



УНИВЕРЗИТЕТ У НОВОМ САДУ  
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У  
НОВОМ САДУ

---




Стојан Митрић

# **Софтвер за класификацију објеката на покретној траци**

Дипломски рад  
- Основне академске студије -

Нови Сад, 2016.



	УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ <b>ТЕХНИЧКИХ НАУКА</b> 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	<b>ЗАДАТАК ЗА ДИПЛОМСКИ РАД</b>	Датум:
СТУДИЈСКИ ПРОГРАМ:	Рачунарство и аутоматика	
РУКОВОДИЛАЦ СТУДИЈСКОГ ПРОГРАМА:	Проф. др. Мирослав Поповић	

(Податке уноси предметни наставник - ментор)

Студент:	Стојан Митрић	Број индекса:	РА39/2011
Област:	Софт компјутинг		
Ментор:	Проф. Др. Ђорђе Обрадовић		
НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА: <ul style="list-style-type: none"> <li>- проблем – тема рада;</li> <li>- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;</li> </ul>			

### НАСЛОВ ДИПЛОМСКОГ РАДА:

**Софтвер за класификацију објеката на покретној траци**

### ТЕКСТ ЗАДАТКА:

Специфицирати и имплементирати софтверски систем за препознавање и класификацију објеката на покретној траци. Описати стање у области са посебним освртом на технологије и методе које се користе. Спецификацију извршити UML језиком. Верификацију имплементираног система извршити на изабраним примерима.

Руководилац студијског програма:	Ментор рада:
Примерак за: о - Студента; о - Ментора	

# Садржај

1. Увод .....	5
2. Основни појмови и дефиниције .....	7
2.1. Рачунарска визија.....	7
2.1.1. Претраживање слике на основу садржаја и препознавање објеката.....	7
2.1.2. Представљање слика у рачунару .....	8
2.1.3. <i>RGB</i> модел боја .....	8
2.1.4. <i>HSV</i> модел боја.....	9
2.1.5. Избор модела боја.....	11
2.1.6. Сегментација слике.....	11
2.2. Математичка морфологија .....	12
2.2.1. Логичке операције .....	12
2.2.2. Морфолошке операције.....	13
2.2.2.1. Операција отварање .....	13
2.2.2.2. Операција затварање .....	14
2.2.2.3. Ерозија .....	15
2.2.2.4. Дилатација.....	15
2.3. Гринова теорема .....	15
2.4. Класификација, фази логика, фази скупови .....	17
3. Спецификација и имплементација софтвера за препознавање и класификацију објеката.....	19
3.1. Модел за класификацију објеката .....	19
3.2. <i>OpenCV</i> .....	21
3.2.1. Добијање слике .....	21
3.2.2. Издвајање објеката из позадине .....	22
3.2.2.1. Конверзија слике из <i>RGB</i> у <i>HSV</i> модел боја .....	22
3.2.2.2. Проналажење контура (детекција ивица).....	23
3.2.2.3. <i>Moments</i> метода.....	23
3.2.3. Одређивање узорака, калибрација .....	25
3.2.3.1. Калибрација боја.....	26
3.2.3.2. Калибрација бројања објеката.....	27
3.2.4. Исцртавање облика на видеу .....	28
3.2.5. Снимање видеа.....	30
3.2.5.1. Аутоматски режим снимања видеа.....	30
3.2.5.2. Мануелни режим снимања видеа.....	31
3.3. Вођење евиденције о објектима.....	31
3.4. Команде.....	31
4. Верификација.....	31
4.1. Симулација траке .....	31
4.2. Повезивање рачунара и камере телефона .....	33
4.3. Специјални случај код бројања.....	33
5. Закључак.....	36
Литература .....	37
Биографија.....	39
Кључна документацијска информација.....	40
<i>Key words documentation</i> .....	41

