



Физически факултет

Специалност: "Компютърно инженерство",

Дипломна работа
за
придобиване на образователно-квалификационна
степен "бакалавър"
на
Петко Петков, факултетен №15037

Тема: Определяне параметрите на
калиярни мостове посредством
машинно зрение

Научен ръководител:

/доц. д-р Христо Илиев/

София, юли 2024

Съдържание

1 Въведение	3
2 Експериментална постановка	3
3 Търсени параметри	4
4 Модели за сегментация	6
4.1 Прагова обработка с метод на Оцу	6
4.2 Невронна мрежа "Segment Anything"	10
5 Измерване на параметрите	20
5.1 Отделяне на контури	20
5.2 Измерване на шийка (neck)	21
5.3 Измерване на горна, долната, лява и дясна граници	22
5.4 Конструиране на елипси	26
5.5 Изчисляване на съотношенията	28
6 Резултати от експеримент	28
6.1 Резултати от прагова обработка с метод на Оцу	28
6.2 Резултати от невронни мрежи	29
6.3 Сравнение на резултатите	30
6.4 Подробни резултати	30
6.5 Лимитации на софтуера	36
7 Софтуер	37
7.1 Използвани библиотеки	38
7.2 Графичен интерфейс	38
8 Заключение	38

1 Въведение

Машинното зрение е област на компютърните науки и инженерството, която се занимава с автоматизираната обработка и анализ на визуална информация чрез компютърни системи. Основната му цел е да даде на компютрите способността да "виждат" и разбират околната среда по начин, подобен на човешкото зрение. Машинното зрение има много приложения. Някои от тях са промишлена автоматизация (проверка на качеството, монтаж и сортиране), медицинска диагностика (анализ на рентгенови снимки, снимки от ядрено-магнитен резонанс, откриване на заболявания), сигурност (разпознаване на лица, наблюдение и контрол на достъпа), транспорт (автономни автомобили, системи за помощ при шофиране и трафик) и други.

Машинното зрение може да се разглежда отделно от компютърното зрение ([1]), което е част от компютърните науки. Някои от проблемите, с които се занимава тази област, са класификация (присвояване на категория на изображението въз основа на неговите характеристики), разпознаване на обекти (откриване и идентифициране на обекти в изображение), сегментация на изображения (разделяне на изображението на отделни обекти или области на интерес), проследяване на обекти (проследяване на движението на обектите в последователност от изображения) и други.

Ръчното измерване на параметрите от експериментални снимки на капилярни мостове е неточно и бавно. Автоматизирането на измерванията повишава точността, улеснява работата и спестява време, което позволява провеждането на повече експерименти и повече работа върху теория. Целта на дипломната работа е да се разработи софтуер, който имплементира различни модели, чрез които може да се автоматизира измерването на параметрите (глава 2) на капилярни мостове с минимална грешка ($\sim 2\%$) спрямо ръчните измервания. Софтуерът трябва да може да контролира основни параметри на камерата (експозиция, скорост на заснемане) и да заснема снимки и видеа на капилярните мостове, чиито параметри се измерват. Графичният интерфейс на софтуера, който е показан в глава 7.2, позволява по-добър контрол над целия процес по създаване на структурите, оптимизиране на извършването на експеримента, заснемане на измерените параметри и сравнението им с необработените снимки, контрол над скоростта на заснемане и времето за експозиция на камерата, измерване на параметрите на видео в реално време или на една снимка. Обработването на изображенията в реално време се използва за оптимизация, настройка на системата и наблюдаване на параметрите. В този режим на работа, в много случаи точността на измерванията може да е по-ниска.

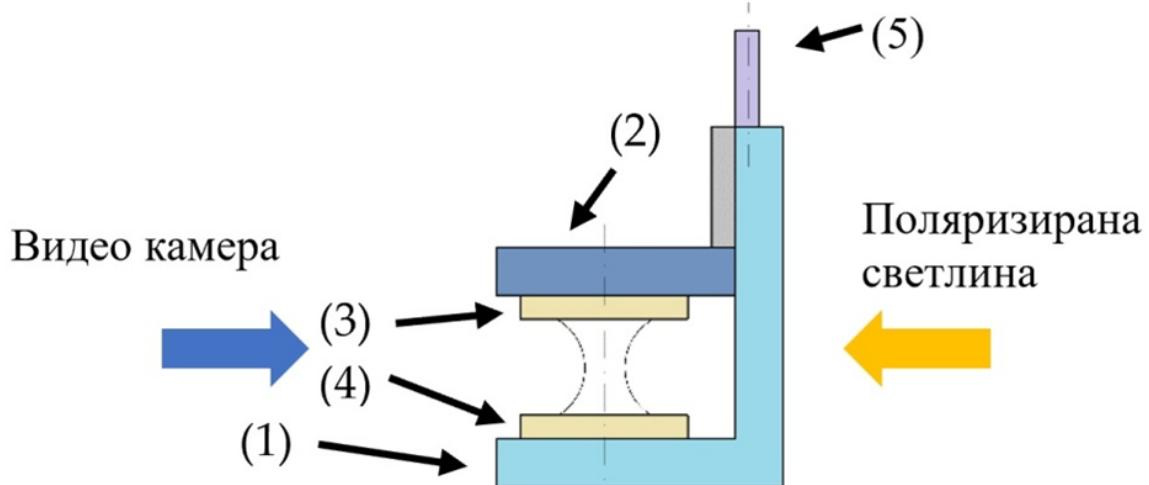
След като са заснети снимки на капилярен мост чрез експерименталната постановка показана в глава 2, трябва да се определят параметрите визуализирани на фиг. 2. Снимките на капилярните мостове от експеримента не са идентични. Те зависят от извършването на самия експеримент, осветлението, отражения, настройките на камерата и т.н. Също така структурите не са симетрични. Тези различия между експериментите не позволяват използването на аналитичен модел за автоматичното измерване на параметрите. Нужно е да се използват други модели, които работят правилно за различните снимки.

2 Експериментална постановка

Експерименталната постановка [2] представлява монтирано оборудване, което се състои от специално проектирана клетка. Тя се състои от две опорни площи (горна и

долна). Долната е фиксирана (точка 1 на фиг. 1), а горната (точка 2 на фиг. 1) може да се движи контролирано нагоре и надолу (точка 5 на фиг. 1), чрез микрометричен винт. Върху плочите са поставени опорни масички (горна показана на точка 3 на фиг. 1 и долната показана на точка 4 на фиг. 1). На тях се поставят предварително почистени покривни стъклa $22 \times 22\text{mm}$ с дебелина 0.13 до 0.16 mm (покривно стъкло за микроскопия на Deltalab). Покривните стъклa се сменят при всяко поставяне на нова капка за създаването на капилярен мост.

Видео камерата, която се използва е Basler ace U acA800-510uc. Използва USB 3.0 интерфейс. Максималната скорост на заснемане, която може да се постигне с нея е 511 кадъра в секунда. Минималното време за експозиция е 59 μs , а максималното е 1 секунда. Резолюцията на камерата е 800×600 пиксела. Скоростта на записване и времето за експозиция може да се променят чрез софтуера. Оптичната система позволява да се записват изображения в поляризирана светлина, което значително увеличава контраста на изображенията. Цялото оборудване е инсталирано на четириосна система за прецизно позициониране, което позволява да се постигне много висока точност при позиционирането на камерата спрямо анализираните структури.



Фиг. 1: Схема на експерименталната постановка (взаимствано от [2])

3 Търсени параметри

Параметрите, които трябва да се определят са следните (Фиг. 2):

- дължина на шийка (neck) - разстоянието между двете най-близки точки
- дължина на горната граница
- дължина на долната граница
- дължина на лявата граница
- дължина на дясната граница
- дължина на оси на лява елипса



Фиг. 2: Примерна снимка на капилярен мост с визуализация на търсените параметрите

- дължина на оси на дясна елипса

В средата на примерната снимка на фиг. 2 е самата течност, която се използва за експеримента. Вдълбинните представляват отражения и те се използват за намиране на параметрите.

Крайният резултат на програмата са дължините на параметрите измерени в μm . За да се достигне до него се измерват дължините на правите, които представляват брой пиксели. След това броят пиксели се конвертира в микрометри чрез следната функция

$$c(p) = \frac{p \times 3659.269}{1920}$$

където 1920 е броят пиксели, на които съответстват $3659.269 \mu\text{m}$, а p са пикселите на текущото изображение, което се измерва.

За измерването на всички тези параметри е нужно да се намерят границите на обектите в изображението. Този проблем се нарича сегментация. Пространството отляво и отдясно на течността е региона, който трябва да се сегментира. Двата региона (лев и десен) ще се наричат обекти за сегментацията или по-кратко обекти. След това трябва да се намерят началните и крайните точки на правите, които представляват дължините на търсените параметрите. За намирането на шийка

(neck) трябва да се построи права между най-близките точки от границите на обектите, за горната граница - между двете горни точки, за долната граница - между двете долнни точки, за лявата граница - между горната и долната леви точки и за дясната граница - между горната и долната десни точки.

Параметрите са нужни, за да се изчислят определени съотношения, които се използват за развиване на теория (глава 5.5). Измерването на параметрите и съотношенията представлява крайния резултат на програмата.

4 Модели за сегментация

Сегментацията е задачата за класифициране на всеки пиксел от изображение. В конкретния случай, пикселите на всяко изображение се разделят на две категории - обекти и фон. За сегментацията на обектите в изображението са имплементирани и сравнени два основни модела и няколко техни вариации с различни конфигурации:

- Прагова обработка (image thresholding) с метод на Оцу (глава 4.1) [3]
 1. Без филтър за увеличаване на контраста
 2. С филтър за увеличаване на контраста (глава 4.1.1)
- Невронна мрежа "Segment Anything" (глава 4.2)([4])
 1. Базов модел "Segment Anything" (глава 4.2.3)
 2. Невронна мрежа "FastSAM" (вариант на "Segment Anything") (глава 4.2.4) ([5])
 3. "Segment Anything" с фина настройка (fine-tuning) (глава 4.2.5)

Преди сегментация изображението се конвертира от RGB цветови модел към полутоново изображение по следния начин:

$$I_{\text{gray}} = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B$$

където

- R е канала на червеното
- G е канала на зеленото
- B е канала на синьото
- константите 0.2989, 0.5870, 0.1140 са стандартни и представляват относителното възприятие на яркостта на червеното, зеленото и синьото на човек

4.1 Прагова обработка с метод на Оцу

Праговата обработка е един от най-често използваните методи за сегментиране на обекти в изображение. Основната идея е да се избере прагова стойност T , която разделя пикселите на две групи: тези с интензитет, по-малък или равен на T , и тези с интензитет, по-голям от T .

$$I'(x, y) = \begin{cases} 0, & \text{ако } I(x, y) \leq T \\ 1, & \text{ако } I(x, y) > T \end{cases}$$

където:

- $I(x, y)$ е интензитет на пиксела на координати (x, y) в оригиналното изображение.
- $I'(x, y)$ е интензитет на пиксела на координати (x, y) в резултатното бинарно изображение.

- T е праговата стойност.

Тук праговата стойност T е константа:

$$T = \text{const}$$

Тази версия на алгоритъма работи за по-прости задачи и за изображения, в които не се среща шум и осветлението е разпределено равномерно. Тези условия не са изпълнени за всички снимки на капилярни мостове, чиито параметри трябва да се определят. За да се сегментират правилно обектите в изображението е нужен алгоритъм с адаптивна прагова стойност. В този случай стойността на T се променя в зависимост от локалните свойства на изображението. Алгоритъмът, който се използва е метод на Оцу ([3]):

Хистограма на изображението (брой пиксели спрямо интензитет):

$$h(z) = \#\{I(x, y) \mid I(x, y) = z\} \quad z \in [0, L]$$

Брой пиксели на обекта:

$$n_o(T) = \sum_{z=T+1}^L h(z)$$

Брой пиксели на фона:

$$n_b(T) = \sum_{z=0}^T h(z)$$

Средна стойност на интензитет на обекта:

$$m_o = \frac{\sum_{z=T+1}^L zh(z)}{n_o(T)}$$

Средна стойност на интензитет на фонда:

$$m_b = \frac{\sum_{z=0}^T zh(z)}{n_b(T)}$$

Цветова дисперсия:

$$\sigma(T) = n_b(T)n_o(T) \left[m_b(T) - m_o(T) \right]^2$$

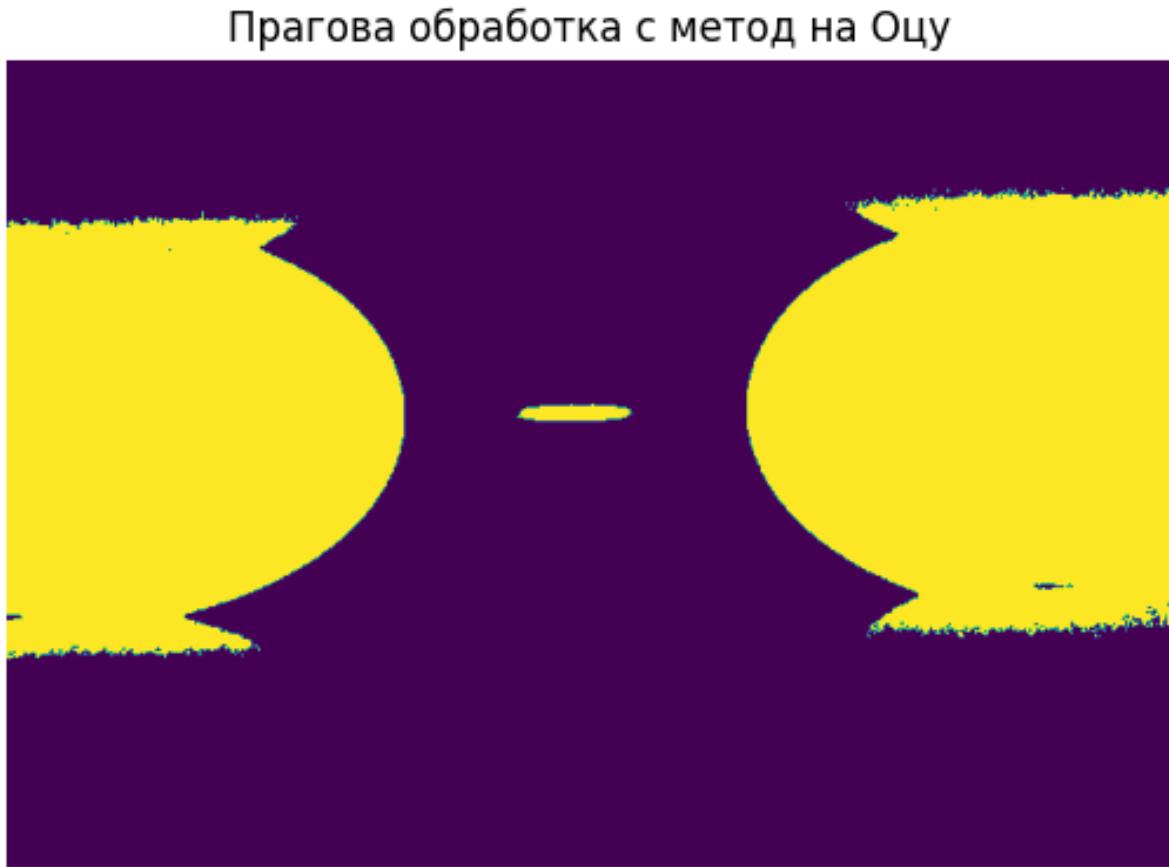
Оптимална прагова стойност:

$$T : \sigma(T) \rightarrow \max$$

Методът на Оцу работи по-добре от праговата обработка с константна прагова стойност, но не дава добри резултати при изображения, чиито хистограми не съдържат ясно отделени двата класа региони за сегментация (обекти и фон), което може да се получи при изображения с отражения и неравномерно разпределено осветление.

