МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительной математики и информатики**

**КУРСОВАЯ РАБОТА**

**Анализ алгоритмов мультиобъектной трассировки для задач машинного зрения**

Работу выполнил Н.А. Дульцев

(подпись, дата)

Факультет математики и компьютерных наук 2 курс Направление 02.03.01 математика и компьютерные науки

Научный руководитель

доцент кафедры ВМИ,

кандидат технических наук Р.Ю. Вишняков

(подпись, дата)

Нормоконтролер

преподаватель А.А. Цыбенко

(подпись, дата)

Краснодар 2018

СОДЕРЖАНИЕ

[Введение 4](#_Toc515485783)

[1 Алгоритм Лукаса-Канаде 6](#_Toc515485784)

[1.1 Идея алгоритма Лукаса-Канаде 6](#_Toc515485785)

[1.2 Математическая модель алгоритма Лукаса-Канаде 6](#_Toc515485786)

[1.3 Подбор инструмента реализации алгоритма 7](#_Toc515485787)

[2 Алгоритм TLD Predator 8](#_Toc515485788)

[2.1 Идея алгоритма TLD Predator 8](#_Toc515485789)

[2.2 Математическая модель алгоритма TLD Predator 8](#_Toc515485790)

[2.3 Подбор инструмента для реализации алгоритма TLD Predator 9](#_Toc515485791)

[3 Алгоритм MIL 10](#_Toc515485792)

[3.1 Идея алгоритма MIL 10](#_Toc515485793)

[3.2 Математическая модель алгоритма MIL 10](#_Toc515485794)

[3.3 Подбор инструмента для реализации алгоритма MIL 11](#_Toc515485795)

[4 Алгоритм KCF 13](#_Toc515485796)

[4.1 Идея алгоритма KCF 13](#_Toc515485797)

[4.2 Математическая модель алгоритма KCF 13](#_Toc515485798)

[4.3 Подбор инструмента для реализации алгоритма KCF 13](#_Toc515485799)

[5 Алгоритм HOG 15](#_Toc515485800)

[5.1 Идея алгоритма HOG 15](#_Toc515485801)

[5.2 Модель алгоритма HOG 15](#_Toc515485802)

[5.3 Подбор инструмента для реализации алгоритма HOG 16](#_Toc515485803)

[6 Тестирование алгоритмов 17](#_Toc515485804)

[6.1 Описание тестов 17](#_Toc515485805)

[6.1.1 Тест под углом 17](#_Toc515485806)

[6.1.2 Статичный тест 18](#_Toc515485807)

[6.1.3 Динамический тест 18](#_Toc515485808)

[6.2 Таблица данных проведенных тестов 19](#_Toc515485809)

[6.3 Анализ проведенных тестов для алгоритмов 20](#_Toc515485809)

[6.3.1 Анализ алгоритма Лукаса-Канаде 20](#_Toc515485810)

[6.3.2 Анализ алгоритма TLD 20](#_Toc515485811)

[6.3.3 Анализ алгоритма KCF 21](#_Toc515485812)

[6.3.4 Анализ алгоритма MIL 21](#_Toc515485813)

[6.3.5 Анализ алгоритма HOG 21](#_Toc515485814)

[Заключение 23](#_Toc515485815)

[Список использованных источников 24](#_Toc515485816)

# ВВЕДЕНИЕ

Анализ – это исследование, а в случае алгоритма это еще и реализация в купе со множеством тестов. Его результат ложится в основу научных трудов.

Если взглянуть на современные мировые тенденции, то не трудно выделить несколько направлений, которые пользуются спросом на рынке, в военной промышленности и в научных кругах. Одной из таких тем является машинное зрение. Оно хоть и незаметно, но окружает нас по всюду – от обычной фотокамеры или приложения камеры в телефоне до военных охранных или беспилотных систем. Вне зависимости от применения их объединяет одно – определение требуемого объекта или объектов.

