ИССЛЕДОВАНИЕ МЕТОДОВ ОТДЕЛЕНИЯ ОБЪЕКТОВ ОТ ФОНА НА ФОТОГРАФИЯХ

Солдатов А.В.[1], Козырева Д.Д.[1], Мазиков Я.А.[1], Лукашов С.А.[2]
Национальный исследовательский университет «Высшая школа экономики»,
[1]департамент прикладной математики
[2]департамент электронной инженерии
МИЭМ НИУ ВШЭ

Аннотация

В работе рассматриваются несколько методов отделения объектов от фона на изображениях. Данное исследование проводится в рамках разработки системы для автоматической разметки датасета деталей для последующего применения в системе по контролю сборки механизмов в ручном производстве. В данной работе будут рассматриваться методы как классического машинного зрения, так и методы глубинного обучения для отделения объектов (деталей) от фона, используемые в системе для автоматической разметки датасета.

Введение

В современном производстве все большую и большую роль играют компьютерные технологии, которые облегчают, ускоряют и делают более точным производственный процесс. Частным случаем такого применения является контроль операций в ручном производстве с помощью методов машинного обучения. Применение методов машинного обучения влечет за собой необходимость быстрой автоматической разметки датасета для новых деталей новых механизмов.

В качестве валидационной выборки для сравнения результатов работы алгоритмов будем использовать набор из 20 фотографий 10 предметов, лежащих на столе однотонного цвета. На каждой фотографии присутствует единственная деталь, которую необходимо отделить от фона.

Методы классического компьютерного зрения

В данном разделе речь пойдет о давно известных ручных методах отделения фона от объекта. Прежде всего, это детектирование краев и обнаружение пикселей, значение цвета в которых выше некоторого порога, характерного для изображения.

Рассмотрим основные методы обнаружения объекта в классическом компьютерном зрении:

1) Сегментация по цвету (segmentation based on color thresholding);
2) Детектирование контуров и алгоритм Canny;
3) Взятие разности кадров;
4) Алгоритм GrabCut;

Алгоритмы сегментации обычно основаны на одном из двух принципов - непоследовательности или сходстве. Использование пороговых значений насыщенности цвета – это пример алгоритма, основанного на сходстве: он делит изображение на зоны, схожие по значению цвета. В итоге на выходе получается изображение, пиксели которого принадлежат одному из двух классов: “объект” или “фон”. [1]

Рассмотрим данный метод подробнее. Пусть t – определенный для изображения порог. Тогда $g(x)$ – значение в пикселе результирующего изображения будет задаваться следующей формулой (1):

$$g(x,y) = \begin{cases} 0, & \text{если } f(x,y) < t \\ 255, & \text{если } f(x,y) \geq t \end{cases} \qquad (1)$$

где $f(x,y)$ – значение в пикселе исходного изображения.

Функцией, реализующей данный алгоритм, в библиотеке алгоритмов компьютерного зрения opencv [2] является cv.threshold. Она принимает черно-белое изображение и пороговое значение. OpenCV предоставляет различные типы пороговых значений. Базовое пороговое значение выполняется с использованием cv.THRESH_BINARY. Ниже перечислены все простые пороговые типы:

- cv.THRESH_BINARY
- cv.THRESH_BINARY_INV
- cv.THRESH_TRUNC
- cv.THRESH_TOZERO
- cv.THRESH_TOZERO_INV

Маски, полученные при использовании данных порогов, приведены на рис. 2.

Заметим, что в данных методах используется заранее выбранное пороговое значение. Однако существуют методы, которые определяют порог в зависимости от изображения, например, метод Оцу [3] – он определяет оптимальное глобальное пороговое значение по гистограмме изображения (рис. 1).