Въпреки неточности при измерването на някои от параметрите, методът на Оцу позволява обработване на изображенията в реално време (~ 30 кадъра в секунда на

компютъра, който се използва за провеждането на експеримента). Визуализация на изображение след обработката може да се види на фиг. 3.



Фиг. 3: Изображение след прагова обработка с метод на Оцу

4.1.1 Филтър за увеличаване на контраста

Понеже някои от изображенията нямат висок контраст между обектите и фона (разликата между интензитета на обектите и фона е малка), не се получава правилно сегментиране при използване на метода на Оцу. Този проблем може да се реши като се добави филтър за увеличаване на контраста между обектите и фона (увеличава интензитета само на частите с по-висок интензитет и запазва интензитета на тези с по-нисък интензитет) преди прилагането на праговата обработка.

\mathbf{I} е входното изображение с размери $m \times n \times 3$, където третото измерение представлява трите RGB цветови канала (червено, зелено и синьо), а \mathbf{I}_o е изходното изображение след прилагане на филтъра.

Двата параметъра β и γ са в следните интервали:

$$\beta \in [-255, 255]$$

$$\gamma \in [-127, 127]$$

Ако $\beta \neq 0$

$$s = \begin{cases} \beta & \text{ако } \beta > 0 \\ 0 & \text{ако } \beta \leq 0 \end{cases}$$

$$h = \begin{cases} 255 & \text{ако } \beta > 0 \\ 255 + \beta & \text{ако } \beta \leq 0 \end{cases}$$

$$\alpha_b = \frac{h - s}{255}$$

$$\gamma_b = s$$

Прилагане на яркостта към изображението:

$$\mathbf{I}_o = \alpha_b \cdot \mathbf{I} + \gamma_b$$

Ако $\beta = 0$, входното изображение остава непроменено:

$$\mathbf{I}_o = \mathbf{I}$$

Ако $\gamma \neq 0$

$$f = \frac{131 \cdot (\gamma + 127)}{127 \cdot (131 - \gamma)}$$

$$\alpha_c = f$$

$$\gamma_c = 127 \cdot (1 - f)$$

Прилагане на контраста към изображението:

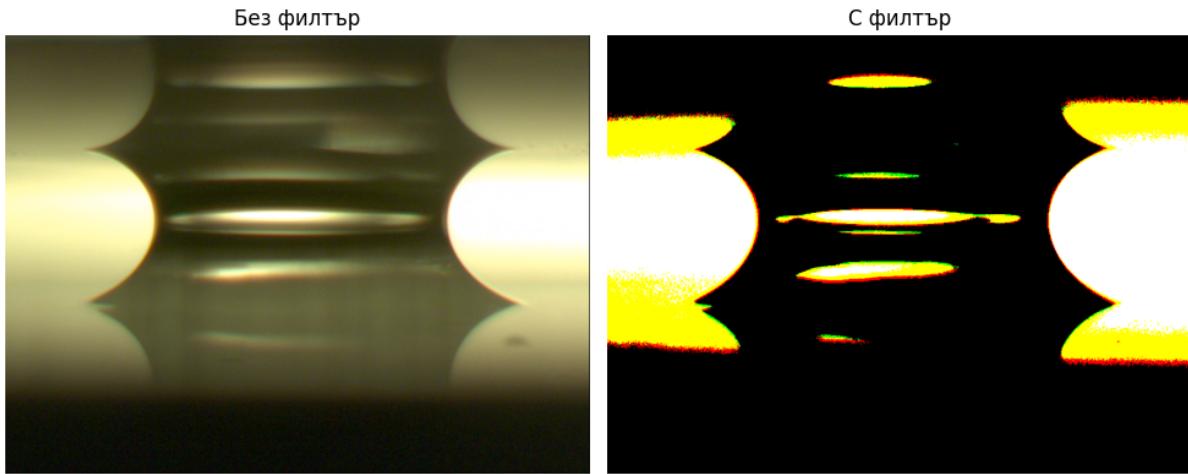
$$\mathbf{I}_o = \alpha_c \cdot \mathbf{I} + \gamma_c$$

където:

- β е яркостта
- γ е контраста

Прилагането на този филтър води до по-точни измервания на параметрите за някои от изображенията (глава 6.1.2).

Сравнение между изображение без филтър за увеличаване на контраста и изображение с филтър може да се види на фиг. 4.



Фиг. 4: Сравнение между изображение без филтър за увеличаване на контраста и изображение с филтър. Използва се 127 за стойност на параметъра γ .

4.2 Невронна мрежа "Segment Anything"

Праговата обработка с метод на Оцу не сегментира правилно всички изображения понеже някои от тях не са с равномерно разпределено осветление или имат твърде много отражения, което затруднява намирането на горните и долните краища на обектите. Друг подход, който е по-универсален от праговата обработка с метод на Оцу и позволява сегментирането на изображения с неравномерно разпределено осветление и отражения е използването на невронни мрежи [6] за сегментацията. Сравнени са резултатите между три невронни мрежи, които са варианти на "Segment Anything". Използвани са базовия модел на "Segment Anything" (глава 4.2), оптимизирана архитектура с конволюционна невронна мрежа "FastSAM" (глава 4.2.4) и модел с фина настройка (fine-tuning) (глава 4.2.5).

4.2.1 Основа на невронните мрежи

Изкуствените невронни мрежи са опростен модел на биологичните невронни мрежи. Те се състоят от слоеве от "неврони", които са свързани помежду си. Всеки "неврон" получава входни данни, обработва ги и предава резултата към следващия слой неврони.

1. Всеки вход x_i се умножава по съответното тегло (weight) w_i .

$$z = \sum_{i=1}^n w_i x_i + b$$

където z е сумата на входните данни умножени по теглата и отместване (bias) b .

2. Резултатът z се подава на активационна функция $\sigma(z)$, която добавя нелинейност към модела. Активационна функция, която се използва често е сигмоида:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

или ReLU (Rectified Linear Unit):

$$\sigma(z) = \max(0, z)$$

3. Изходът на неврона е $\hat{y} = \sigma(z)$.

$$\hat{y} = \sigma \left(\sum_{i=1}^n w_i x_i + b \right)$$

Тези операции се повтарят за всеки "неврон" и всеки слой в мрежата. Мрежата произвежда изход, който може да бъде използван за класификация (включително сегментация), регресия или други задачи.

Пример за напълно свързан слой с 3 входа и 1 изход:

$$z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

$$\hat{y} = \sigma(z)$$

Този процес се повтаря за всички слоеве в мрежата, като изходът на един слой става вход за следващия. В края на мрежата се получава финален изход, който представлява прогнозата (prediction) на модела за съответните входни данни.

4.2.2 Обучение на невронна мрежа

Обучението на невронната мрежа представлява процеса на оптимизиране на теглата и отместванията, така че мрежата да може да прави по-точни прогнози за съответните входни данни. Основният метод за обучение е алгоритъма за обратно разпространение на грешката (backpropagation) ([7]), който използва градиентно спускане (gradient descent).

1. Теглата w и отместванията b се инициализират със случаини стойности.
2. Входните данни x преминават през мрежата, за да се изчисли прогнозата \hat{y} .

$$\hat{y} = \sigma \left(\sum_{i=1}^n w_i x_i + b \right)$$

3. Изчислява се грешката между прогнозата \hat{y} и правилната стойност y . Функция, която често се използва за изчисляването на грешката е средноквадратичната грешка:

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

4. Изчисляват се градиентите на грешката спрямо всяко тегло w и отместване b чрез диференциране на сложни функции с верижното правило (chain rule).

За теглото w :

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_i}$$

За отместването b :

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial b}$$

5. Теглата и отместванията се актуализират с малка стъпка в посоката, противоположна на градиента на грешката, за да се намали грешката. В най-простиия случай стойността на стъпката е константа. За теглото w :

$$w_i := w_i - \eta \frac{\partial L}{\partial w_i}$$

За отместването b :

$$b := b - \eta \frac{\partial L}{\partial b}$$

където η е скоростта на обучение (learning rate).

Този процес се повтаря за всяка извадка в набора от данни за обучаване (training set). Едно пълно преминаване през всички извадки в набора от данни за обучаване се нарича епоха. Целият процес на обучаване обикновено протича за няколко епохи, докато не се минимизира функцията на грешката до желаното ниво.

Примерна стъпка на градиентно спускане:

$$w_i := w_i - \eta (\hat{y} - y) \cdot x_i \cdot \sigma'(z)$$

$$b := b - \eta (\hat{y} - y) \cdot \sigma'(z)$$

където $\sigma'(z)$ е производната на активационната функция.

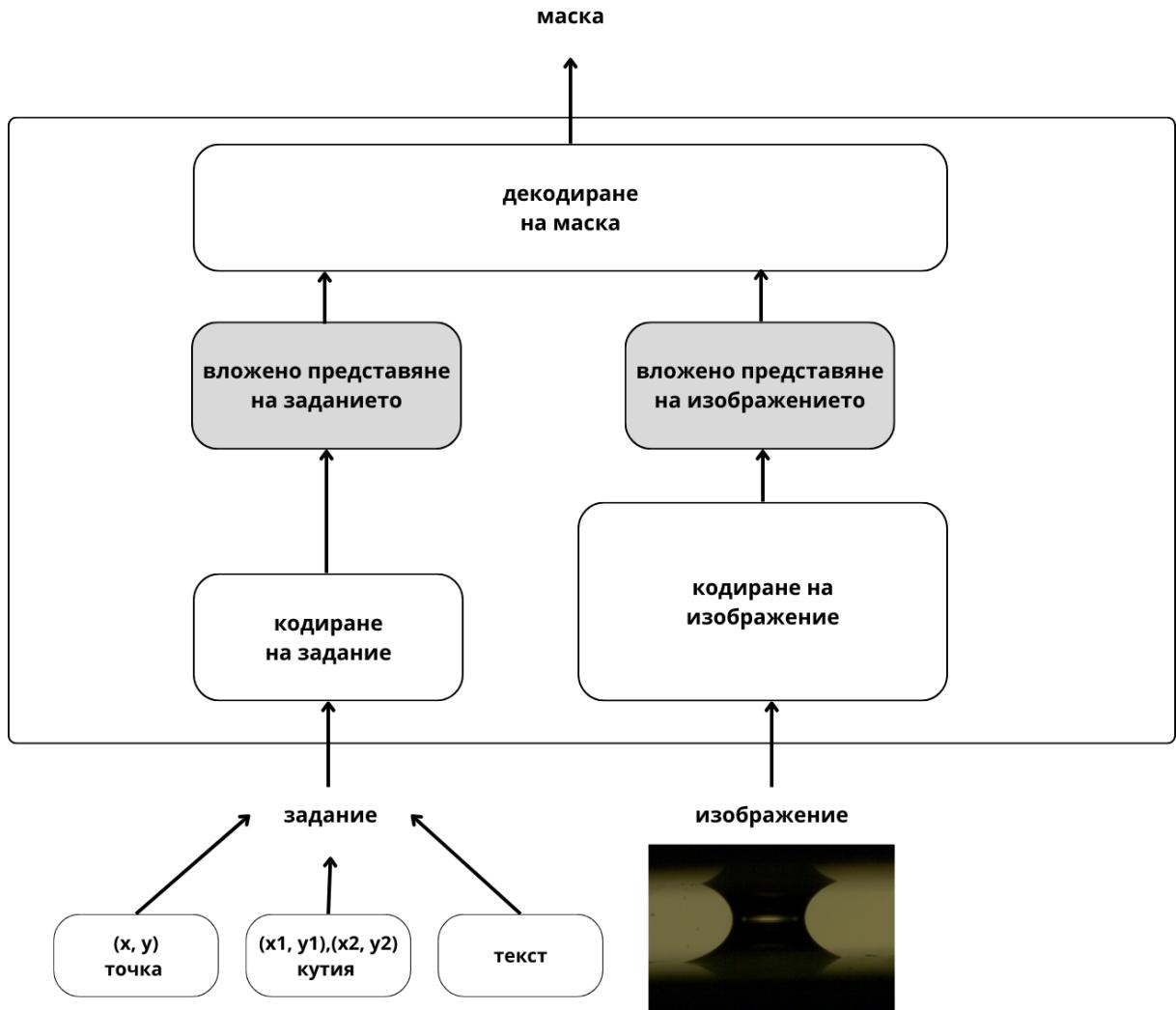
4.2.3 Базов модел "Segment Anything"

За разлика от традиционните модели за сегментация, които изискват специфично моделиране за всяка отделна задача, "Segment Anything" [4] е универсален модел, който се представя добре при голем набор от задачи за сегментация. Моделът може да прави прогнози чрез посочване на определено задание (prompt), което може да е под формата на точки, кутии и маски, които обозначават обекта, който трябва да се сегментира, или под формата на текст, който описва обекта. "Segment Anything" е разделен на три основни по-малки модела:

- кодиране на изображение (image encoder) - представлява невронна мрежа с архитектурата vision transformer ([8]), която е обучена чрез метода masked auto encoding ([9]). Входното изображение на "Segment Anything" модела се трансформира до вектор с намалена размерност чрез тази част (image encoder) на модела. Това се нарича вложено представяне на изображението (image embedding). То представлява обща форма на представяне на изображението, която е устойчива на вариации като промени в осветлението, мащаба и ротацията.
- кодиране на задание (prompt encoder) - създава се вложено представяне на заданието (точки, кутии, текст и маски), което е под формата на вектор с намалена размерност. Ако се ползва текст за задание, той също има вложено представяне, което се създава чрез CLIP ([10])

- декодиране на маска (mask decoder) - взема резултатите от кодирането на изображението и кодирането на заданието, за да направи прогноза под формата на маска, която представлява резултата от сегментацията

Опростена диаграма на архитектурата на "Segment Anything" може да се види на фиг. 5.