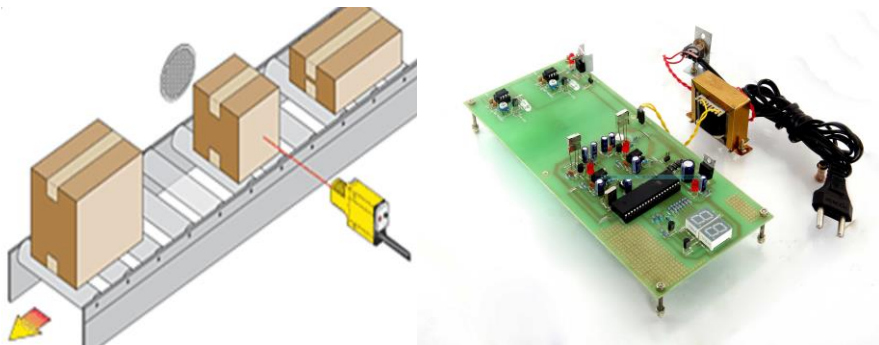
# 1. Увод

Дигитална слика као нумеричка репрезентација дводимензионалне слике је представљена коначним скупом елемената који се називају пиксели. Пиксели су дигиталне вредности који носе одређене информације. Зависно од резолуције слике, број пиксела може бити велики, а самим тим повећава се и неодређеност информација које носе. Класификација и препознавање објеката, који представљају скупове пиксела, у области рачунарске визије подразумева коришћење техника обраде дигиталних слика помоћу рачунара ради издвајања података који су смештени у пикселима. Примена препознавања објеката на слици је вишеструка, користи се у индустрији и разним наукама попут биологије, медицине, хемије, физике. Један од врло занимљивих подручја примене препознавања објеката је примена у роботизици, односно у системима који могу потпуно аутономно радити у простору прилагођеном људима или сарађивати са људима и олакшати им рад. Ови системи могу потпуно заменити људске ресурсе на пословима који могу бити аутоматизовани и са много више прецизности и ефикасности обављани од стране робота. У овом раду ће бити специфициран модел и имплементиран софтвер једног таквог система, који ће препознавати објекте и вршити њихову класификацију. Биће описано стање у области препознавања објеката и могућности употребе већ постојећих метода за препознавање објеката и класификацију. Модел ће бити представљен UML дијаграмом где ће бити наведени атрибути и њихове методе потребне за имплементацију софтвера. Биће представљене методе за тестирање софтвера и верификацију његовог рада, као и проблеми који се могу јавити и могућности даљег развоја софтвера и области примене софтвера.

За препознавање и класификацију објеката постоје развијени различити системи. Неки су модернији, неки застарели али се користе и једни и други, јер имају своје предности и мане.

## **Хардверски системи:**

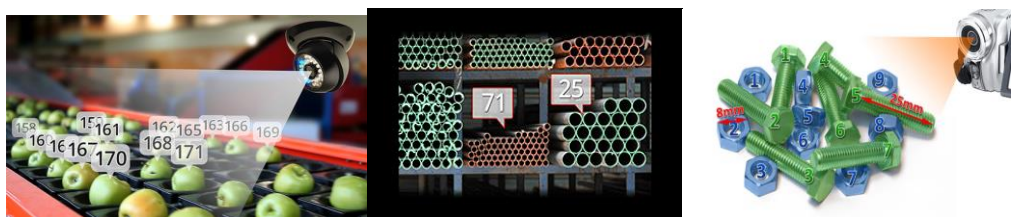
Помоћу ласера који емитују светлосни сноп и сензора који су осетљиви на њих могуће је пребројати објекте на траци. Објекат наилази на светлосни сноп и прекида његов пут до сензора, на основу тога објекат се детектује и броји. Критеријум за класификацију објекта овом методом може бити његова величина, односно временски период током којег је светлосни сноп прекинут, што може бити недовољан критеријум за препознавање. Хардверска решења могу бити скупа и ограничавајућа што се тиче простора, домета ласера и организације целог система.(Слика 1.).



**Слика 1.** Бројање објеката помоћу хардвера

### **Софтверски системи:**

Базирају се на примени компјутерске визије, конкретно обради и анализи слике. Овим методама могуће је добити пазличите информације о објектима као што су димензије, облик, боја. Могуће је упоређивати сличне објекте и донети закључке који су исправни а који нису. Предност је што се софтверска решења могу лако повезати са другим хардверским и софтверским решењима и послати им информације које директно могу утицати на њихов рад. У прехранбеној индустрији где се на покретној траци крећу намирнице, уколико нека од намирница има неки дефект, систем за препознавање може послати систему за раздвајање производа информације о позицији на којој се дефектни производ налази како би га систем одстранио. Све што је потребно за имплементацију од хардверских компоненти су камера и рачунар и начин за њихово умрежавање. Предност је што камера и рачунар могу бити бесконачно удаљени и да систем функционише. (Слика 2.).