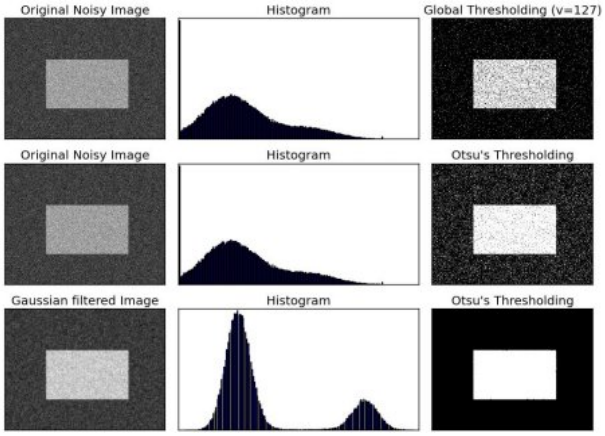

Рис.1. Пример работы метода Оцу.

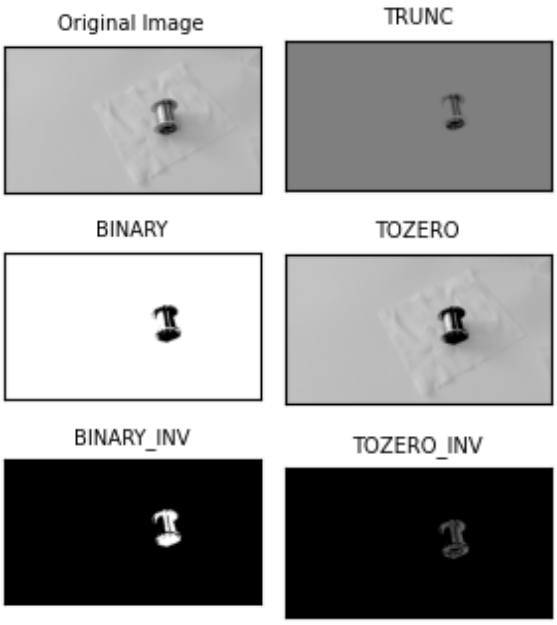

Рис.2. Примеры сегментации по цвету с использованием различных библиотечных порогов.

Алгоритм Оцу пытается найти пороговое значение t, которое минимизирует взвешенную внутриклассовую дисперсию, заданную соотношением:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t), \text{где}$$

$$q_1(t) = \sum_{i=1}^{t} P(i), \qquad q_2(t) = \sum_{i=t+1}^{I} P(i),$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)}, \qquad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)},$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} \frac{\big(i-\mu_1(t)\big)^2 P(i)}{q_1(t)},$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{I} \frac{\big(i-\mu_2(t)\big)^2 P(i)}{q_2(t)}$$

Таким образом, находим значение t, лежащее между двумя пиками так, что отклонения для обоих классов минимальны.

Детектирование контуров и алгоритм Canny [4]

Другим подходом к отделению объекта от фона является детектирование контуров на изображении. Рассмотрим многоуровневый алгоритм обнаружения краев, разработанный в 1986 году Джоном Ф. Кэнни. Алгоритм включает в себя:

1. Размытие изображения для удаления шума.
2. Поиск градиентов. Границы отмечаются там, где градиент изображения приобретает максимальное значение.
3. Подавление не-максимумов (NMS).
4. Двойная пороговая фильтрация и получение итоговых границ.

На первом шаге избавляемся от шума на изображении с помощью фильтра Гаусса. Для этого применяется метод свертки изображения с ядром Гаусса (3×3, 5×5, 7×7 и т. д.). Размер ядра зависит от ожидаемого эффекта размытия. Уравнение для ядра фильтра Гаусса размером $(2k + 1) \times (2k + 1)$ имеет вид:

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\big(i-(k+1)\big)^2 + \big(j-(k+1)\big)^2}{2\sigma^2}\right)$$
$$1 \le i, j \le (2k+1)$$

Далее находим градиент интенсивности изображения. Для этого в библиотечной функции cv.Canny используется пара сверток (G_x, G_y) и затем находятся градиент силы G и направление Θ. Направление округляется до одного из четырех возможных углов (0, 45, 90 или 135).