Фиг. 5: Архитектура на "Segment Anything"

Базовият модел е невронната мрежа, която е предварително обучена и се използва директно върху снимките на капилярни мостове без да е обучавана на подобни снимки преди. Има три основни предварително обучени модели, които се различават по броя на параметрите (теглата и отместванията). Моделът, който се използва за сегментацията на снимките на капилярни мостове е най-малкия от трите предварително обучени (91 милиона параметри - тегла и отмествания). За задание се използват 2 кутии b_l и b_r дефинирани със следните точки

$$x_l = I_w - (I_w - 50)$$

$$x_r = I_w - 50$$

$$y = \frac{I_h}{2} + 50$$

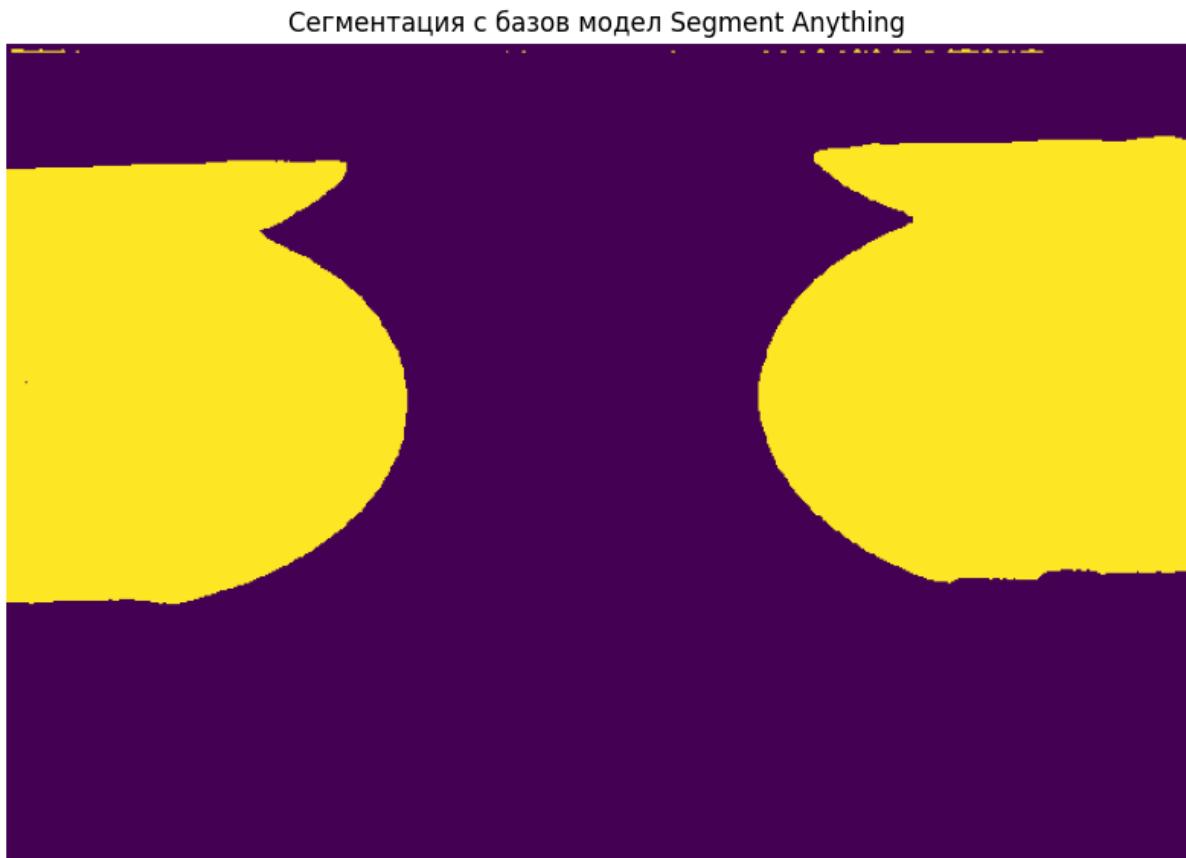
$$b_l = (x_l, y)$$

$$b_r = (x_r, y)$$

където:

- I_w е ширината на изображението
- I_h е височината на изображението

Визуализация на сегментация с базов модел "Segment Anything" на фиг. 6.



Фиг. 6: Сегментация с базов модел "Segment Anything"

4.2.4 Невронна мрежа "FastSAM"

Vision transformer модела [8], който е основната част от архитектурата на "Segment Anything" невронната мрежа, изисква голямо количество изчислителни ресурси. Това прави модела неподходящ за задачи, които трябва да се изпълняват на компютри с малко изчислителни ресурси или задачи, които трябва да работят в реално време. "FastSAM" е по-оптимизирана и по-бърза версия на "Segment Anything". Основната причина за това е използването на конволюционна невронна мрежа вместо vision transformer архитектура.

Конволюционната невронна мрежа [11] е вид невронна мрежа, при която връзката между невроните е подобна на организацията на зрителната система на хората.

Съставена е от конволюционни слоеве, които съдържат филтри. Филтрите във всеки слой обхождат входните данни за слоя. Извършва се скаларно умножение на филтъра и входните данни. Обикновено след всеки конволюционен слой се добавя обединяващ слой (pooling layer). Обединяващите слоеве компресират резултата от конволюционните слоеве (намаляват размерността, но се опитват да запазят важната информация). Най-често използваните видове обединявания са осреднено (average pooling), сумарно (sum pooling) и максимално (max pooling). След конволюционните и обединяващите слоеве се добавят напълно свързани слоеве (всеки "неврон" от слоя е свързан с всеки "неврон" от предишния слой). Целта на напълно свързаните слоеве е да класифицират характеристиките, които са получени като резултат от предишните слоеве, към една от категориите. Използва се активационна функция Softmax

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad \text{за } i = 1, 2, \dots, K$$

Softmax функцията е често използвана при класификационни задачи с множество категории. Тя преобразува вектор от реални числа z във вероятностно разпределение. z_i е i -тия елемент от входния вектор z , а K е броят на категориите. За всеки елемент z_i от входния вектор се изчислява експоненциалната функция e^{z_i} . Тази стъпка увеличава стойностите на входните числа и прави всички стойности положителни. Сумират се всички експоненцирани стойности и всеки елемент e^{z_i} се дели на тази сума. Това гарантира, че получените стойности са в интервала $[0, 1]$ и тяхната сума е равна на 1, което ги прави подходящи за интерпретиране като вероятности. Така се преобразува входния вектор z в изходен вектор $\sigma(z)$, където всяка стойност $\sigma(z_i)$ представлява вероятността входният вектор да принадлежи на i -тия клас.

Пример за опростена архитектура на конволюционна мрежа може да се види на фиг. 7.

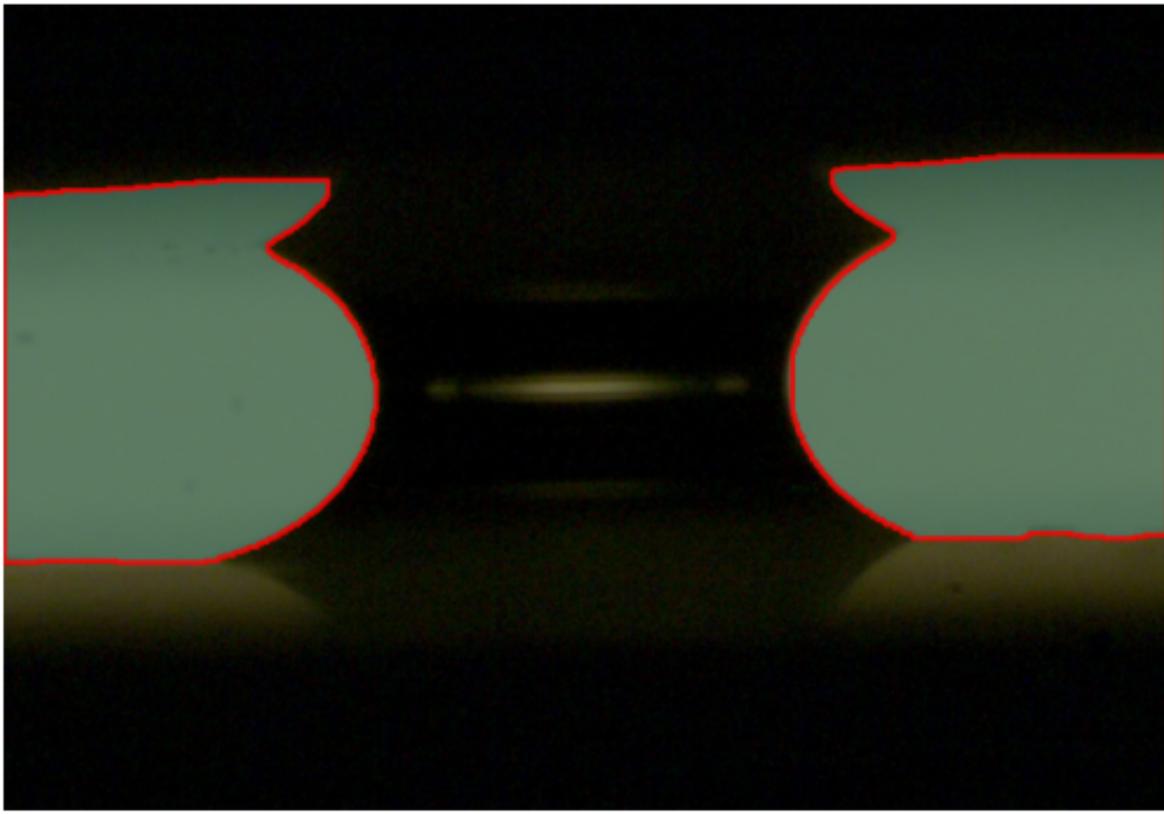


Фиг. 7: Опростена архитектура на конволюционна невронна мрежа. Конволюционният слой съдържа филтър с размер 5×5 , а обединяването е с размер 2×2 .

"FastSAM" се състои от два основни компонента. Вместо vision transformer се използва "Yoloact" архитектурата [12], която представлява конволюционна невронна мрежа. Тя е обучена да прави прогнози за маските на обекти само за две категории - маска и фон (в тази част от модела няма разграничение между категориите както при оригиналната невронна мрежа "Segment Anything"). Вторият компонент на архитектурата е селектирането на маски спрямо заданието, което представлява и крайния резултат на модела. За кодирането на заданието (prompt encoder) се използва подобен модел както при "Segment Anything" невронната мрежа, който също е базиран на CLIP [10]. Резултатите от обработването на изображенията след сегментация с "FastSAM" (глава 6.2.2) в някои случаи са по-точни от "Segment Anything".

Визуализация на сегментация с "FastSAM" на фиг. 8.

Сегментация с FastSAM

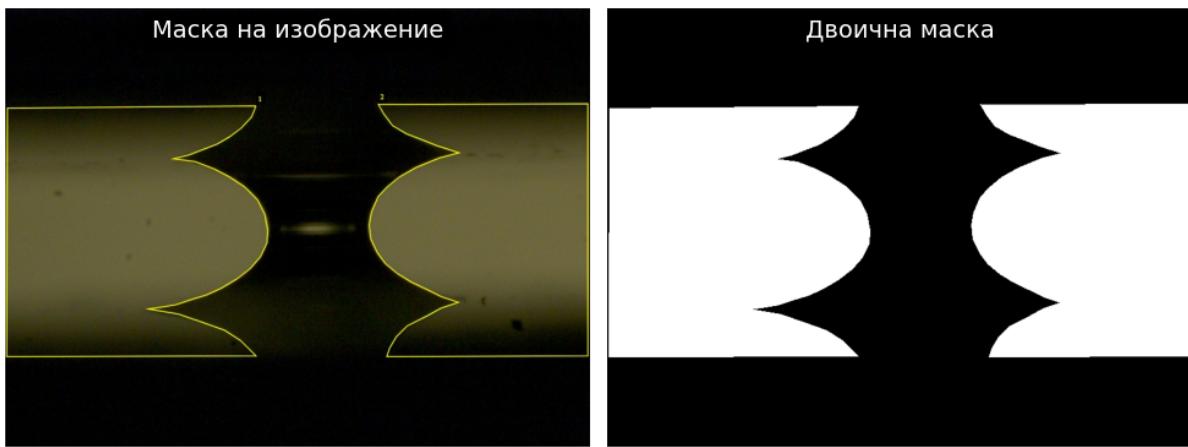


Фиг. 8: Сегментация с "FastSAM"

4.2.5 Невронна мрежа "Segment Anything" с фина настройка

Невронната мрежа, която не е обучена да сегментира изображенията на капилярни мостове не се справя достатъчно точно с някои от тях. За да се реши този проблем, вместо да се обучава напълно нов модел, може да се вземе такъв, който е предварително обучен върху голям брой различни изображения и вече може добре да апроксимира обектите в голяма част от изображения, върху които не е обучен. След това този модел може да се обучи върху специфичните снимки на капилярни мостове, като по този начин се подобрява представянето на модела върху подобен тип изображения без да е нужно да се обучава напълно нов модел.

За целта е нужен набор от данни (dataset), който се състои от различни изображения на капилярни мостове и частите от изображенията с правилните сегментирани региони (маски). Създаването на маските представлява ръчно отбелязване на краищата на обектите във всяко едно изображение в набора от данни. За създаването на маските е използван софтуера VGG Image Annotator (VIA) ([13]). След това маските се конвертират към черно-бяло изображение. Визуализация на създадена маска може да се види на фиг. 9.



Фиг. 9: Мaska на примерно изображение

След като са създадени маски за всички изображения, наборът от данни се разделя на две части - набор от данни за обучаване (training set) и набор от данни за тестване (test set). Невронната мрежа се обучава върху набора от данни за обучаване и след това се тества върху набора от данни за тестване, за да се провери ефективността на модела върху данни, които не е виждал до момента. Използвани са 115 изображения за обучаване и 50 изображения за тестване. Използва се библиотеката PyTorch [14], видео карта NVIDIA Titan V на Физическия факултет и библиотеката Compute Unified Device Architecture (CUDA), която предоставя приложно-програмен интерфейс за видео картите на NVIDIA.

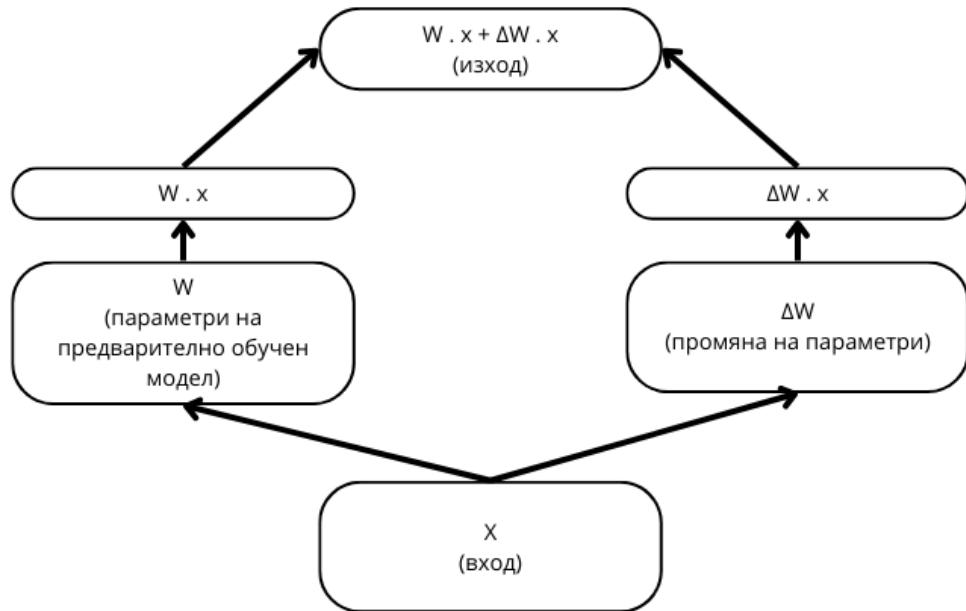
Пълният процес на фина настройка може да бъде много бавен, особено ако моделът е с много параметри. По-ефективна алтернатива на обучаването е използването на адаптер. В този случай параметрите на модела са "замразени" (не се променят по време на обучаване). Адаптерите се добавят като отделни слоеве на модела и след това се обучават. В този случай е избран адаптера LoRA (Low-Rank Adaptation) ([15]). При традиционната фина настройка (фиг. 10) се променят параметрите W на предварително обучена невронна мрежа, за да се адаптира към новата задача. Промените, направени в W по време на фина настройка, са представени като ΔW , така че актуализираните параметри могат да бъдат изразени като $W + \Delta W$. Вместо да модифицираме W директно, чрез LoRA (фиг. 11) се стремим да декомпозираме ΔW . Това разлагане е решаваща стъпка за намаляване на изчислителните разходи, свързани с фината настройка на големи модели. Чрез LoRA се предлага представяне ΔW като произведение на две по-малки матрици, A и B , с по-нисък ранг. Двете матрици имат определени размери $d \times r$ и $r \times d$. Като посочим ранг $r < x$, можем да намалим размера на параметрите и да представим задачата с по-малък ранг. Произведенietо $B \times A$ дава матрица с размерност $d \times d$, така че не се губи информация, но моделът научава ново представяне по време на обучението. По този начин матрицата с актуализираните параметри W' става:

$$W' = W + B \cdot A$$

Матрицата W остава "замразена" (не се актуализира по време на обучение). Матриците B и A са с по-ниска размерност, като тяхното произведение $B \cdot A$ представлява

