно нормали. Так как необходимо найти отрезок с одним из заданных углов, то считается, что p задает прямую, угол которой наименее всего отличается от угла l.

RANSAC

- Алгоритм Кэнни для каждого пикселя находит градиент интенсивности в этом пикселе. Если пиксель граничный, то градиент является нормалью к кривой границы.
- Таким образом каждый пиксель p задает прямую l, проходящую через него перпендикулярно нормали. Так как необходимо найти отрезок с одним из заданных углов, то считается, что p задает прямую, угол которой наименее всего отличается от угла l.
- Среди соседей p отбираются пиксели, которые не слишком сильно удалены от прямой l. Этот процесс продолжается по цепочке.

RANSAC

- Алгоритм Кэнни для каждого пикселя находит градиент интенсивности в этом пикселе. Если пиксель граничный, то градиент является нормалью к кривой границы.
- Таким образом каждый пиксель p задает прямую l, проходящую через него перпендикулярно нормали. Так как необходимо найти отрезок с одним из заданных углов, то считается, что p задает прямую, угол которой наименее всего отличается от угла l.
- Среди соседей p отбираются пиксели, которые не слишком сильно удалены от прямой l. Этот процесс продолжается по цепочке.
- Выбранные пиксели объявляются поддержкой гипотезы пикселя p.

RANSAC

- Алгоритм Кэнни для каждого пикселя находит градиент интенсивности в этом пикселе. Если пиксель граничный, то градиент является нормалью к кривой границы.
- Таким образом каждый пиксель p задает прямую l, проходящую через него перпендикулярно нормали. Так как необходимо найти отрезок с одним из заданных углов, то считается, что p задает прямую, угол которой наименее всего отличается от угла l.
- Среди соседей p отбираются пиксели, которые не слишком сильно удалены от прямой l. Этот процесс продолжается по цепочке.
- Выбранные пиксели объявляются поддержкой гипотезы пикселя p.
- На очередном шаге алгоритма случайным образом выбирается небольшое количество пикселей, для каждого из которых вычисляется поддержка.

RANSAC

- Алгоритм Кэнни для каждого пикселя находит градиент интенсивности в этом пикселе. Если пиксель граничный, то градиент является нормалью к кривой границы.
- Таким образом каждый пиксель p задает прямую l, проходящую через него перпендикулярно нормали. Так как необходимо найти отрезок с одним из заданных углов, то считается, что p задает прямую, угол которой наименее всего отличается от угла l.
- Среди соседей p отбираются пиксели, которые не слишком сильно удалены от прямой l. Этот процесс продолжается по цепочке.
- Выбранные пиксели объявляются поддержкой гипотезы пикселя p.
- На очередном шаге алгоритма случайным образом выбирается небольшое количество пикселей, для каждого из которых вычисляется поддержка.
- Пиксель с наибольшей поддержкой и всеми пикселями, которые его поддержали, заменяются на отрезок.

RANSAC, пример работы

RANSAC, пример работы

Линеаризация контура

- К контуру предмета можно применить аналогичную идею.

Линеаризация контура

- К контуру предмета можно применить аналогичную идею.
- Однако контур не является функцией интенсивности, и на нем нельзя запустить алгоритм Кэнни.

Линеаризация контура

- К контуру предмета можно применить аналогичную идею.
- Однако контур не является функцией интенсивности, и на нем нельзя запустить алгоритм Кэнни.
- Как следствие, для каждого пикселя контура не известно направление по которому нужно откладывать прямую.

Линеаризация контура

- К контуру предмета можно применить аналогичную идею.
- Однако контур не является функцией интенсивности, и на нем нельзя запустить алгоритм Кэнни.
- Как следствие, для каждого пикселя контура не известно направление по которому нужно откладывать прямую.
- Поэтому нужно для каждого пикселя перебрать направление среди q зафиксированных.

Эвристическая функция

- Пусть p пиксель изображения, а E – множество граничных пикселей на изображении. Тогда обозначим $d(p) = \min\limits_{e \in E}\{dist(p, e)\}$, где $dist$ – евклидов расстояни между точками.