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

$$G = \sqrt{G_x^2 + G_y^2} \qquad \Theta = \arctan\left(\frac{G_y}{G_x}\right)$$

В идеале итоговые границы должны быть тонкими линиями. Для сглаживания полос и нужен третий шаг: на нем удаляются пиксели, которые не считаются частью границы. Идея подавления в следующем: все одинаково ориентированные соседствующие пиксели из ряда “сворачиваются” в один с максимальным градиентом G. Так, с рисунка, приведенного ниже, останутся только пиксели, обведенные белым контуром, после процедуры подавления не-максимумов (см. рис. 3).

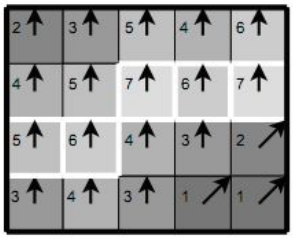

Рис. 3. Пример подавления не-максимумов.

Далее алгоритм Canny использует два порога (верхний и нижний):

– Если градиент пикселя выше верхнего порога, пиксель принимается как край.

– Если значение градиента пикселя ниже нижнего порога, он отклоняется.

– Если градиент пикселя находится между двумя порогами, он будет принят, только если он связан с пикселем выше верхнего порога.

Чем меньше порог, тем больше границ будет находиться, но вместе с тем более восприимчивым к шуму станет результат, и будут выделяться лишние данные изображения. Высокий же порог может проигнорировать слабые края или получить границу фрагментами.

Canny рекомендовал соотношение для верхнего/нижнего порога от 2:1 до 3:1.

Применение метода Canny вдобавок к маске, полученной алгоритмом threshold, позволило получить следующий результат (рис. 4):

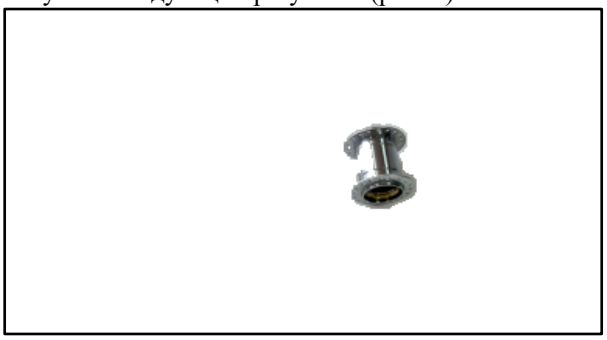

Рис. 4. Использование threshold + canny алгоритмов в качестве маски на примере детали из датасета.

Алгоритм GrabCut [5]

Данный метод работает следующим образом: пользователь помещает интересующий объект в прямоугольник. Края прямоугольника необязательно должны точно подходить к границам выделяемого объекта. Далее алгоритм сегментации отделяет сам объект от фона в заданном прямоугольнике.

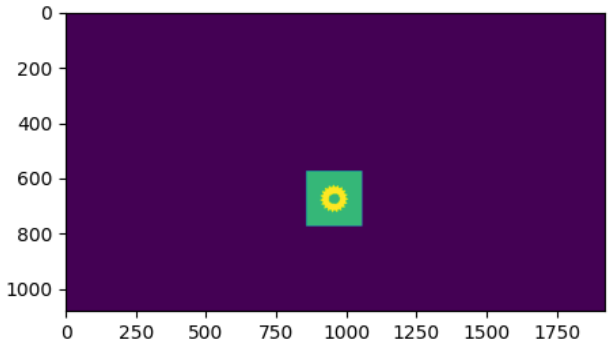

Рис.5. Пример работы алгоритма GrabCut.