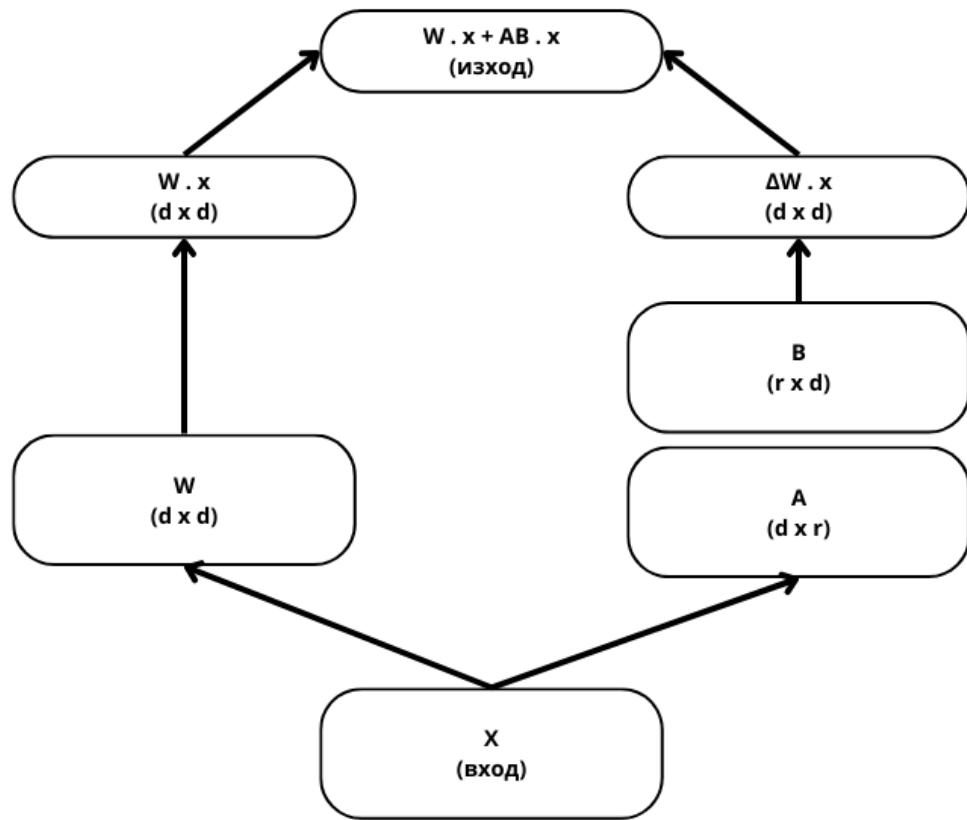
приближение от нисък ранг на ΔW .

Чрез използването на матриците A и B с по-нисък ранг r , броят на обучаемите параметри е значително намален. Например, ако W е $d \times d$ матрица, традиционно актуализирането на W ще включва d^2 параметри. Вместо това, използването на B и A с размери $d \times r$ и $r \times d$ съответно, общият брой параметри намалява до $2 \cdot d \cdot r$, което е много по-малко, когато $r \ll d$.

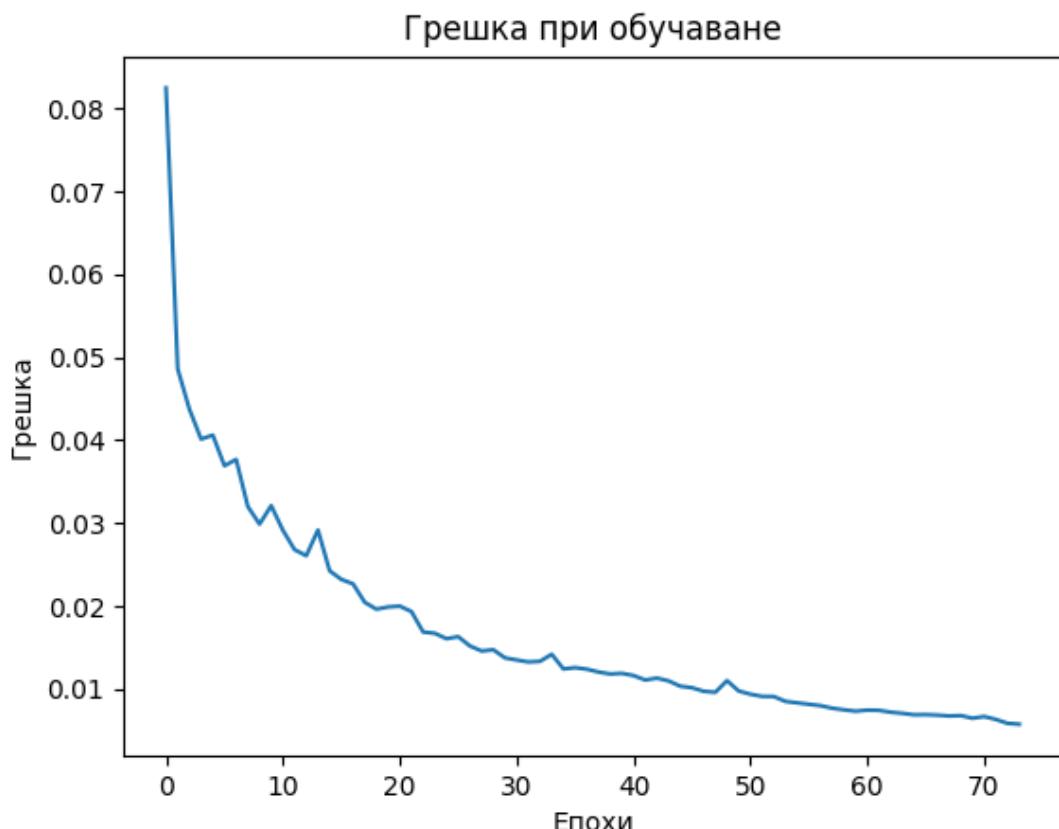


Фиг. 10: Традиционна фина настройка. W не се променя по време на обучение, ΔW се променя по време на обучение

Визуализация на грешката при обучаване е илюстрирана на фиг. 12. Невронната мрежа се обучава за 74 епохи. Използва се скорост на обучение (learning rate) 1×10^{-4} и LoRA ранг 512. Обучаването с тези параметри отнема около 1 час.

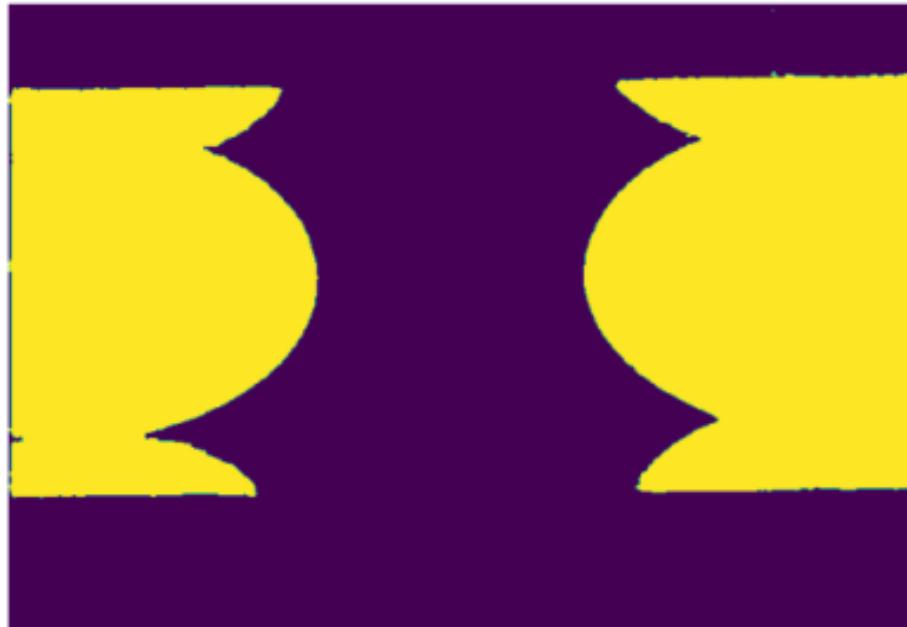


Фиг. 11: Фина настройка с LoRA. ΔW се разлага на две матрици А и В, които имат по-ниска размерност от $d \times d$



Визуализация на изображение след сегментация чрез невронна мрежа с фина настройка може да се види на фиг. 13.

Сегментация чрез невронна мрежа с фина настройка



Фиг. 13: Изображение след сегментанция чрез невронна мрежа с фина настройка

Резултатите от измерванията след сегментация с невронна мрежа с фина настройка (глава 6.2.3) са по-добри от тези след сегментация с прагова обработка с метод на Оцу (таблица 8). Въпреки по-точните измервания, сегментацията с невронна мрежа е по-бавна и не позволява обработване на изображенията от камерата в реално време.

5 Измерване на параметрите

След като изображението е сегментирано (обектите и фонъ са отделени) трябва да се намерят точките, които са необходими за измерването на параметрите.

5.1 Отделяне на контури

Пикселите на границата между обекта и фонъ се наричат контур. Нужно е да се намерят контурите, които описват границите на левия обект и десния обект. Използваният алгоритъм за намиране на двата контура (отляво и отдясно) е топологичен структурен анализ по граница (Topological structural analysis by border follow-

ing) ([16]). Използва се имплементацията на библиотеката OpenCV ([17]). Алгоритъмът може да намери повече от два контура и затова се прилага сортиране по площта на контурите и се вземат двата контура с най-голяма площ. Резултатът представлява списък от пиксели, които формират контура.

Левият контур се обозначава с:

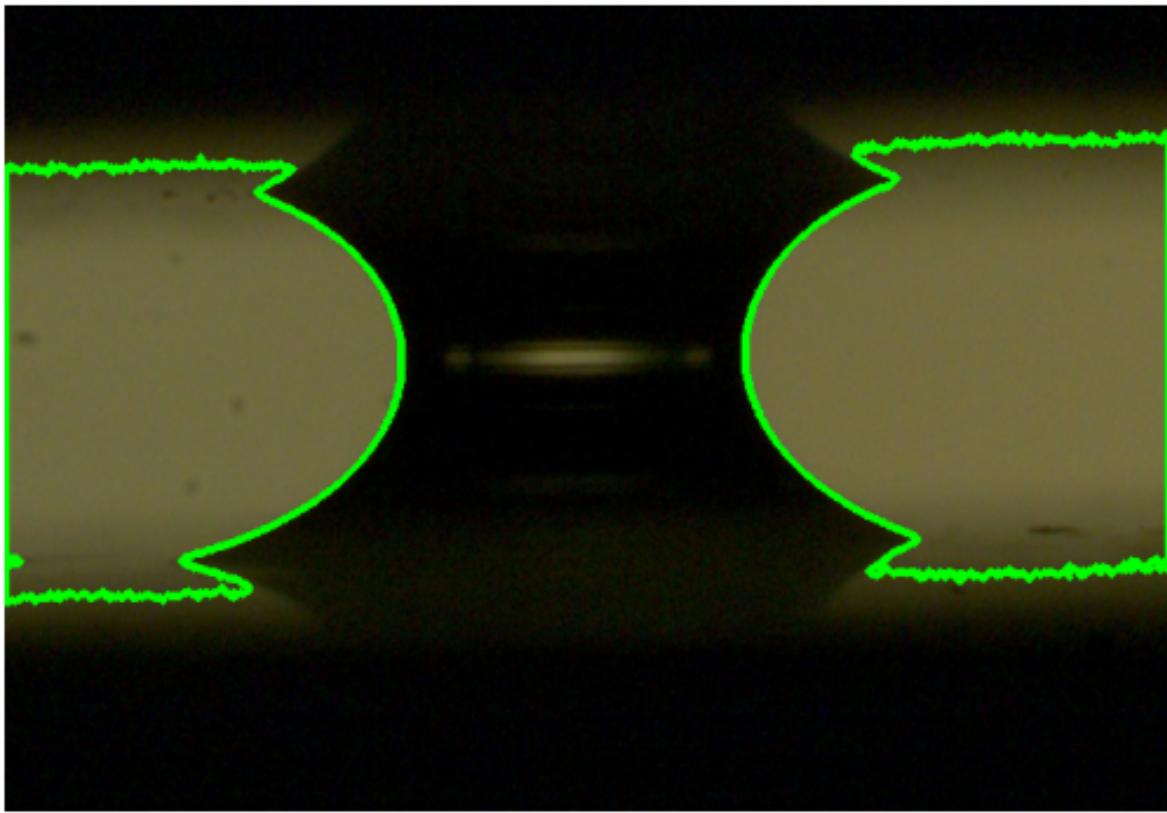
$$C_l = \{(x_i, y_i) \mid i = 1, 2, \dots, n_l\}$$

Десният контур се обозначава с:

$$C_r = \{(x'_j, y'_j) \mid j = 1, 2, \dots, n_r\}$$

Визуализация на изображение след отделяне на контури може да се види на фиг. 14.

Отделяне на контури



Фиг. 14: Изображение след отделяне на контури

5.2 Измерване на шийка (neck)

За да се измери дължината на шийката (neck) е нужно да се намерят най-близките точки от двете страни (най-дясната точка на левия контур и най-лявата точка на десния контур).

За контурите $C_l = \{(x_i, y_i) \mid i = 1, 2, \dots, n_l\}$ и $C_r = \{(x'_j, y'_j) \mid j = 1, 2, \dots, n_r\}$, можем да изчислим дължината на шийката по следния начин:

1. Изчисляване на евклидовото разстояние между всички точки:

$$d_{i,j} = \sqrt{(x_i - x'_j)^2 + (y_i - y'_j)^2}$$

където (x_i, y_i) са точките в C_l и (x'_j, y'_j) са точките в C_r .