Для анализа алгоритмов мультиобъектной трассировки требуется поставлены следующие задачи:

1. Изучение теоретической идеи алгоритма.
2. Разбор математической модели алгоритма.
3. Подбор требуемого инструмента для реализации алгоритма.
4. Реализация алгоритма.
5. Тестирование алгоритма.
6. Сбор данных и создание базы по ключевым параметрам.

Требования к анализу:

1. Проанализировать более трех алгоритмов с целью выбора лучшего.
2. Структурировать полученные данные в виде таблицы.
3. Дать описание соотношению определенных задач и наиболее

подходящих для них алгоритмов.

При создании курсовой работы использовался язык программирования Python версии 3.6 и среда программирования IDLE (Integrated DeveLopment Environment). Для более удобной работы с камерой и другой периферией компьютера используется библиотека компьютерного зрения OpenCV (Open Source Computer Vision Library) версии 3.3.0. Техническим составляющим является ноутбук со следующими характеристиками:

Процессор Intel i3-5005.

Оперативное запоминающее устройство KINGSTON VALUERAM KVR16LS11/4 DDR3L 8ГБ, 1600МГ.

## **1 Алгоритм Лукаса-Канаде**

## 

## **1.1 Идея алгоритма Лукаса-Канаде**

Алгоритм Лукаса-Канаде является прародителем всех отслеживающих алгоритмов и методов. Разработан был в 1981 году Брюсом Лукасом и Такео Канаде. Его локальный метод вычисления оптического потока лежит в основе большинства современных алгоритмов.

Принцип работы алгоритма заключается в конвертации цветного кадра видеопотока в черно-белый. Затем на нем проводится фиксация точек. Их определение происходит по принципу поиска контрастных мест кадра (граница черных и белых пикселей). Для работы с различными градациями оттенков черного и серого применяется фильтр допущения яркости.

По движениям определенных точек выстраиваются вектора движения оптического потока. После выделение объекта рамкой на основе векторов смещения потока происходит изменение положения и формы самой рамки.

## 

## **1.2 Математическая модель алгоритма Лукаса-Канаде**

Рассмотрим случай с одной точкой. После того, как на кадре была выбрана подходящая по условию точка, определяются ее координаты . Определим вектор смещения , затем составляется градиент: . Также его можно записать в виде суммы частных производных: .

Полученное уравнение с двумя неизвестными значениями вектора смещения. Предполагается, что равны. Тогда вводится весовой коэффициент пикселя, задаваемый функцией и по нему выстраивается вектор смещения: . Приравняем к нулю:

Таким образом определена длина смещения одной точки (пикселя). При работе не с точкой, а с целым фрагментом, нужно брать сумму от векторов смещения для каждой точки (пикселя), что дает в итоге вектор смещения не для точки, а для фрагмента (объекта).

## 

## **1.3 Подбор инструмента реализации алгоритма**

Помимо библиотеки OpenCV, импортируемая в код как:

import cv2

Используется графическая библиотека draw str, импортируемая в код как:

import draw\_str

## **2 Алгоритм TLD Predator**

## 

## **2.1 Идея алгоритма TLD Predator**

Алгоритм TLD Predator (Tracks – Learns – Detects) был создан Зденек Калалом. Его отличает надежность сопровождения объектов, даже после выхода объекта из кадра и возвращении его обратно спустя даже продолжительное время, алгоритм все равно продолжает отслеживать объект.

Это достигается с помощью обучения детектора, который определяет объект в процессе распознавания. Как только объект был опознан и помещен в рамку при помощи алгоритма, в основе которого лежит алгоритм Лукаса-Канаде, траектория смещения начинает наблюдаться двумя процессами.

Рассматривается объект, его модель и область, покрывающая их. Первый процесс будем называть расширяющим так как выбирается траектория движения объекта и обновляется модель объекта. Если после этого обновления покрытие содержит хотя бы одну часть модели, то покрытие расширяется. Так как первый процесс всегда увеличивается, то вторым мы будем урезать. Поэтому назовем его урезающим процессом. Идея его состоит в обрезании некой поверхности из модели. Урезание совершается, если в модели оказывается слишком большое количество объектов, не относящихся к рассматриваемому объекту.