За основу взят алгоритм интерактивной сегментации GraphCuts. Изображение рассматривается как вектор $z = (z_1, \dots, z_N)$, где z_n – интенсивность n-ого пикселя. Всего пикселей N штук. Далее, алгоритм определяет массив прозрачности пикселей $a = (a_1, \dots, a_N)$, где a_n – интенсивность n-ого пикселя, которая может принимать только 2 значения: 0, если пиксель принадлежит фону и 1, если пиксель принадлежит объекту. Строится гистограмма распределения интенсивности заднего и переднего плана: $\Theta = \{h(z;\, a), a = 0, 1\}$. Задача найти неизвестные a_n, то есть решить задачу сегментации. Вводится функция энергии:

$$U(a, \Theta, z) = -\sum_n \ln\left(h(z_n, a_n)\right)$$

$$V(a, z) = \sum_{(m,n)\in C} \frac{1}{dis(m,n)}[a_n \neq a_m] exp(-\beta(z_m - z_n)^2)$$

$U(a, \Theta, z)$ - слагаемое, отвечающее за качество отделения объекта от фона, $V(a, z)$ – слагаемое, отвечающее за участие пар пикселей в сумме. $dis(n, m)$ – евклидовое расстояние. Далее решается задача поиска глобального минимума функции E для получения массива прозрачности $\alpha = argmin_a E(a, \Theta)$. Для поиска минимума изображение описывается как граф и производится поиск его минимального разреза. Основное отличие алгоритма GrabCut от GraphCuts в том, что в GrabCut для описания цветовой статистики используется модель смеси гауссиан (Gaussian Mixture Model – GMM). Пиксели в данном описании рассматриваются в RGB формате. На рис. 6 показана примерная схема работы алгоритма GrabCut, на рисунках 7, 8 показан результат применения данного алгоритма.

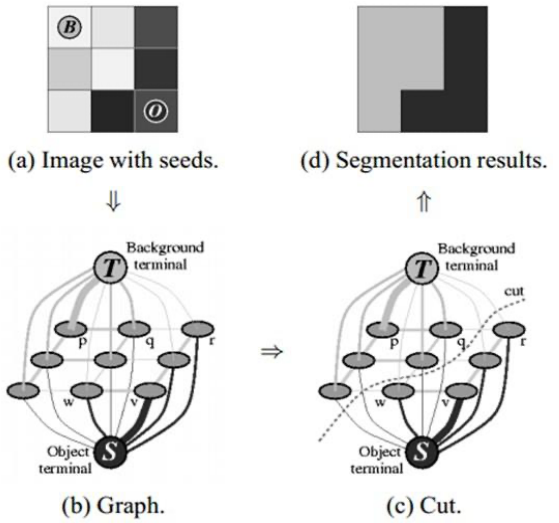

Рис.6. Примерная визуализация GrabCut алгоритма

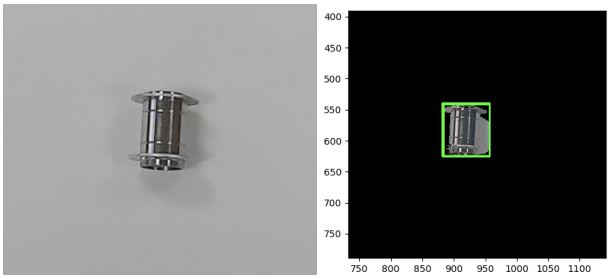

Рис.7 Реальное изображение и результат работы алгоритма GrabCut

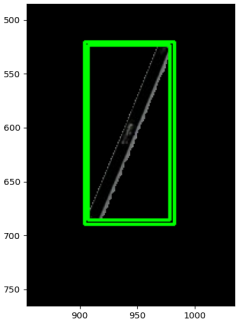

Рис.8 Точная разметка и результат GrabCut на одном изображении

Метод взятия разности кадров [6]

Данный метод основан на вычислении абсолютного различия между текущим и предыдущим кадром. Полученный при вычитании массив показывает изменения пикселей на текущем кадре по сравнению с эталонным (предыдущим) кадром.

$$dst(I) = abs(src1(I) - src2(I))$$

$scr1$ - первый исходный массив данных (эталонное изображение)

$scr2$ - второй исходный массив данных (текущее изображение)

dst - выходной массив (разность кадров)

Вычитаемые массивы должны быть одинакового размера.