**Слика 2.** Бројање и класификација објеката помоћу софтвера

## **2. Основни појмови и дефиниције**

### **2.1. Рачунарска визија**

Рачунарска визија је једна од области коју проучава вештачка интелигенција која комбинује рачунарску интелигенцију са дигитализованом визуелном информацијом. Настоји да анализом, процесирањем и обрадом слика из реалног света извуче нумеричке или симболичке информације које се могу искористити у различитим научним истраживањима. Развој рачунарске визије потиче из жеље за дупликацијом људског вида помоћу електронског сагледавања слике. Са развојем модерних рачунарских система и могућности брзе и ефикасне обраде података, развила се потреба за имплементацијом решења за интерпретирање слика на рачунару а потом за њиховим разумевањем и анализом. Рачунарска визија обухвата методе за прикупљање, обраду, анализу и разумевање слике. Основни циљ је анализа високо-димензионих података из стварног света зарад добијања нумеричке или симболичке информације разумљиве машини. Велики изазов представља сама рачунарска представа природних величина попут боје, тоналитета и слично, па је тема у развоју ове области унапређивање могућности људске визије електронским сагледавањем и поимањем слике. Ова слика компјутерског разумевања (визије) може се посматрати као размршавање симболичке информације из сликовних података, помоћу модела изграђених уз помоћ геометрије, физике, статистике и слично. Рачунарска перцепција слике омогућава брзу, аутоматизовану и у неким случајевима поузданију анализу слике од људског ока. Примећивање детаља и препознавање облика је област у којој рачунари заостају за људским оком, док је задатак пребројавања објеката боље препустити програмираној машини. Као научна дисциплина, рачунарска визија се бави теоријом која стоји иза вештачких система који преузимају информације из слика. Врсте података из којих се узимају подаци поред слике могу бити - видео секвенце, прикази са више камера или мулти-димензиони подаци са медицинских скенера. Ова област уско је повезана са другим областима из обраде података, па се у истој подразумева примена метода дигиталне обраде сигнала, препознавања облика, спектралне анализе итд. [5][4][1]

#### **2.1.1. Претраживање слике на основу садржаја и препознавање објеката**

Претраживање слика на основу садржаја представља широку подобласт компјутерске визије. Бави се проблемом проналажења садржаја на слици сличних унапред одређеном узорку. За формулисање сличности ће се користити

искључиво садржај слике (боја, облици, текстуре). У пољу рачунарске визије препознавање објеката представља задатак проналажења и идентификације објекта на слици. Људи могу препознати и класификовати мноштво објеката на слици са мало труда, упркос чињеници да представа објекта на слици може да варира на различите начине. На слици објекти могу бити разне величине, транслирани или ротирани итд. Људи могу да идентификују или наслуте који објекат је у питању чак иако фали велики део тог објекта или ако се неки делови објекта не виде. Овај задатак је одувек био изазов за системе компјутерске визије. Постоји више техника за поређење садржаја и формирање критеријума препознавања (препознавање по боји, облику, текстури). Свака од техника има своје предности и мане. Зависно од услова под којима је потребно направити систем за препознавање и класификацију потребно је одабрати адекватну методу препознавања.[4][6]

У овом раду ће бити објашњене методе за препознавање објеката на основу боје.