При динамическом использовании двух процессов мы создаем систему, которая стабильно обучается под различные ситуации на видеоряде.

## 

## **2.2 Математическая модель алгоритма TLD Predator**

Пусть – прямоугольник, описывающий сопровождающий объект. Рассматриваем в момент времени . Пиксели внутри описываются вектором смещения . Последовательно описанные в каждый момент времени пиксели из , есть множество, которое называется треком и обозначается: . Полученный трек определяет траекторию объекта в кадре.

- объединяет все состояния объекта. Состояние определяется путем объединения всех признаков сравнения. Массив признаков сравнения называется . Тогда – треком признаков. Данное множество описывает траекторию в пространстве признаков .

Дальше объект сопровождается при помощи трека , а траектория в пространстве признаков анализируется двумя событиями: для расширяющего процесса и для урезающего процесса. пытается расширить , а – урезать. Оба события работают параллельно, стремясь к .

## 

## **2.3 Подбор инструмента для реализации алгоритма TLD Predator**

Помимо библиотеки OpenCV, импортируемая в код как:

import cv2

Используется Numpy – библиотека для сложных научных вычислений. Импортируется она как:

import numpy as np

Модуль Sys, который предоставляет данные системы. Импортируется как:

import sys

Модуль для работы со временем Time

import time

## **3 Алгоритм MIL**

## 

## **3.1 Идея алгоритма MIL**

Алгоритм MIL - Multiple Instance Learning (Многократное обучение) был создан для решения проблем изменения освещенности или вариаций позы. Сначала алгоритм делит объект на 9 блоков. Каждый блок оценивается и ему присваивается либо положительное значение (1), либо отрицательное (0).

Положительный блок получается путем обрезания фона у соседних ему блоков. Отрицательный же получается путем обрезания экземпляров из области кольцевого пространства вокруг блока с изменяемым радиусом. Предполагается, что особенности всех блоков удовлетворяют Гауссовскому распределению и тогда можно найти слабый классификатор с параметрами (1100), путем исследования всех блоков в положительном и отрицательном значениях.

Затем выбирается сильный классификатор путем выбора мощных из слабых. Дальше алгоритм вычисляет объект между слабым и сильным классификатором. После чего имеющиеся на каждом блоке сильные классификаторы используются для отслеживания объектов в последовательных кадрах. Классификаторы в последующем кадре выбираются из соседнего местоположения объекта в предыдущем кадре. Далее процесс повторяется заново.

## 

## **3.2 Математическая модель алгоритма MIL**

Объект делится на 9 блоков , затем для блока элементы из блока и фона обрезаются для создания положительных и отрицательных блоков.

Положительный блок «1» получается с помощью окружающих его обрезанных элементов, удовлетворяющих равенству , где – позиция, а - радиус окружности блока .

Отрицательный блок «0» получается с помощью обрезанных элементов из области кольцевого пространства вокруг блока, удовлетворяющих равенству , где – радиус внутренней окружности, а – радиус внешней окружности блока .

При разделении элементов блока на положительные и отрицательные, мы изучаем слабые классификаторы и получаем массив классификаторов: , где – это -й слабый классификатор для -го блока.

Образцы для положительных и отрицательных блоков представлены в виде признаков Хаара, которые удовлетворяют Гауссовскому распределению. После чего -й слабый классификатор получается путем обучения -й особенности всех экземпляров. Процесс обучения происходит одновременно и в положительных и в отрицательных блоках.

Сильный классификатор генерируется путем выбора самых сильных из слабых классификаторов K из массива : .

После применения вышеописанного алгоритма на каждом блоке, сильные классификаторы используются для отслеживания объектов.