Метод глубинного обучения

В методах глубинного обучения в задачах сегментации изображений преимущественно используются сверточные нейронные сети, которые очень хорошо подходят для работы с многомерными матрицами пикселей. Однако в применении таких сетей существуют ограничения.

Одним из самых сильных недостатков, с которым сталкиваются программисты при обучении сверточных сетей, является необходимость создания огромной обучающей выборки, поскольку для получения наиболее точных предсказаний требуется очень тонкая настройка всех параметров сети и с маленьким начальным набором данных такого результата достаточно проблематично достичь. В связи с этим при решении задач сегментации, детекции или распознавания, как правило, используются инструменты аугментации.

Аугментация – это построение дополнительных данных из исходных путем применения масштабирования, поворота, обрезки и тому подобного.

Например, в работе [7] по сегментации изображений мембран нейронов в электронной микроскопии выборка создается искусственно: для каждого пикселя выбирается малая окрестность, включающая его, что позволяет значительно увеличить датасет. Однако у такого подхода есть два недостатка: во-первых, такая нейросеть довольно медленная из-за необходимости постоянной обработки обособленных регионов для каждого пикселя отдельно и, во-вторых, количество верных ответов нейросети обратно пропорционально зависит от размера выбранных регионов. Чем меньше регион, тем меньше контекстной информации он содержит, следовательно сеть будет не способна в дальнейшем качественно распознавать изображения с большим содержанием информации и наоборот.

Существуют более элегантные решения задачи сегментации. Одним из вариантов является U-Net, которую мы использовали в своей работе.

Архитектура U-Net

Нейросеть состоит из сжимающей части (левая ветвь) и увеличивающей части (правая ветвь) (рис. 9). Для левой ветви используются стандартные для сверточной сети алгоритмы: свертка и уменьшение масштаба по максимальным элементам (max pooling). При этом на каждом шаге вниз число каналов (карт свойств) для изображения увеличивается вдвое. Для правой ветви используются алгоритмы увеличения масштаба (upsampling) и свертки. Такие действия уменьшают число каналов на каждом шаге вверх вдвое. В итоге используется свертка для сокращения числа карт свойств и получения желаемого числа классов (масок для них).

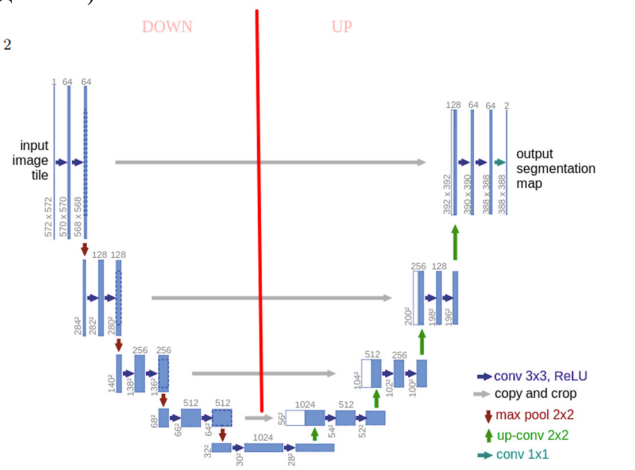

Рис. 9. Архитектура U-Net (пример изображения с разрешением 32×32 пикселя — самым низким). Каждый синий квадрат соответствует многоканальной карте свойств. Количество каналов приведено в верхней части квадрата. Размер x-y приведен в нижнем левом краю квадрата. Белые квадраты представляют собой копии карты свойств. Стрелки обозначают различные операции.