Эвристическая функция

- Пусть p пиксель изображения, а E – множество граничных пикселей на изображении. Тогда обозначим $d(p) = \min\limits_{e \in E}\{dist(p, e)\}$, где $dist$ – евклидов расстояни между точками.
- Чем дальше пиксель из контура от граничных пикселей изображения при наложении, тем хуже наложение.

Эвристическая функция

- Пусть p пиксель изображения, а E – множество граничных пикселей на изображении. Тогда обозначим $d(p) = \min\limits_{e \in E}\{dist(p, e)\}$, где $dist$ – евклидов расстояни между точками.
- Чем дальше пиксель из контура от граничных пикселей изображения при наложении, тем хуже наложение.
- Поэтому за эвристическую функцию принимают $f = \sum\limits_{p} d(p)$, где p пробегает все пиксели контура при его наложении на изображение.

Оптимизации

- Разумным подходом является заранее подсчитать для каждого пикселя p величину $d(p) = \min\limits_{e \in E}\{dist(p, e)\}$.

Оптимизации

- Разумным подходом является заранее подсчитать для каждого пикселя p величину $d(p) = \min\limits_{e \in E}\{dist(p, e)\}$.
- Данная задача решается методом динамического программирования с применением идеи convex hull trick.

Оптимизации

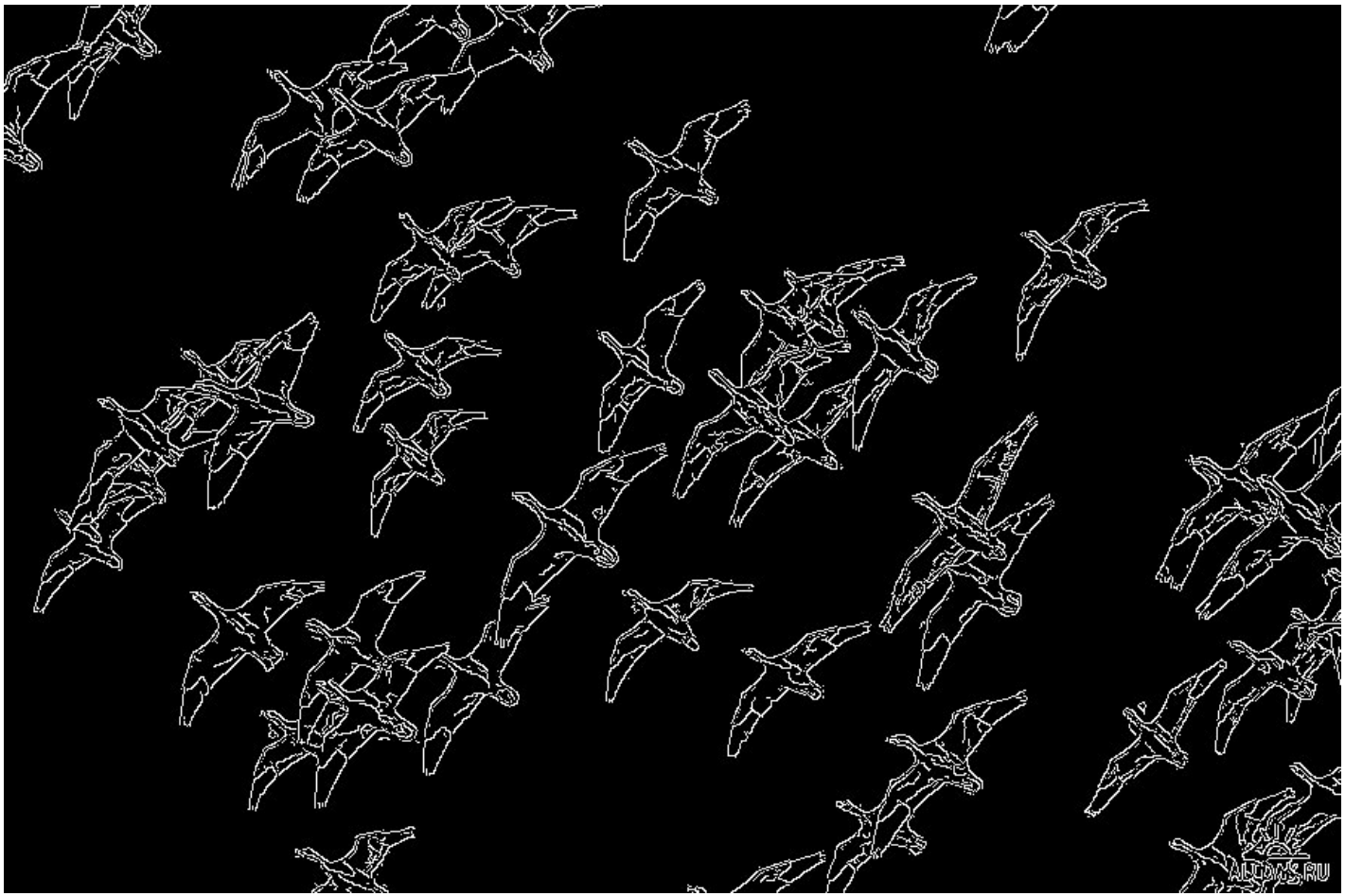
- Разумным подходом является заранее подсчитать для каждого пикселя p величину $d(p) = \min\limits_{e \in E}\{dist(p, e)\}$.
- Данная задача решается методом динамического программирования с применением идеи convex hull trick.
- Для каждого из q направлений можно посчитать массивы частичных сумм. Это позволит за $\mathcal{O}(1)$ времени суммировать $d(p)$ для всех p из одного отрезка.