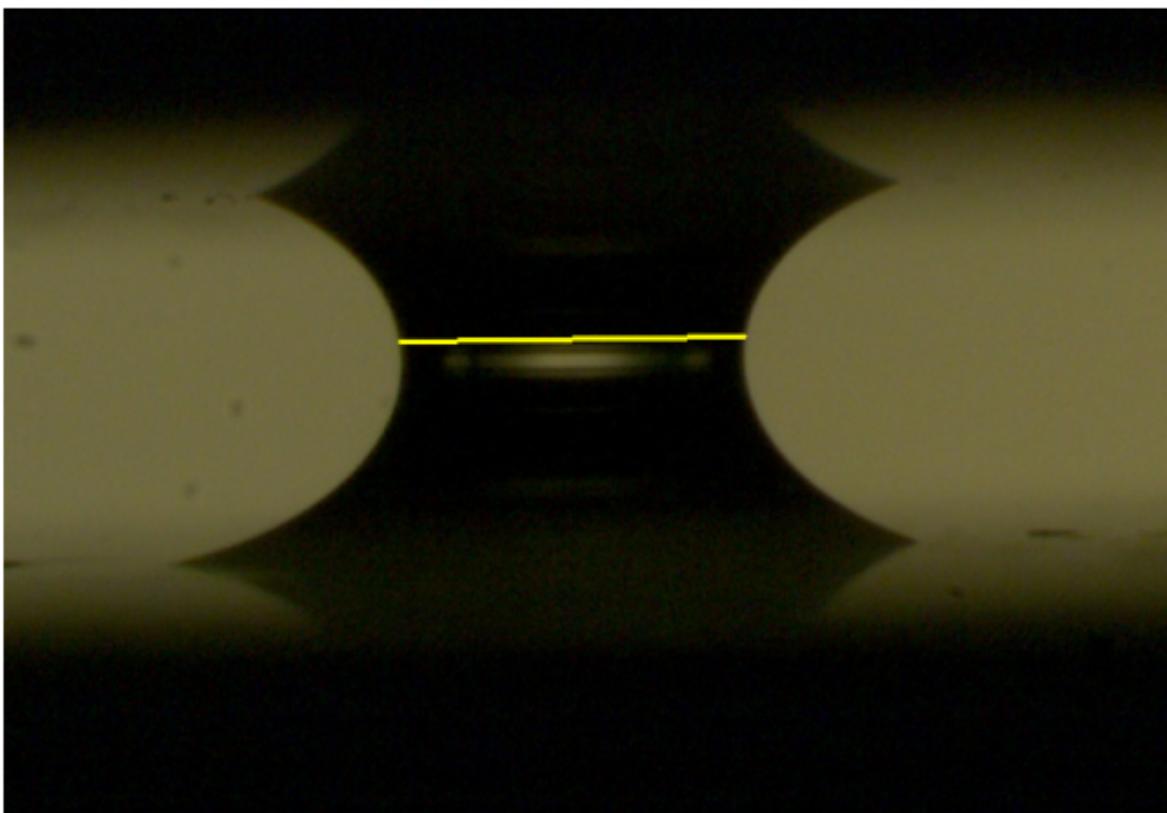
2. Намиране на индексите на минималното разстояние:

$$(i, j) = \arg \min_{i,j} d_{i,j}$$

Най-близките точки са $(x_i, y_i), (x'_j, y'_j)$

Визуализация на измерването може да се види на фиг. 15.

Измерване на шийка (neck)



Фиг. 15: Изображение с измерена шийка (neck)

5.3 Измерване на горна, долна, лява и дясна граници

За крайни точки на горната и долната граница се приемат най-вдълбнатите точки от дясната страна на левия контур и най-вдълбнатите точки от лявата страна на десния контур. За да се намерят се измерва изпъкналата обвивка (convex hull) на двата контура. Изпъкналата обвивка представлява най-малкото множество точки, които ограждат целия контур, като образуват най-малкото външно множество, което

е изпъкнало. Използва се имплементацията на библиотеката OpenCV ([17]), която използва следния алгоритъм [18].

За двата контура

$$C_l = \{(x_i, y_i) \mid i = 1, 2, \dots, n_l\}$$

$$C_r = \{(x'_j, y'_j) \mid j = 1, 2, \dots, n_r\}$$

получаваме изпъкнналите обвивки

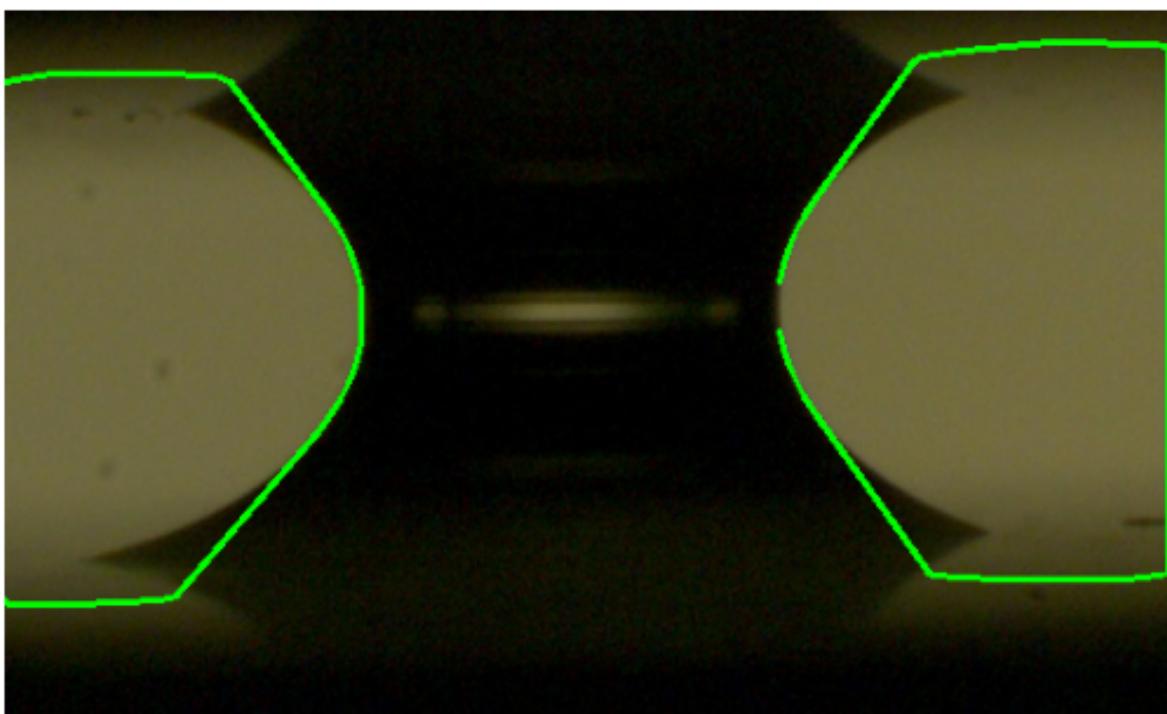
$$H_l = H(C_l)$$

$$H_r = H(C_r)$$

За C_l , изпъкналата обвивка H_l е представена от точките $\{(x_i, y_i) \mid i = 1, 2, \dots, n_l\}$. За C_r , изпъкналата обвивка H_r е представена от точките $\{(x'_j, y'_j) \mid j = 1, 2, \dots, n_r\}$.

Визуализация на намерена изпъкнала обвивка може да се види на фиг. 16.

Изпъкнала обвивка



Фиг. 16: Изображение с намерени изпъкнали обвивки

След това се намират дефектите на изпъкналост (convexity defects). Те представляват всяко отклонение на контура от изпъкналата му обвивка. Най-голямото отклонение е най-вдълбнатата точка. Използва се алгоритъм на библиотеката OpenCV ([17]).

За контурите C_l , C_r и техните изпъкнали обвивки H_l , H_r , дефектите на изпъкналост се получават от контурите и изпъкнналите обвивки

$$D_l = D(C_l, H_l)$$

$$D_r = D(C_r, H_r)$$

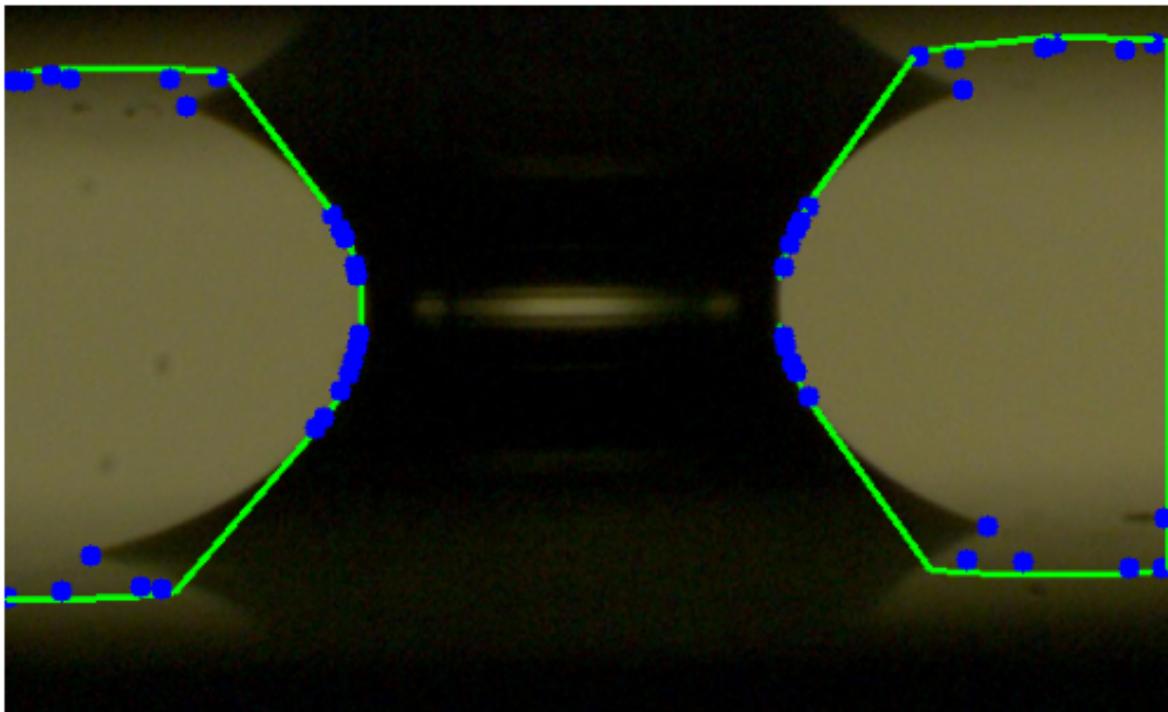
и съдържат множество от дефекти, където всеки дефект d_i представлява наредена двойка (f, d)

където

- f е индекса на най-далечната от изпъкналата обвивка точка
- d е разстоянието между най-далечната точка и изпъкналата обвивка

Визуализация на дефектите на изпъкналост може да се види на фиг. 17.

Дефекти на изпъкналост



Фиг. 17: Изображение с намерени дефекти на изпъкналост

Нека p_u и p_d са двете най-отдалечени (най-вдълбнати) точки на контура, а max наредена двойка (max_0, max_1) от двете най-големи разстояния.

За всяко разстояние между най-далечната точка и изпъкналата обвивка d_i в D , съответната точка в контура C се намира чрез индекса f_i и се означава с C_{f_i} . След това се извършват следните стъпки:

ако $d_i > max_0$:

$$\begin{cases} max_1 \leftarrow max_0 \\ max_0 \leftarrow d_i \\ p_u \leftarrow p_d \\ p_d \leftarrow C_{f_i} \end{cases}$$

или ако $d_i > max_1$:

$$\begin{cases} max_1 \leftarrow d_i \\ p_u \leftarrow C_{f_i} \end{cases}$$

Същите стъпки се повтарят за двата контура и така се получават четирите най-вдълъбнати точки, чрез които могат да се намерят търсените параметри:

За да се намерят дължините на параметрите U, D, L, R чрез точките U_l, U_r, D_l, D_r , където

- U е дължината на горната граница
- D е дължината на долната граница
- L е дължината на лявата граница
- R е дължината на дясната граница
- U_l е горната точка на левия контур
- U_r е горната точка на десния контур
- D_l е долната точка на левия контур
- D_r е долната точка на десния контур

се изчислява евклидовото разстояние за съответните граници

$$E(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

$$U = E(U_l, U_r)$$

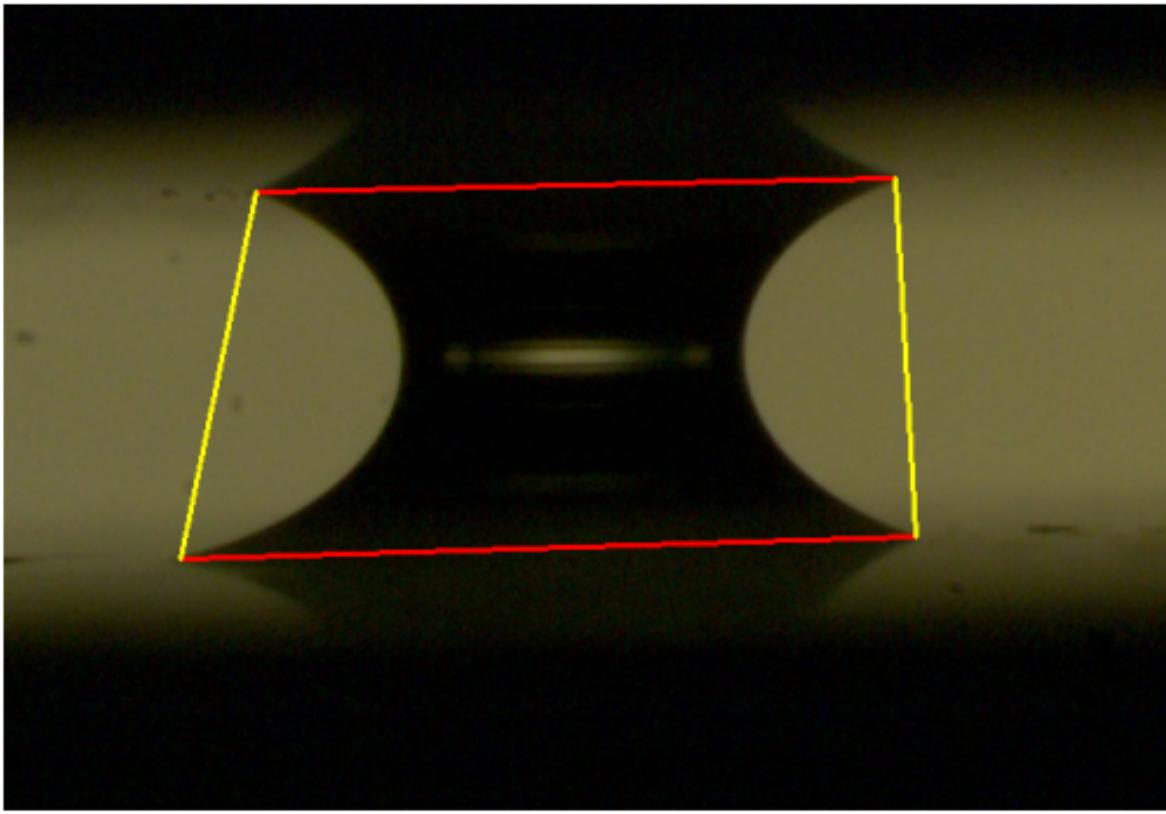
$$D = E(D_l, D_r)$$

$$L = E(U_l, D_l)$$

$$R = E(U_r, D_r)$$

Визуализация на измерените граници може да се види на фиг. 18.

Измерване на горна, долната, лява и дясна граници



Фиг. 18: Изображение с измерени горна, долната, лява и дясна граници

5.4 Конструиране на елипси

За конструирането на елипсите се използва имплементация на библиотеката OpenCV [17]. Нужно е първо да се намерят контурите. Точките на контурите се използват, за да се конструират съответните елипси. Имплементацията е базирана на алгоритъма [19]. След това се намират голямата ос и малката ос на всяка от двете елипси.