U-net довольно быстро и качественно способна выполнять задачу сегментации изображений для различных наборов входных данных. На испытании сегментации изображений клеток (ISBI) в 2015 году данная нейросеть победила, достигнув значения IOU (пересечение над объединением) 0.9203 на наборе

PhC-U373 и значения 0.7756 на наборе DIC-HeLa (данные взяты из [8]).

Процесс реализации классических методов

Для тестирования приведенных выше алгоритмов предварительно была вручную выделена рабочая область стола, чтобы избежать неинформативных засветов по краям. Далее к одному и тому же датасету были применены в разных сочетаниях различные алгоритмы. Результаты их работы показаны ниже на рис. 11.

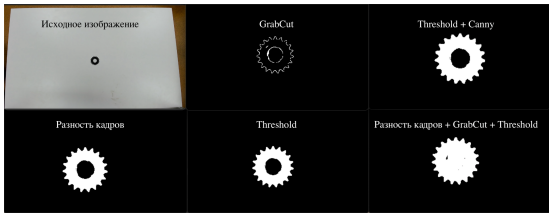

Рис.11 Отделение фона классическими алгоритмами.

Эксперименты по отделению фона проводились в около идеальных условиях при хорошей освещенности, детали размещались на однотонном белом столе. Также на изучаемых фотографиях отсутствовали посторонние предметы и артефакты. Была сделана эталонная фотография пустого стола, после чего было сфотографировано некоторое количество деталей на этом же столе в тех же условиях. В качестве метрики качества используется мера Жаккара (Intersection over Union – IoU). По ней сравниваются бинарная маска вручную размеченной фотографии и маска этой же фотографии, полученная с помощью описанных выше алгоритмов.

Эксперименты для глубокого обучения

Обучив U-Net на расширенном датасете деталей для велосипедной втулки (449 изображений), нашей команде удалось достичь среднего по выборке коэффициента IoU 0.64.

Пример выходного изображения для модели приведен ниже на рис. 10:

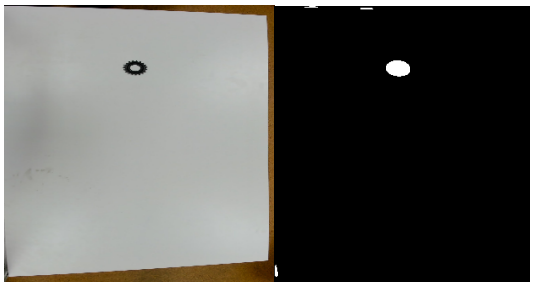

Рис. 10. Реальное изображение и предполагаемая маска U-Net

В дополнение к теме сегментации изображений с помощью нейросетей, была проверена не обученная U-Net, в которой использовались общедоступные веса ImageNet. На датасете деталей было получено значение IoU = 0.0004, следовательно, для использования U-Net необходимо обучение для получения приемлемых результатов. Ниже приведена таблица 1 со значениями метрики для различных алгоритмов и их комбинаций.

Метод	IoU	Преимущества	Недостатки
Разность кадров	0.65	Быстрота работы и простота метода	Сильно зависим от освещенности, теней и посторонних предметов
GrabCut (область детектирования – весь стол)	0.08	Эффективен в паре с другими методами	Медленная работа, низкая точность из-за шума по краям
Threshold	0.52	Быстрая работа алгоритма	Чувствителен к освещению
Color Threshold + Canny	0.54	Простая реализация: эталонный фон и разметка датасета не требуются	Зависимость от фона и освещения
GrabCut + разность кадров + threshold	0.81	Высокая точность	Необходимость эталонного фото
U-Net (обученная)	0.64	Высокая производительность, нужен небольшой датасет благодаря аугментации	Необходимость ручной разметки датасета
U-Net (необученная, с весами ImageNet)	0.0004	Нет необходимости обучать модель	Не подходит для практического использования

Таблица 1. Сравнение алгоритмов отделения фона на изображении

Выводы

С помощью алгоритма threshold удалось добиться качества IoU – 0.52. Данный метод за счет встроенных в библиотеку opencv прост в реализации и не требует дополнительной подготовки в виде фона стола или обучения на датасете.