## 

## **3.3 Подбор инструмента для реализации алгоритма MIL**

Помимо библиотеки OpenCV, импортируемая в код как:

import cv2

Используется Numpy – библиотека для сложных научных вычислений. Импортируется она как:

import numpy as np

Модуль Sys, который предоставляет данные системы. Импортируется как:

import sys

Модуль для работы со временем Time

import time

## **4 Алгоритм KCF**

## 

## **4.1 Идея алгоритма KCF**

Алгоритм KCF – Kernelized Correlation Filters (Кернелизированные корреляционные фильтры) является некой библиотекой различных методов трекинга. Все эти методы работают на матрицах и, так же, как и большинство алгоритмов, KCF видит только один кадр за раз, что влияет на размер каждой обрабатываемой матрицы.

## 

## **4.2 Математическая модель алгоритма KCF**

Рассмотрим функцию регрессии , где – вектор коэффициентов, – совокупность изображений, – логическая переменная, которая принимает значение «1», если объект присутствует в изображении и значение «-1», если объект отсутствует. Эта выпуклая функция минимизирует риск потери объекта.

Она обучается вместе со всеми сдвинутыми и искаженными версиями объекта. Для этого берется циркулянтная матрица , которая является дискретным преобразованием Фурье для матрицы . – диагональная матрица с коэффициентами вектора .

Для получения более лучшей дискриминации, используется кернелизированная функция , с помощью которой определяется линейная интерполяция, которая описывает движение объекта.

## 

## **4.3 Подбор инструмента для реализации алгоритма KCF**

Помимо библиотеки OpenCV, импортируемая в код как:

import cv2

Используется Numpy – библиотека для сложных научных вычислений. Импортируется она как:

import numpy as np

Модуль Sys, который предоставляет данные системы. Импортируется как:

import sys

Модуль для работы со временем Time

import time

## **5 Алгоритм HOG**

## 

## **5.1 Идея алгоритма HOG**

Алгоритм HOG - Histogram of Oriented Gradients (Гистограммы направленных градиентов) впервые описали Навнит Далал и Билл Триггс в 2005 году. Изначально он был разработан для статичных изображений, но уже через год его смогли использовать для определения объектов на видеоряде.

Идея алгоритма заключается в следующем. Изображение разбивается на области и в них по особым точкам подсчитывается количество направлений градиента. На их основе строится сетка ячеек, которые оказываются равномерно распределены по всему изображению. По ним и определяется объект и его изменения. Так как освещение объекта не постоянно, то используется дополнительная нормализация, перекрывающего локального контраста, чем и достигается увеличение точности.

## 

## **5.2 Модель алгоритма HOG**

Допускается, что внешний вид, форма, и положение объекта на локальном участке изображения могут быть описаны при помощи распределения градиентов интенсивности или направлением краев объекта.

Для этого изображение должно быть поделено на локальные зоны – ячейки, которые связаны между собой. Для каждого пикселя внутри каждой ячейки производится расчет гистограммы градиентов и направлений.

Комбинация всех гистограмм называется дескриптором. Затем для увеличения точности, локальные гистограммы нормализуются по контрасту при помощи вычисления меры интенсивности на большом фрагменте изображения, который называется блоком. Блок может объединять несколько локальных зон.

## **5.3 Подбор инструмента для реализации алгоритма HOG**

Помимо библиотеки OpenCV, импортируемая в код как:

import cv2

Используется Numpy – библиотека для сложных научных вычислений. Импортируется она как:

import numpy as np

Модуль для работы со временем Time

import time

## **6 Тестирование алгоритмов**

Каждый алгоритм поочередно запускается на видеофайле. Работа каждого отслеживается по следующим параметрам:

1. Производительность – оценивается нагрузкой на систему и частотой

кадров выходящего видеоряда ().

1. Стабильность ведения объекта – оценивается количеством

прерываний или искажением рамки объекта ().

1. Количество потерь - оценивается количеством полностью

утерянных объектов из 10 выбранных. (), если потерь нет, то оценка = 1.5.

1. Среднее количество кадров, нужное для потери объекта

().

1. Количество неверных переопределений из 10 выбранных ().
2. Средний показатель по всем параметрам.