Другой пример

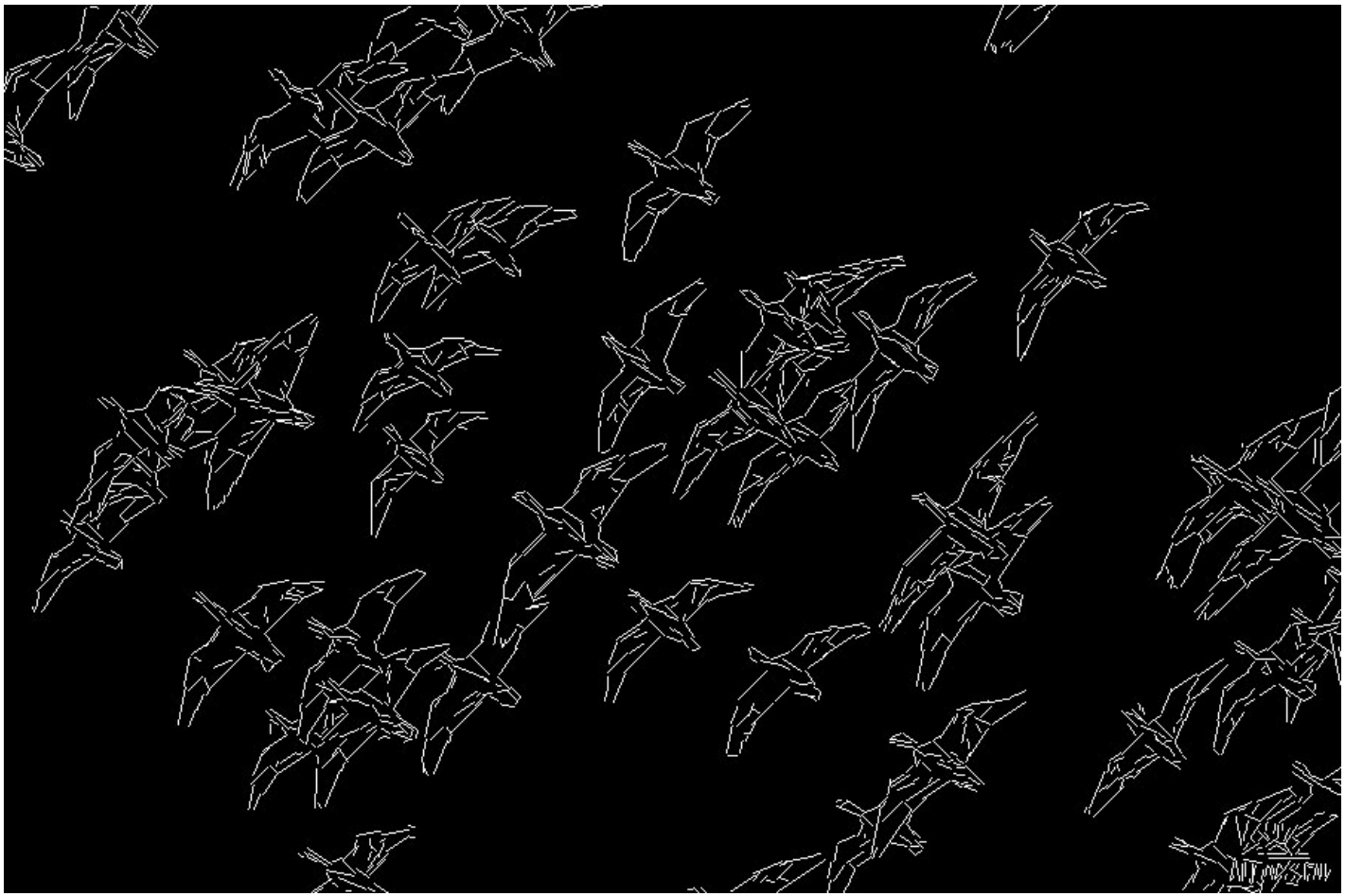
Изображения контуров увеличены.