Конструираната елипса има следните параметри:

$$E = ((x_c, y_c), (d_1, d_2), \theta)$$

където:

- (x_c, y_c) са координатите на центъра на елипсата
- d_1 и d_2 са дълчините на голямата и малката ос
- θ е ъгълът на наклона на елипсата спрямо хоризонталата

Определя се дължината на голямата полуос:

$$r_{\text{major}} = \frac{\max(d_1, d_2)}{2}$$

Коригира се ъгъла:

$$\text{ако } \theta > 90^\circ \text{ тогава } \theta = \theta - 90^\circ \text{ иначе } \theta = \theta + 90^\circ$$

Намират се точките на краищата на голямата ос:

$$\begin{aligned}x_1 &= x_c + r_{\text{major}} \cos(\theta) \\y_1 &= y_c + r_{\text{major}} \sin(\theta) \\x_2 &= x_c + r_{\text{major}} \cos(\theta + 180^\circ) \\y_2 &= y_c + r_{\text{major}} \sin(\theta + 180^\circ)\end{aligned}$$

Определя се дължината на малката полуос:

$$r_{\text{minor}} = \frac{\min(d_1, d_2)}{2}$$

Коригира се ъгъла отново:

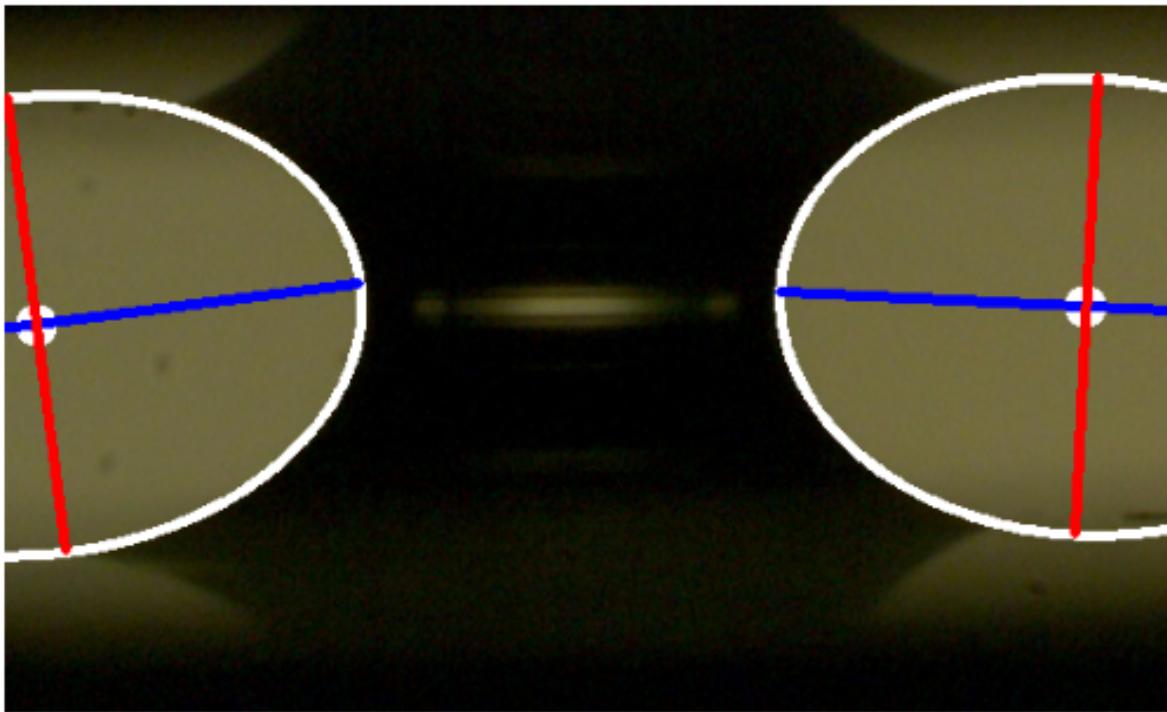
ако $\theta > 90^\circ$ тогава $\theta = \theta - 90^\circ$ иначе $\theta = \theta + 90^\circ$

Намират се точките на краищата на малката ос:

$$\begin{aligned}x_1 &= x_c + r_{\text{minor}} \cos(\theta) \\y_1 &= y_c + r_{\text{minor}} \sin(\theta) \\x_2 &= x_c + r_{\text{minor}} \cos(\theta + 180^\circ) \\y_2 &= y_c + r_{\text{minor}} \sin(\theta + 180^\circ)\end{aligned}$$

Визуализация на конструирани елипси и техните оси може да се види на фиг. 19.

Конструиране на елипси



Фиг. 19: Конструиране на елипси и намиране на големи и малки оси

5.5 Изчисляване на съотношенията

След като имаме всички необходими измервания, можем да изчислим съотношенията. Изчисляваме ги по следните формули като използваме параметрите, които са дефинирани в глава 5.2 и глава 5.3:

Изчисляване на база:

$$b = \frac{U + D}{2}$$

Изчисляване на височина:

$$h = \frac{L + R}{2}$$

Изчисляване на x :

$$x = \frac{b}{n}$$

Изчисляване на y :

$$y = \frac{h}{n}$$

6 Резултати от експеримент

Сравнени са резултатите от програмата с ръчната обработка на изображенията и е показана грешката в измерванията на програмата. Измерването на грешката е чрез средна абсолютна процентна грешка (mean absolute percentage error):

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \left| \frac{A_i - F_i}{A_i} \right| \times 100$$

Където:

- n е броят на наблюденията.
- A_i е теоретичната (реалната) стойност за i -тото наблюдение.
- F_i е експерименталната (прогнозната) стойност за i -тото наблюдение.

6.1 Резултати от прагова обработка с метод на Оцу

Резултатите от прагова обработка с метод на Оцу не са достатъчно точни за някои от изображенията поради лимитациите на алгоритъма. Въпреки това, чрез алгоритъма може да се обработват изображенията от камерата в реално време, което позволява настройване на експерименталната постановка по такъв начин, че алгоритъмът да прави по-точни измервания.

6.1.1 Без филтър за увеличаване на контраста

На таблица 7 са резултатите от измерванията при прагова обработка с метод на Оцу без филтър за увеличаване на контраста. Желаната точност ($\sim 2\%$) е постигната при голяма част от тестовите изображения. Алгоритъмът не успява да сегментира правилно определени изображения поради лимитациите на софтуера (глава 6.5). Това води до високи проценти грешка за някои от параметрите и увеличава средната грешка. Средната грешка е показана на таблица 1.

Грешка	Долна граница	Лява граница	Шийка	Дясна граница	Горна граница
Средно	3.16 %	5.44 %	2.85 %	9.91 %	5.29 %

Табл. 1: Средна грешка за прагова обработка с метод на Оцу без филтър за увеличаване на контраста

6.1.2 С филтър за увеличаване на контраста

На таблица 8 са резултатите от измерванията при прагова обработка с метод на Оцу с приложен филтър за увеличаване на контраста. Прилагането на този филтър води до неправилно сегментиране на някои от изображенията поради неуспешно намиране на двета отделни контура, но подобрява точността за някои от параметрите при определени изображения. Измерването на шийката се осъществява върху изображението преди да бъде приложен филтър, което води до идентични резултати като тези от сегментиране с метод на Оцу без филтър за увеличаване на контраста. Използва се 127 за стойност на параметъра γ , който е дефиниран в глава 4.1.1. Средната грешка е показана на таблица 2.

Грешка	Долна граница	Лява граница	Шийка	Дясна граница	Горна граница
Средно	4.63 %	9.45 %	2.85 %	18.51 %	4.75 %

Табл. 2: Средна грешка за прагова обработка с метод на Оцу с приложен филтър за увеличаване на контраста. В средния резултат са включени само успешно сегментираните изображения.

6.2 Резултати от невронни мрежи

Невронните мрежи подобряват измерванията за някои от параметрите, но изискват повече изчислителни ресурси.

6.2.1 Базов модел "Segment Anything"

На таблица 9 са резултатите от измерванията чрез базов модел "Segment Anything". Базовата невронна мрежа успява да сегментира точно част от изображенията без да е обучавана върху подобни примери предварително. Средната грешка за шийката е по-ниска отколкото при метод на Оцу. Също така, разликата в грешката при измерване на горна граница и долна граница, в сравнение със сегментация чрез метод на Оцу, е по-малка от 1 %. Средната грешка е показана на таблица 3.

Грешка	Долна граница	Лява граница	Шийка	Дясна граница	Горна граница
Средно	6.76 %	21.33 %	1.20 %	24.24 %	5.93 %

Табл. 3: Средна грешка за базов модел "Segment Anything"

6.2.2 Невронна мрежа "FastSAM"

На таблица 10 са резултатите от измерванията чрез конволюционна невронна мрежа "FastSAM". Тази невронна мрежа изисква по-малко изчислителни ресурси от "Segment Anything", но не постига същата точност. Средната грешка е показана на таблица 4.

Грешка	Долна граница	Лява граница	Шийка	Дясна граница	Горна граница
Средно	8.07 %	15.15 %	9.65 %	16.31 %	7.69 %

Табл. 4: Средна грешка за невронна мрежа "FastSAM"

6.2.3 Невронна мрежа с фина настройка

На таблица 11 са резултатите от измерванията чрез невронна мрежа с фина настройка. Този модел постига най-висока точност за 3 от параметрите - добра граница, лява граница и шийка. Разликата в грешката при измерване на дясна граница е по-малка от 1 % спрямо най-точния модел за този параметър. Сегментирането на областта, в която се намира горната граница не е толкова точно колкото сегментирането чрез метод на Оцу. Средната грешка е показана на таблица 5.

Грешка	Долна граница	Лява граница	Шийка	Дясна граница	Горна граница
Средно	1.93 %	3.35 %	1.03 %	10.49 %	9.10 %

Табл. 5: Резултати от невронна мрежа с фина настройка

6.3 Сравнение на резултатите

Модел	Долна граница	Лява граница	Шийка	Дясна граница	Горна граница
Метод на Оцу (без филтър)	3.16 %	5.44 %	2.85 %	9.91 %	5.29 %
Метод на Оцу (с филтър)	4.63 %	9.45 %	2.85 %	18.51 %	4.75 %
Базов "Segment Anything"	6.76 %	21.33 %	1.20 %	24.24 %	5.93 %
"FastSAM"	8.07 %	15.15 %	9.65 %	16.31 %	7.69 %
Фина настройка	1.93 %	3.35 %	1.03 %	10.49 %	9.10 %

Табл. 6: Сравнение на резултати

6.4 Подробни резултати

Показани са резултатите от измерванията за всяко едно от изображенията в набора от данни за тестване за всеки модел.

Изображение	Долна граница	Лява граница	Шийка	Дясна граница	Горна граница
1	9.58 %	80.97 %	1.84 %	22.18 %	44.19 %
2	6.47 %	3.73 %	2.24 %	77.85 %	70.87 %
3	9.83 %	27.99 %	1.18 %	52.97 %	79.45 %
4	13.83 %	1.36 %	34.31 %	53.73 %	2.90 %
5	2.41 %	1.24 %	0.74 %	0.34 %	4.70 %
6	1.95 %	0.65 %	0.68 %	1.64 %	3.20 %
7	30.20 %	2.97 %	73.99 %	69.14 %	0.87 %
8	0.38 %	7.61 %	0.23 %	5.57 %	0.14 %
9	1.36 %	1.26 %	0.95 %	0.62 %	2.16 %
10	0.15 %	1.14 %	1.09 %	0.07 %	1.44 %
11	0.67 %	0.34 %	1.39 %	4.04 %	0.59 %
12	0.36 %	1.70 %	0.55 %	1.61 %	0.28 %
13	0.67 %	0.73 %	0.25 %	5.07 %	0.70 %
14	0.01 %	0.33 %	0.51 %	1.90 %	0.68 %
15	1.34 %	0.36 %	0.49 %	5.59 %	0.18 %
16	0.84 %	1.05 %	0.25 %	2.39 %	0.34 %
17	0.67 %	0.09 %	0.24 %	3.74 %	1.02 %
18	0.50 %	1.09 %	0.48 %	2.89 %	0.51 %
19	0.18 %	1.56 %	0.23 %	1.93 %	0.34 %
20	4.44 %	0.33 %	0.74 %	4.21 %	0.34 %
21	0.68 %	73.61 %	0.47 %	85.93 %	1.14 %
22	0.51 %	2.54 %	0.03 %	3.33 %	1.56 %
23	0.34 %	0.59 %	0.26 %	1.99 %	0.68 %
24	0.01 %	0.57 %	0.24 %	3.33 %	0.01 %
25	0.33 %	2.82 %	0.26 %	0.37 %	0.68 %
26	1.02 %	0.37 %	0.52 %	4.26 %	0.17 %
27	0.68 %	1.57 %	0.24 %	2.85 %	1.90 %
28	1.36 %	0.82 %	0.53 %	2.86 %	1.21 %
29	0.51 %	0.11 %	0.02 %	2.44 %	1.05 %
30	3.18 %	1.87 %	3.41 %	1.34 %	0.91 %
31	2.11 %	0.51 %	0.57 %	1.00 %	1.96 %
32	3.25 %	1.03 %	0.00 %	1.51 %	0.93 %
33	3.82 %	8.25 %	2.04 %	6.15 %	0.01 %
34	1.04 %	0.91 %	0.51 %	2.85 %	1.04 %
35	6.99 %	0.11 %	0.01 %	0.32 %	0.01 %
36	7.59 %	0.62 %	0.01 %	8.73 %	1.60 %
37	0.53 %	1.13 %	0.80 %	5.43 %	2.18 %
38	1.59 %	1.60 %	0.80 %	5.83 %	3.98 %
39	1.95 %	3.06 %	1.08 %	4.05 %	4.87 %
40	2.48 %	2.14 %	1.33 %	3.71 %	5.25 %
41	3.02 %	3.05 %	1.06 %	2.23 %	5.45 %
42	6.19 %	3.94 %	0.53 %	2.08 %	0.88 %
43	0.87 %	3.05 %	0.25 %	4.43 %	1.94 %
44	0.02 %	2.92 %	0.28 %	0.55 %	2.01 %
45	6.57 %	1.25 %	2.08 %	2.11 %	2.86 %
46	6.96 %	2.70 %	0.79 %	1.80 %	2.34 %
47	7.95 %	1.10 %	1.05 %	4.70 %	0.91 %
48	0.71 %	5.18 %	0.27 %	4.62 %	0.75 %
49	0.01 %	7.88 %	0.48 %	2.66 %	0.36 %
50	0.00 %	0.43 %	0.01 %	4.55 %	0.87 %
Средно	3.16 %	5.44 %	2.85 %	9.91 %	5.29 %

Табл. 7: Резултати от прагова обработка с метод на Оцу без филтър за увеличаване на контрастта