Совокупность threshold с методом выделения краев Canny не дала сильного прироста в качестве (значение IoU получилось 0.54). Однако, при ближайшем рассмотрении можно заметить, что алгоритм Canny дает немного более четкие границы выделяемого объекта. При плохом освещении, попадании в кадр других объектов или объектах на цветном фоне метод выбора порогового значения цвета перестаёт работать и детектирует лишнее.

Алгоритм взятия разности кадров показал неплохое значение метрики IoU – 0.65. Однако данный алгоритм очень чувствителен к внешним факторам, таким как тени, уровень освещенности, наличие посторонних предметов. Поэтому использование его, как самостоятельного алгоритма для отделения фона может привести к серьезным неточностям.

Был проведен эксперимент только с одним алгоритмом GrabCut, когда рассматриваемый прямоугольник – вся рабочая область. Хотя на маске можно различить довольно четкие границы детали, метрика IoU получилась близкой к 0, потому что помимо самой детали, на маске оказалось много постороннего шума, который повлиял на значение метрики. (на фото шум не попал в кадр, так как объект сильно приближен и находится в центре, а шум расположен по краям). Из-за большой площади начальной области детектирования алгоритм работает долго, затрачивая примерно по 3-4 минуты на одно фото в Full HD.

Используя U-Net (предварительно обучив), можно добиться довольно хороших результатов в сегментации изображений деталей (IoU от 0.64 и выше). При этом преимущество данной архитектуры в её скорости работы. Однако необходимость обучения и ручной разметки датасета увеличивает время, которое нужно выделить для работы с этим методом (без обучения результаты не получится использовать для решения практических задач – IoU около 0.0004).

В качестве самого интересного подхода можно выделить алгоритм GrabCut в тандеме с методом взятия разности кадров двух изображений и threshold с фиксированным порогом. С помощью операции разности кадров был сгенерирован примерный прямоугольник, в котором находится объект. Данный прямоугольник был подан на вход алгоритму GrabCut, который уже окончательно определил область нахождения объекта на изображении. По фиксированному порогу (подбирался вручную) изображение было переведено в маску из 0 и 1. Данный подход имеет преимущество в том, что нет необходимости вручную задавать начальный прямоугольник, в котором будет осуществляться поиск объекта. На валидационной выборке удалось достичь коэффициента качества IoU в среднем 0.84.

Список литературы

[1] Kulkarni N. Color thresholding method for image segmentation of natural images //International Journal of Image, Graphics and Signal Processing. – 2012. – Т. 4. – №. 1. – С. 28.

[2] OpenCV-Python Tutorials // Open Source Computer Vision URL: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

[3] OpenCV-Python Tutorials // Open Source Computer Vision URL: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

[4] Canny J. A computational approach to edge detection //IEEE Transactions on pattern analysis and machine intelligence. – 1986. – №. 6. – С. 679-698.

[5] Rother C., Kolmogorov V., Blake A. " GrabCut" interactive foreground extraction using iterated graph cuts //ACM transactions on graphics (TOG). – 2004. – Т. 23. – №. 3. – С. 309-314.

[6] OpenCV: Operations on arrays // Open Source Computer Vision URL: https://docs.opencv.org/3.4.7/d2/de8/group__core__array.html#ga6fef31bc8c4071cbc114a758a2b79c14

[7] Ciresan D. et al. Deep neural networks segment neuronal membranes in electron microscopy images //Advances in neural information processing systems. – 2012. – Т. 25.

[8] Ronneberger O., Fischer P., Brox T. U-net: Convolutional networks for biomedical image segmentation //International Conference on Medical image computing and computer-assisted intervention. – Springer, Cham, 2015. – С. 234-241.