## 

## **6.1 Описание тестов**

## 

## **6.1.1 Тест под углом**

Для теста используется видеозапись формата .MP4 с разрешением пикселей. Количество кадров равно 1500. Частота кадров равна 15 в секунду. На видеоряде представлена пешеходная зона, обзор на которую ведется под углом, что дает искажение. Среднее число людей, находящихся в кадре – больше 50. Присутствуют велосипедисты, большие скопления людей, сидящие люди, деревья, фонарные столбы, засветы и вывески. Люди перекрывают друг друга, сталкиваются, а также идут вместе. Расстояние до объектов от 10 до 50 метров.

## 

## **6.1.2 Статичный тест**

Для данного теста используется видеозапись формата .MP4 с разрешением пикселей. Количество кадров равно 165. Частота кадров равна 15 в секунду. На видеоряде представлен тротуар, обзор на который ведется перпендикулярно. Люди размеренно проходят, изредка перекрывая друг друга. Расстояние до объектов от 2 до 5 метров. Максимальное количество людей в кадре – 10.

## 

## **6.1.3 Динамический тест**

Для данного теста используется видеозапись формата .MP4 с разрешением пикселей. Количество кадров равно 750. Частота кадров равна 15 в секунду. На видеоряде обзор представлен из кабины автобуса на проезжую часть и тротуары. Присутствует тряска, люди идут, стоят, перекрывают друг друга, перебегают дорогу. Есть также теневые и засвеченные области. Расстояние до объектов от 5 до 50 метров. Максимальное количество людей в кадре – около 20.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6 | 0.15814 | 1.1788 | 1.0114 | 1.4996 | 2.32 |
| 5 | 0.0006+0.0007+0.0004 | 0.14+0.15+0.16 | 0.007+0.2+0.001 | 0.1+0.1+0.1 | 1+1+0.2 |
| 4 | 0.003+0.004+0.002 | 0.86+0.18+0.013 | 0.086+0.15+0.033 | 0.75+0.24+0.04 | 0.9+1+0.4 |
| 3 | 0.1+0.1+0.1 | 1.5+0.2+1 | 0.1+0.7+0.1 | 1.5+1+1 | 1.5+1.5+1 |
| 2 | 0.01+0.01+0.01 | 1.1+0.3+0.27 | 1.6+0.43+0.5 | 1.4+0.6+0.6 | 1+1+1 |
| 1 | 0.15+0.15+0.15 | 0.003+0.01+0.008 | 0.33+0.36+0.46 | 0.026+0.032+0.01 | 0.03+0.05+0.02 |
|  | Лукаса-Канаде | TLD | KCF | MIL | HOG |

Таблица 1. Данные проведенных тестов.

## **6.2 Таблица проведенных тестов**

## **6.3 Анализ проведенных тестов для алгоритмов**

## 

## **Анализ алгоритма Лукаса-Канаде**

Безусловно преимуществом этого алгоритма является его производительность. Правда, на этом его достоинства кончаются. Каждый новый кадр искривляет все сильнее рамку объекта и после 50 кадров рамка вовсе исчезает. После потери объекта алгоритм нужно перезапускать, так как переопределение не происходит.

В итоге алгоритм Лукаса-Канаде хорош для использования в среде с малым количеством оттенков цветов и теней и при высоком разрешении видеоряда. Алгоритм нахождения вектора смещения точек может лечь в основу более серьезного трекера.

## 

## **6.3.2 Анализ алгоритма TLD**

Высокая точность и стабильность – главные преимущества алгоритма TLD. Искажения и переопределения происходят в случае попадания объекта в схожий по цветовой гамме фрагмент кадра. После выхода из полученной зоны, объект переопределяется без искажения или сдвигов.

Не минус, но и не плюс – это попытки алгоритма найти схожие объекты после того, как изначальный вышел за кадр. Обычно это заканчивается переопределением на объект со схожими характеристиками.

Минусы – низкая производительность и требовательность к качеству. Количество вычислений на столько велико, что частота кадров становится равной от до кадра в секунду.