Изображение	Долна граница	Лява граница	Шийка	Дясна граница	Горна граница
1			1.84 %		
2			2.24 %		
3			1.18 %		
4			34.31 %		
5			0.74 %		
6			0.68 %		
7			73.99 %		
8			0.23 %		
9	6.55 %	5.04 %	0.95 %	93.95 %	15.28 %
10	10.79 %	7.71 %	1.09 %	0.92 %	12.87 %
11			1.39 %		
12			0.55 %		
13	0.51 %	1.65 %	0.25 %	4.57 %	0.35 %
14	0.18 %	0.77 %	0.51 %	0.97 %	0.17 %
15	0.72 %	0.09 %	0.49 %	97.26 %	5.33 %
16	1.12 %	0.55 %	0.25 %	23.94 %	1.86 %
17	1.28 %	0.39 %	0.24 %	25.82 %	0.36 %
18	1.36 %	0.67 %	0.48 %	2.90 %	2.25 %
19	0.16 %	1.11 %	0.23 %	1.93 %	0.68 %
20	2.61 %	0.36 %	0.74 %	10.03 %	0.87 %
21	5.24 %	9.22 %	0.47 %	26.81 %	1.85 %
22	3.09 %	1.97 %	0.03 %	3.32 %	3.99 %
23	1.19 %	0.17 %	0.26 %	2.94 %	2.24 %
24	1.42 %	0.53 %	0.24 %	25.26 %	0.54 %
25	0.49 %	27.41 %	0.26 %	5.05 %	1.06 %
26	3.41 %	1.91 %	0.52 %	1.82 %	2.97 %
27	2.91 %	0.64 %	0.24 %	2.36 %	4.68 %
28	3.24 %	0.79 %	0.53 %	3.30 %	4.34 %
29	2.77 %	0.13 %	0.02 %	1.93 %	4.39 %
30	6.01 %	3.79 %	3.41 %	1.83 %	4.75 %
31	4.94 %	0.01 %	0.57 %	0.01 %	4.47 %
32	6.67 %	1.53 %	0.00 %	1.95 %	5.04 %
33	7.83 %	7.28 %	2.04 %	7.05 %	4.39 %
34	3.41 %	0.39 %	0.51 %	77.61 %	5.88 %
35	6.29 %	0.62 %	0.01 %	0.83 %	0.01 %
36	3.04 %	9.77 %	0.01 %	90.90 %	7.17 %
37	3.40 %	2.03 %	0.80 %	3.59 %	5.47 %
38	5.86 %	2.40 %	0.80 %	4.09 %	6.88 %
39	9.26 %	1.90 %	1.08 %	3.19 %	8.12 %
40	23.64 %	90.45 %	1.33 %	3.75 %	13.04 %
41	37.68 %	60.98 %	1.06 %	97.21 %	33.13 %
42	0.88 %	3.92 %	0.53 %	0.48 %	1.94 %
43	0.70 %	3.59 %	0.25 %	2.56 %	3.36 %
44	0.17 %	0.82 %	0.28 %	0.89 %	3.30 %
45	9.65 %	97.54 %	2.08 %	52.04 %	5.45 %
46	5.40 %	12.85 %	0.79 %	28.95 %	7.25 %
47	0.57 %	2.52 %	1.05 %	15.60 %	0.54 %
48	0.53 %	6.48 %	0.27 %	5.06 %	1.51 %
49	0.17 %	7.83 %	0.48 %	3.60 %	1.09 %
50	0.18 %	0.03 %	0.01 %	4.10 %	1.23 %
Средно	4.63 %	9.45 %	2.85 %	18.51 %	4.75 %

Табл. 8: Резултати от прагова обработка с метод на Оцу с приложен филтър за увеличаване на контраста. Празните клетки в таблицата означават неуспешно сегментиране на изображението. В средния резултат са включени само успешно сегментираните изображения.

Изображение	Долна граница	Лява граница	Шийка	Дясна граница	Горна граница
1	2.54 %	1.60 %	0.79 %	0.25 %	0.21 %
2	0.98 %	1.15 %	0.74 %	4.97 %	1.54 %
3	2.47 %	3.18 %	2.40 %	1.39 %	0.89 %
4	19.41 %	12.51 %	1.71 %	1.89 %	0.20 %
5	34.64 %	11.11 %	1.28 %	11.89 %	3.02 %
6	19.13 %	10.20 %	1.45 %	1.64 %	0.83 %
7	16.61 %	15.33 %	0.65 %	0.32 %	1.06 %
8	16.21 %	32.90 %	1.20 %	40.39 %	6.66 %
9	10.88 %	0.81 %	2.19 %	9.64 %	0.07 %
10	7.03 %	1.24 %	2.61 %	36.96 %	8.46 %
11	3.00 %	1.16 %	2.26 %	5.30 %	0.07 %
12	2.27 %	1.76 %	2.84 %	0.04 %	2.27 %
13	2.58 %	0.69 %	0.50 %	2.13 %	0.35 %
14	8.51 %	45.19 %	2.06 %	41.92 %	3.63 %
15	1.78 %	1.88 %	0.50 %	49.81 %	9.65 %
16	1.19 %	1.85 %	1.00 %	0.55 %	2.41 %
17	9.39 %	1.31 %	0.75 %	40.74 %	11.26 %
18	5.46 %	41.41 %	1.01 %	47.85 %	4.17 %
19	1.03 %	0.92 %	0.50 %	0.43 %	3.63 %
20	0.86 %	2.76 %	0.50 %	0.32 %	3.46 %
21	0.29 %	1.35 %	0.75 %	82.39 %	8.74 %
22	3.95 %	39.46 %	2.02 %	43.26 %	2.09 %
23	4.81 %	45.94 %	1.28 %	45.98 %	3.81 %
24	1.88 %	1.60 %	0.81 %	51.16 %	16.08 %
25	7.34 %	46.28 %	1.02 %	91.98 %	1.21 %
26	4.56 %	4.19 %	1.27 %	46.33 %	11.07 %
27	1.04 %	4.32 %	1.77 %	7.50 %	6.92 %
28	0.69 %	5.81 %	2.04 %	19.75 %	9.61 %
29	10.58 %	39.07 %	2.05 %	38.20 %	4.40 %
30	14.23 %	94.53 %	2.89 %	21.86 %	23.81 %
31	0.83 %	37.81 %	1.90 %	0.49 %	8.63 %
32	2.64 %	46.41 %	1.11 %	1.13 %	16.09 %
33	0.17 %	53.57 %	0.01 %	49.68 %	7.87 %
34	27.18 %	37.92 %	0.28 %	39.86 %	9.77 %
35	8.35 %	37.45 %	0.00 %	89.96 %	12.70 %
36	7.90 %	72.33 %	0.01 %	11.39 %	13.11 %
37	11.15 %	89.03 %	0.28 %	13.98 %	8.55 %
38	16.18 %	80.11 %	0.55 %	92.99 %	17.07 %
39	13.56 %	6.60 %	0.81 %	22.51 %	12.99 %
40	10.89 %	68.67 %	1.65 %	42.49 %	11.12 %
41	8.02 %	42.55 %	2.18 %	85.60 %	10.56 %
42	3.18 %	3.40 %	0.52 %	0.58 %	3.02 %
43	0.90 %	0.87 %	0.79 %	3.15 %	1.96 %
44	2.33 %	0.51 %	1.34 %	0.86 %	2.21 %
45	0.19 %	0.90 %	1.56 %	0.80 %	3.40 %
46	1.42 %	1.54 %	0.25 %	1.06 %	0.54 %
47	0.71 %	2.89 %	0.52 %	3.03 %	0.56 %
48	1.63 %	3.37 %	1.93 %	0.97 %	0.36 %
49	3.01 %	7.34 %	0.52 %	3.29 %	0.55 %
50	2.47 %	1.85 %	0.80 %	1.29 %	3.70 %
Средно	6.76 %	21.33 %	1.20 %	24.24 %	5.93 %

Табл. 9: Резултати от базов модел "Segment Anything"

Изображение	Долна граница	Лява граница	Шийка	Дясна граница	Горна граница
1	0.61 %	1.17 %	6.51 %	1.10 %	3.10 %
2	2.22 %	3.53 %	1.12 %	3.53 %	0.39 %
3	28.59 %	8.53 %	54.29 %	1.33 %	36.77 %
5	2.41 %	1.97 %	0.74 %	0.74 %	4.70 %
6	1.27 %	2.67 %	1.70 %	4.11 %	3.53 %
7	15.96 %	24.47 %	1.13 %	1.53 %	0.25 %
8	1.16 %	11.59 %	0.70 %	1.60 %	1.64 %
9	3.59 %	1.26 %	0.43 %	0.44 %	4.40 %
10	0.15 %	2.60 %	1.99 %	1.45 %	2.56 %
11	0.64 %	1.07 %	1.68 %	1.18 %	1.85 %
12	2.68 %	0.60 %	0.05 %	1.04 %	3.66 %
13	44.42 %	92.14 %	93.35 %	93.54 %	39.28 %
14	6.13 %	0.85 %	0.25 %	91.54 %	11.62 %
15	1.68 %	2.55 %	0.69 %	4.22 %	1.90 %
16	2.01 %	3.78 %	0.52 %	0.80 %	3.08 %
17	5.90 %	77.65 %	0.01 %	55.10 %	0.13 %
18	0.68 %	1.37 %	1.50 %	2.43 %	0.34 %
19	58.08 %	74.15 %	95.50 %	21.73 %	77.89 %
20	0.69 %	0.80 %	0.51 %	2.99 %	0.34 %
21	46.04 %	157.35 %	83.04 %	15.10 %	14.62 %
22	0.70 %	2.39 %	1.77 %	2.87 %	0.87 %
23	0.50 %	0.62 %	1.02 %	3.41 %	1.37 %
24	0.18 %	1.26 %	0.29 %	3.32 %	0.17 %
25	10.03 %	36.15 %	1.76 %	0.75 %	3.70 %
26	0.33 %	0.82 %	1.05 %	3.31 %	0.34 %
27	0.18 %	2.26 %	1.52 %	0.34 %	2.59 %
28	0.51 %	2.66 %	1.78 %	2.34 %	1.91 %
29	0.53 %	5.12 %	2.05 %	0.25 %	1.92 %
30	8.31 %	0.39 %	2.89 %	28.02 %	6.19 %
31	1.41 %	0.00 %	1.69 %	0.85 %	2.86 %
32	4.15 %	3.07 %	0.28 %	3.44 %	2.05 %
33	8.03 %	5.78 %	1.44 %	11.03 %	0.74 %
34	8.26 %	0.83 %	1.82 %	59.41 %	0.80 %
35	6.99 %	0.11 %	0.01 %	0.32 %	0.01 %
36	7.59 %	0.62 %	0.01 %	8.73 %	1.60 %
37	5.03 %	1.79 %	0.84 %	41.48 %	11.26 %
38	1.06 %	2.24 %	0.17 %	5.32 %	2.89 %
39	8.85 %	32.17 %	1.62 %	85.02 %	4.86 %
40	0.55 %	2.54 %	0.73 %	80.51 %	4.02 %
41	7.82 %	76.13 %	1.08 %	34.78 %	4.10 %
42	6.19 %	3.94 %	0.53 %	2.08 %	0.88 %
43	0.70 %	0.98 %	1.34 %	0.35 %	2.64 %
44	0.35 %	2.27 %	1.61 %	0.73 %	2.92 %
45	2.83 %	1.30 %	2.60 %	1.15 %	6.98 %
46	0.89 %	1.73 %	0.80 %	1.88 %	4.87 %
47	74.35 %	71.09 %	95.79 %	97.09 %	82.44 %
48	2.15 %	3.16 %	0.37 %	7.97 %	2.46 %
49	0.88 %	8.59 %	0.29 %	4.87 %	3.64 %
50	1.40 %	2.47 %	0.01 %	2.14 %	3.51 %
Средно	8.07 %	15.15 %	9.65 %	16.31 %	7.69 %

Табл. 10: Резултати от невронна мрежа "FastSAM"

Изображение	Долна граница	Лява граница	Шийка	Дясна граница	Горна граница
1	2.66 %	1.32 %	0.03 %	0.91 %	0.06 %
2	0.02 %	2.10 %	0.46 %	4.68 %	1.82 %
3	2.31 %	3.15 %	1.57 %	0.51 %	1.04 %
4	1.20 %	1.33 %	2.38 %	3.28 %	2.70 %
5	0.27 %	2.10 %	2.09 %	11.81 %	21.87 %
6	1.04 %	0.82 %	1.70 %	13.44 %	20.12 %
7	1.70 %	1.12 %	1.32 %	2.34 %	1.38 %
8	1.92 %	5.85 %	1.18 %	8.89 %	0.08 %
9	2.52 %	1.13 %	1.94 %	0.99 %	0.53 %
10	3.02 %	0.55 %	2.29 %	2.52 %	1.68 %
11	7.29 %	52.66 %	1.39 %	1.83 %	2.14 %
12	12.64 %	31.14 %	2.96 %	6.31 %	14.74 %
13	3.43 %	1.76 %	0.01 %	11.00 %	8.03 %
14	1.54 %	2.32 %	1.79 %	11.38 %	8.90 %
15	0.16 %	3.54 %	0.51 %	19.17 %	10.04 %
16	0.67 %	0.87 %	1.00 %	15.59 %	7.38 %
17	0.85 %	0.04 %	0.25 %	8.28 %	6.87 %
18	1.54 %	0.71 %	0.51 %	26.50 %	11.88 %
19	0.52 %	0.96 %	0.75 %	13.58 %	8.07 %
20	2.05 %	2.16 %	0.26 %	16.13 %	9.63 %
21	0.00 %	0.87 %	0.29 %	8.60 %	8.10 %
22	1.56 %	1.99 %	1.26 %	20.61 %	11.07 %
23	1.72 %	0.53 %	1.03 %	13.73 %	8.59 %
24	1.71 %	2.98 %	0.76 %	15.52 %	13.81 %
25	3.57 %	1.34 %	0.78 %	19.70 %	5.45 %
26	2.05 %	1.35 %	1.02 %	1.31 %	4.90 %
27	1.89 %	2.98 %	1.27 %	21.85 %	15.25 %
28	1.37 %	0.02 %	1.78 %	23.69 %	15.60 %
29	3.30 %	1.59 %	1.80 %	22.91 %	15.23 %
30	2.13 %	0.64 %	2.10 %	16.32 %	19.61 %
31	1.60 %	1.16 %	1.92 %	2.19 %	10.74 %
32	0.37 %	0.01 %	0.52 %	0.43 %	10.66 %
33	0.74 %	8.52 %	1.73 %	7.85 %	5.74 %
34	3.14 %	1.27 %	0.27 %	15.65 %	9.62 %
35	0.53 %	0.47 %	0.81 %	14.07 %	11.43 %
36	0.17 %	0.09 %	0.83 %	14.53 %	10.72 %
37	3.23 %	0.93 %	0.55 %	15.07 %	11.82 %
38	1.61 %	0.46 %	0.55 %	7.96 %	11.64 %
39	1.79 %	0.96 %	0.27 %	7.31 %	11.24 %
40	1.96 %	2.32 %	0.55 %	7.96 %	11.44 %
41	1.43 %	1.40 %	0.55 %	6.14 %	11.87 %
42	0.36 %	1.09 %	0.84 %	10.30 %	10.03 %
43	0.89 %	0.29 %	1.06 %	11.91 %	9.83 %
44	2.16 %	1.41 %	1.37 %	3.35 %	9.74 %
45	0.89 %	0.86 %	1.04 %	4.53 %	7.01 %
46	1.25 %	0.50 %	0.29 %	6.57 %	9.25 %
47	0.89 %	2.91 %	0.26 %	12.57 %	9.23 %
48	3.06 %	4.41 %	0.77 %	8.85 %	6.26 %
49	1.60 %	7.00 %	0.01 %	4.79 %	10.95 %
50	2.11 %	1.35 %	1.06 %	18.95 %	9.17 %
Средно	1.93 %	3.35 %	1.03 %	10.49 %	9.10 %