Другой пример

Другой пример



Другой пример



Другой пример

Что еще можно сделать?

- Авторы статьи Fast Directional Chamfer Matching предлагают немного изменить классическую эвристическую функцию.

  Вместо $f = \sum_p d(p)$, предлагается $f' = \sum_p \left[d(p) + \lambda angle(p) \right]$, где $angle(p)$ разница между направлениями (углами) градиента в ближайшем к p граничном пикселе и пикселе контура, который при наложении перешел в p, а λ коэффициента влияния нового слагаемого.

Что еще можно сделать?

- Авторы статьи Fast Directional Chamfer Matching предлагают немного изменить классическую эвристическую функцию.

  Вместо $f = \sum_p d(p)$, предлагается $f' = \sum_p \left[d(p) + \lambda angle(p) \right]$, где $angle(p)$ разница между направлениями (углами) градиента в ближайшем к p граничном пикселе и пикселе контура, который при наложении перешел в p, а λ коэффициента влияния нового слагаемого.
- Провести исследование на большом количестве тестов, чтобы понять, можно ли где-то что-то улучшить.

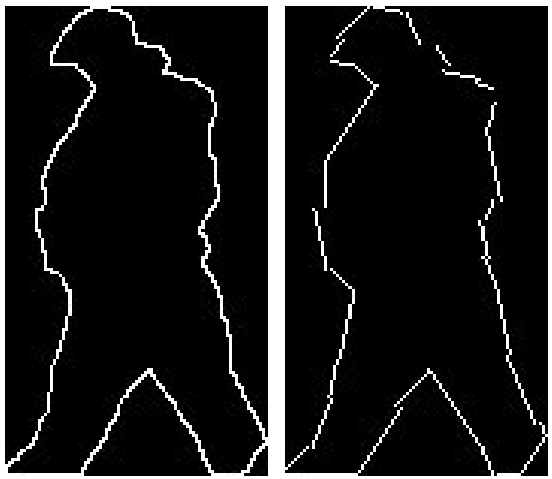
Недостатки Chamfer matching

- Необходимость в точном контуре.
  Если контур будет неправильно повернут или будет иметь неправильный размер, то Chamfer matching не сработает.

Недостатки Chamfer matching

- Необходимость в точном контуре.
  Если контур будет неправильно повернут или будет иметь неправильный размер, то Chamfer matching не сработает.
- Неустойчивость к помехам.