Итог – алгоритм TLD один из самых надежных и стабильных алгоритмов, если использовать его с видеорядом высокого качества. Если повысить производительность, то его использование возможно и в офисных помещениях для отслеживания работников и на улице для отслеживания транспортных средств.

## 

## **6.3.3 Анализ алгоритма KCF**

Преимущества: высокая производительность, отсутствие какого-либо искажения, неприхотливость к разрешению видеоряда.

Минусы: при перекрытии объекта отслеживания любым другим объектом, хоть на 1 кадр – алгоритм теряет объект и больше не восстанавливает.

Идеально подойдет при работе с кратковременным отслеживанием объектов, или же для разработки более сложного алгоритма в качестве дополнительного трекера.

## 

## **6.3.4 Анализ алгоритма MIL**

Плюсы: приемлемая частота кадров, после перекрытия объекта другим, исходный не теряется. Происходит изменение модели объекта и только при определенных условиях перекрывающий объект может перевести рамку на себя.

Минусы: возможное переопределение одного объекта другим и после выхода объекта за пределы кадра, рамка перепривязывается к фону и не исчезает.

Алгоритм MIL подходит для отслеживания любых объектов в различных условиях. Также существует возможность его улучшения, для исключения перечисленных выше проблем.

## **6.3.5 Анализ алгоритма HOG**

В отличие от разобранных выше алгоритмов, алгоритм HOG отличается быстрым переопределением, малым искажением и сопровождением объекта без потерь. Прихотлив к качеству видеоряда. Чем ниже качество – тем выше требуется производительность.

Минусы: определение за исходный объект всех схожих по характеристикам объектов, при высокой динамике видеоряда повышается количество ложных определений объектов.

В итоге алгоритм HOG подходит для работы в местах скопления большого количества объектов. Высокая динамика не влияет на качество трекера, а только сильнее нагружает систему. Возможно использование алгоритма в более сложном.

## **ЗАКЛЮЧЕНИЕ**

В процессе выполнения курсовой работы были выполнены следующие задачи:

1. Изучены теоретические идеи пяти алгоритмов.
2. Разобраны математические модели пяти алгоритмов.
3. Совершен подбор требуемых инструментов для реализации пяти

алгоритмов.

1. Реализованы пять алгоритмов.
2. Протестированы пять алгоритмов.
3. Совершен сбор данных по пяти алгоритмам.
4. Проанализированы пять алгоритмов и выбран лучший.
5. Все данные по пяти алгоритмам записаны в таблице.
6. Описаны приоритетные области работы каждого из пяти

алгоритмов.

Результатом выполнение курсового проекта стал анализ алгоритмов мультиобъектной трассировки в задачах машинного зрения, который позволяет:

1. Узнать принцип работы алгоритмов.
2. Изучить плюсы и минусы алгоритмов.
3. Воспользоваться результатами тестирования для выбора

подходящего под поставленную задачу алгоритма.

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Далал Н., Триггс В. Гистограммы ориентированных градиентов для

обнаружения человека, 2005 г. 1 – 8 с.

1. Джорджа М., Рослинг Б.Х., Джимсон М. Оценка эффективности

отслеживания на основе KCF с использованием набора данных VOT, 2017 г. 2 – 4 с.

1. Калал З. Ошибка вперед-назад: автоматическое обнаружение сбоев

слежения, 2010 г. 23 – 26 с.

1. Лоу Д.Г. Отличительные особенности изображения от масштабно-

инвариантных ключевых точек, 2004 г. 91 – 110 с.

1. Лукас Б.Д., Канаде Т. Итерационная техника регистрации изображений

с применением стереовидения, 1981г. – 130 с.

1. Мессом С.Н., Барчак А.Л. Быстрые и эффективные вращающиеся хаар-

функции с использованием повернутых интегральных образов, 2006г. 1 – 6 с.

1. Чжэньцзе Ван, Лицзя Ванг, Хуа Чжан Алгоритм множественного

обучения признаков для отслеживания объектов, 2017г. 1 – 6 с.

1. Интернет-ресурс:<https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/> (Дата посещения: 12.10.2017)