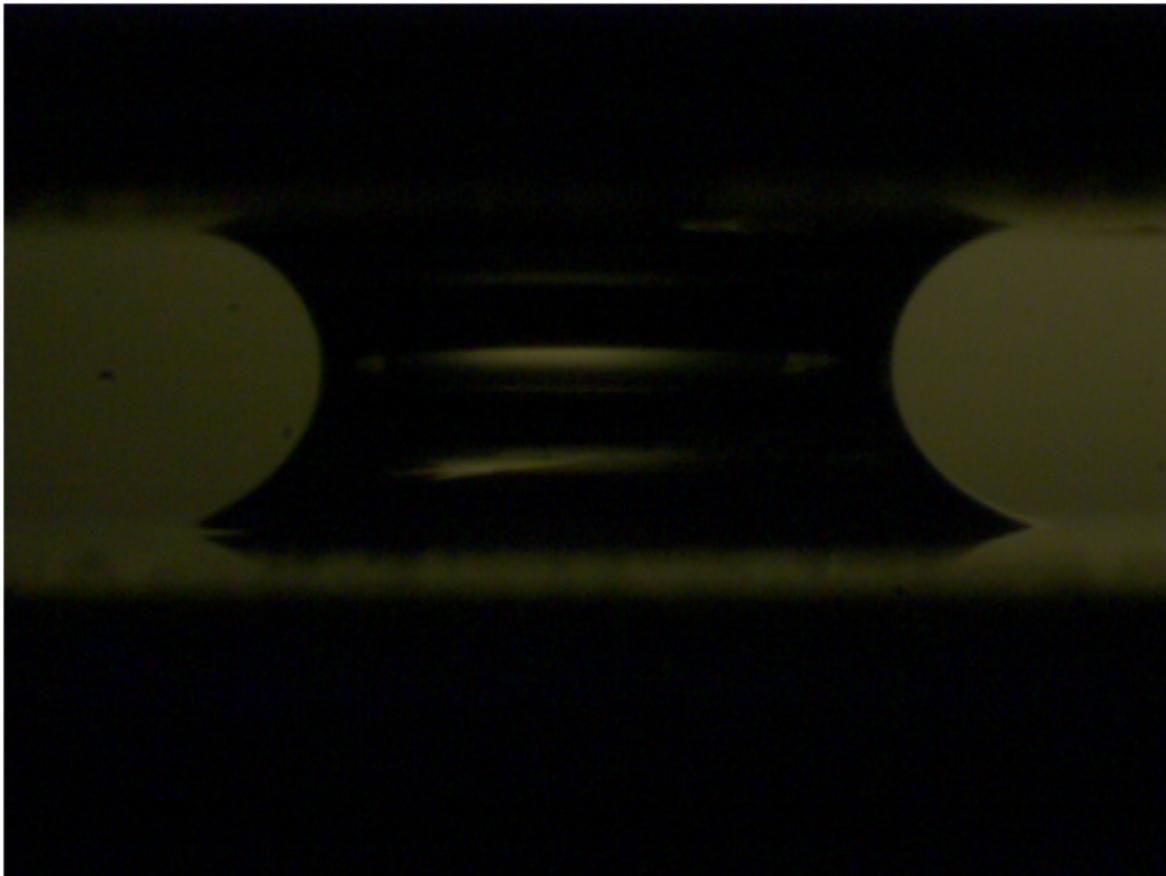
Табл. 11: Резултати от невронна мрежа с фина настройка

6.5 Лимитации на софтуера

Софтуерът се приближава до желаната точност ($\sim 2\%$ грешка спрямо ръчните измервания) за голяма част от изображенията, но не постига достатъчна добра точност за някои от тях.

Основната стъпка от цялостния алгоритъм, която е решаваща за точното измерване на параметрите, е сегментацията на изображенията. Намирането на контурите, изпъкналата обвивка и дефектите на изпъкналост (глава 5), които са нужни, за да се намерят началните и крайните точки на параметрите, до голяма степен зависи от сегментацията. Основната лимитация при сегментацията са нисък интензитет на пикселите в областите от изображението, които съдържат началните и крайните точки на параметрите. Този проблем може да се реши като се увеличи времето за експозиция на камерата или се приложи филтър за увеличаване на контраста. Друга лимитация при сегментацията са отраженията, които са разположени близо до долната или горната граница и интензитетът им е сходен с интензитета на обектите, които трябва да се сегментират. В някои случаи алгоритъмът не може да направи разделянето между левия и десния обект, а в други случаи не може да намери началната и крайната точка на долната или горната граница. Пример за изображение с подобни характеристики, които представляват проблем за алгоритъма е на фиг. 20.

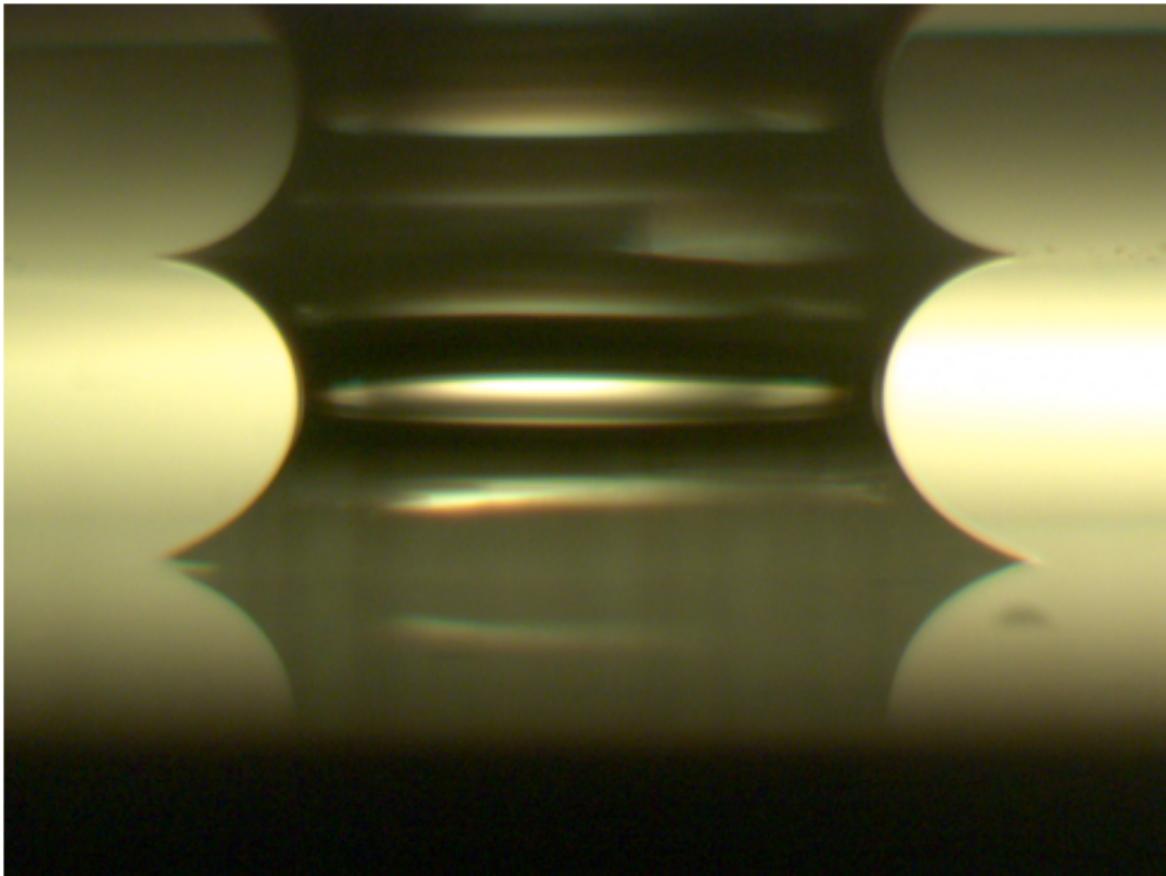
Неподходящо изображение за софтуера



Фиг. 20: Неподходящо изображение за алгоритъма

Алгоритъмът достига желаната точност за изображения, които имат висок контраст между обектите и фона (висок интензитет при обектите и нисък интензитет при фона). Въпреки наличието на множество отражения в центъра на изображението или близо до долните и горните граници, алгоритъмът се справя със сегментацията на обектите, ако отраженията не се припокриват с обектите или се припокриват, но разликата в интензитета между тях и обектите е висока. Пример за изображение с подобни характеристики, което алгоритъмът обработва правилно е на фиг. 21.

Подходящо изображение за софтуера



Фиг. 21: Подходящо изображение за алгоритъма

7 Софтуер

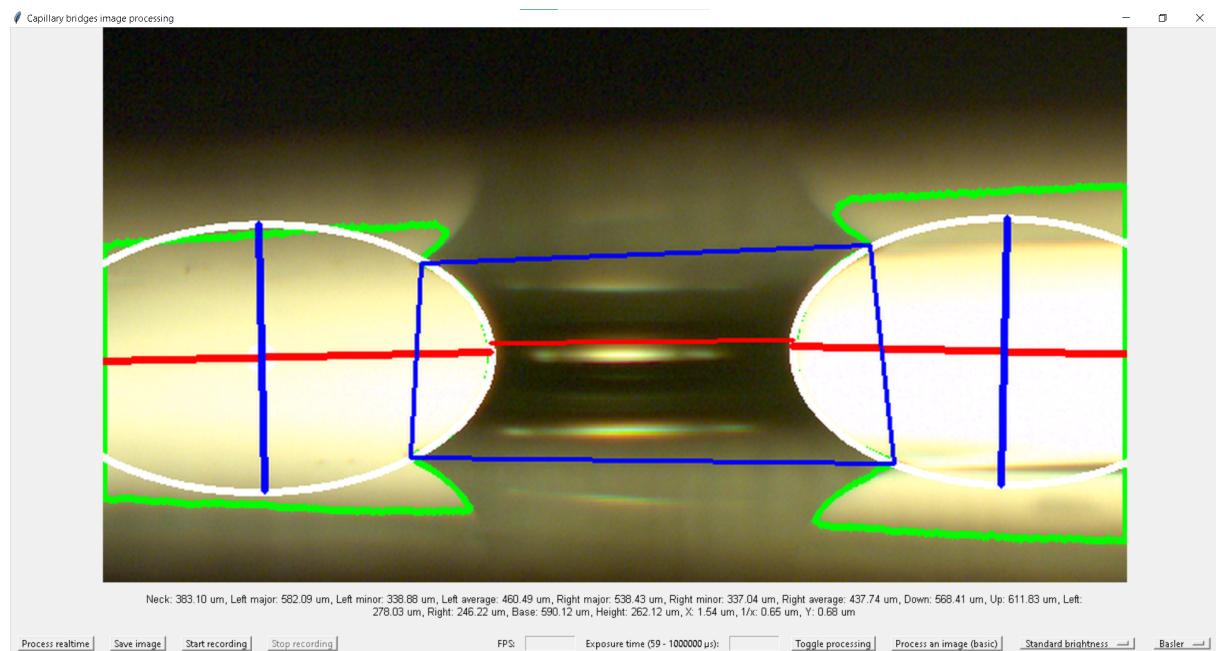
Софтуерът е написан изцяло на Python. Създаден е модул за тестване и сравнение на различни модели за измерване на параметрите, който включва добавянето, премахването и променянето на тестовите изображения върху, които се тестват моделите. Този модул съпоставя измерванията на модела с ръчните измервания, които се въвеждат предварително за всяко изображение. Резултатите от тестването се записват като изображения с измерените параметри, както и самите измервания под формата на таблица. Добавени са функционалности за сравнение на измерените параметри между различните модели. Софтуерът може да работи със стандартни USB камери, както и с Basler камери.

7.1 Използвани библиотеки

Основните библиотеки, които са използвани са "NumPy" [20], "OpenCV" [17], "Matplotlib" [21]. Измерените параметри се записват в Excel файлове чрез библиотеката "openpyxl". За работа с Basler камерата се използва библиотеката "pyylon". За обучаването и използването на невронните мрежи се използва библиотеката "PyTorch" [14]. Създава се изпълним файл чрез библиотеката "PyInstaller", която компилира Python скриптовете и библиотеките, които са нужни на програмата.

7.2 Графичен интерфейс

Графичният интерфейс предоставя по-удобен и интуитивен начин за използване на програмата. Имплементиран е чрез библиотеката Tkinter [22]. Има опции за обработване на видео в реално време, записване на отделни снимки от видеото и резултатите от измерването им в Excel файл, записване на видео, обработване на автоматично разделените кадри от видеото, добавяне на филтър за увеличаване на контраста, промяна на времето за експозиция и скоростта на заснемане. Визуализация на интерфейса може да се види на фиг. 22.



Фиг. 22: Графичен интерфейс на програмата

8 Заключение

Разгледани са и са сравнени два основни модела за сегментация на обектите в изображенията на капилярни мостове - прагова обработка с метод на Оцу и невронна мрежа "Segment Anything". Измерванията след сегментация с невронна мрежа с фина настройка са по-точни за повечето параметри, но този модел не позволява обработване на изображенията в реално време и изисква повече памет, за разлика от сегментация чрез прагова обработка с метод на Оцу. Възможно подобрение на

точността на сегментацията чрез невронна мрежа е обучаване върху повече изображения. Запазването на точността и преодоляването на изискването за повече памет и изчислителни ресурси на невронната мрежа може да се получи като се приложи метода за фина настройка върху по-малък и по-бърз модел с по-малко параметри като "Fast-SAM". Освен моделите за сегментация, като част от дипломната работа, е разработен алгоритъм, който автоматично измерва желаните параметри след като изображението е сегментирано. Разработен е и графичен потребителски интерфейс, чрез който лесно може да се използват моделите и алгоритмите.

Като резултат от дипломната работа е създаден софтуер, чрез който може да се автоматизира ръчното измерване на параметрите на капилярен мостове от експериментални изображения с точност близка до ръчните измервания. Това улеснява работа с експерименталната постановка, помага за контролирането и оптимизирането на експеримента и намалява времето за обработването на данните от експеримента.

Кодът е качен на <https://github.com/petkokp/capillary-bridges-image-analysis>

Използвана литература

- [1] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.
- [2] N. Borisova, P. Petkov, and H. Iliev, “Study of binary liquid capillary bridges stretched between two solid flat surfaces,” 10 2022.
- [3] N. Otsu *et al.*, “A threshold selection method from gray-level histograms,” *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.
- [4] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, “Segment anything,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.
- [5] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, “Fast segment anything,” *arXiv preprint arXiv:2306.12156*, 2023.
- [6] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
- [9] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.
- [10] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, “Learning transferable visual models from natural language supervision,” in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

- [11] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, “Handwritten digit recognition with a back-propagation network,” *Advances in neural information processing systems*, vol. 2, 1989.
- [12] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, “Yolact: Real-time instance segmentation,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9157–9166.
- [13] A. Dutta and A. Zisserman, “The VIA annotation software for images, audio and video,” in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM ’19. New York, NY, USA: ACM, 2019. [Online]. Available: <https://doi.org/10.1145/3343031.3350535>
- [14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in neural information processing systems*, vol. 32, 2019.
- [15] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” *arXiv preprint arXiv:2106.09685*, 2021.
- [16] S. Suzuki, “Topological structural analysis of digitized binary images by border following,” *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.
- [17] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [18] J. Sklansky, “Finding the convex hull of a simple polygon,” *Pattern Recognition Letters*, vol. 1, no. 2, pp. 79–83, 1982.
- [19] A. W. Fitzgibbon, R. B. Fisher *et al.*, *A buyer’s guide to conic fitting*. Citeseer, 1996.
- [20] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: <https://doi.org/10.1038/s41586-020-2649-2>
- [21] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [22] F. Lundh, “An introduction to tkinter,” URL: www.pythonware.com/library/tkinter/introduction/index.htm, 1999.