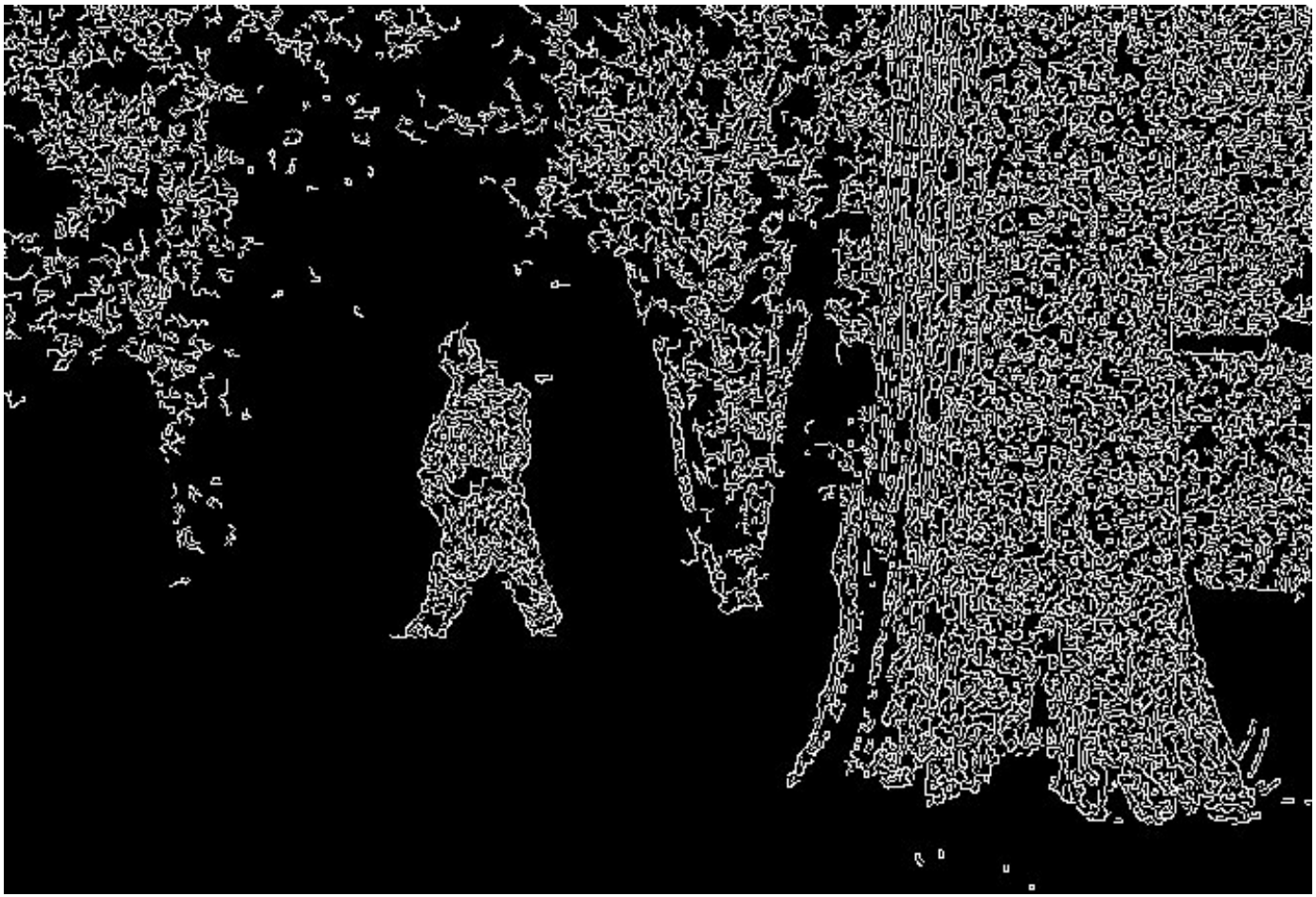
Неустойчивость к помехам



Изображения контуров увеличены.

Неустойчивость к помехам

Неустойчивость к помехам



Неустойчивость к помехам

Неустойчивость к помехам

Заключение о Chamfer matching

Во время работы над проектом:

- Были изучены статьи на тему «компьютерного зрения» (Chamfer matching, Canny algorithm), а также вероятностного поиска некоторой модели среди помех (RANSAC).
- Была написана реализация Chamfer matching на языке Python с использованием пакетов numpy и cv2.
- Для хранения исходного кода использовался GitHub.

Спасибо за внимание!

https://github.com/ReptoidBusters/object-detection