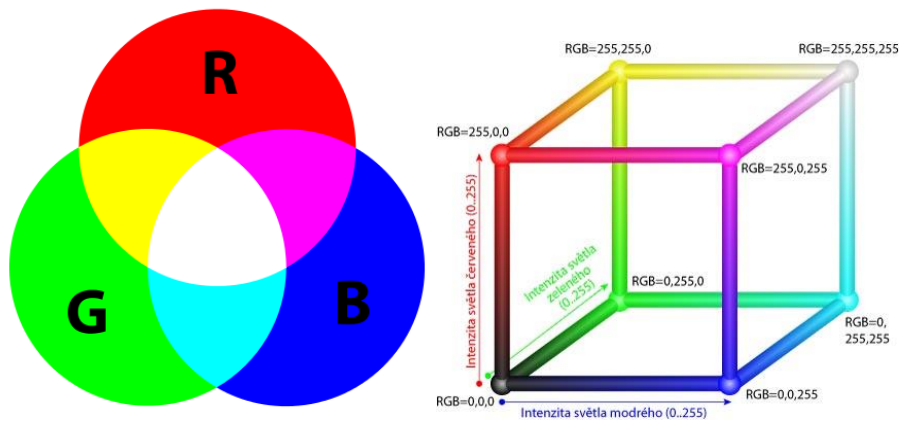
## 2.1.2. Представљање слика у рачунару

Основни проблем компјутерске визије представља сама репрезентација слике у рачунару. Начина на који ћемо на рачунару интерпретирати слику може бити много и они зависе од примене, а како се област обраде слике развија паралелно са првим рачунарским платформама метода за представљање слика има много.

## 2.1.3. *RGB* модел боја

Постоји више модела боја од којих су најзначајнији *RGB* (*Red Green Blue*), *HSV* (*Hue Saturation Value*), *CMYK* (*cyan, magenta, yellow and black*). *RGB* модел боја је адитивни модел у коме се мешају светло црвене зелене и плаве боје да би се добио одређени спектар. Мешањем светла ових боја могу се добити све остале боје (Слика 3.) Главна сврха овог модела боја је репрезентација слика из реалног времена у електронским системима попут телевизора или рачунара. Свака боја се описује кроз збир три вредности које се мешају додавањем на црну боју: део црвеног светла, део зеленог светла и део плавог светла. Сваки део варира између 0% и 100% те боје, а то се на рачунарима представља вредностима између 0 и 255 (један бајт). Мешањем 100% све три боје се добија бела боја, а са 0% сваке од боја се добија црна боја. Када је интензитет сваке од ове три боје једнак добија се сива боја, зависно од интензитета сваке боје тамнија или светлија. У анализи слике и праћењу објеката на основу боје се много више користи *HSV* модел боја јер је много једноставније издвојити боју, из разлога што мешање боја према *RGB* моделу није интуитивно човековом виду и схватању боја. Тешко је претпоставити која ће се боја добити. [8][4]





Слика 3. RGB модел боја

#### 2.1.4. HSV модел боја

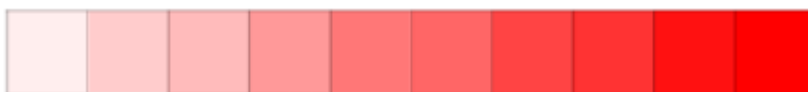
*HSV* модел боја представља трансформацију *RGB* модела који је интуитивнији човеку. Овај модел боја садржи три компоненте: *Hue* – нијанса(тон) боје, *Saturation* (засићеност боје) и *Value* (вредност/сјај боје).

*Hue* – атрибут визуелног осећаја по којем је нека област слична некој од боја (црвеној, жутој, зеленој или плавој) или некој од њихових комбинација. Одређена је таласном дужином светла.

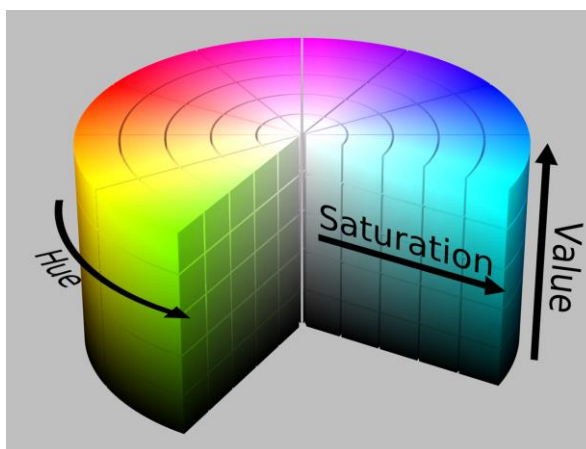
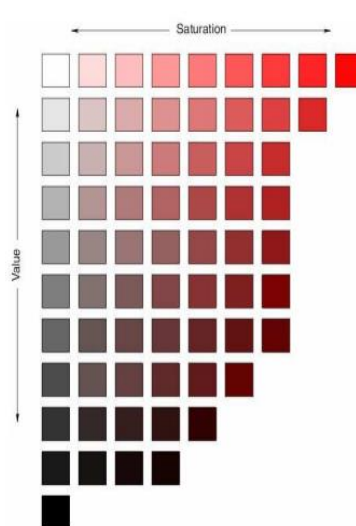
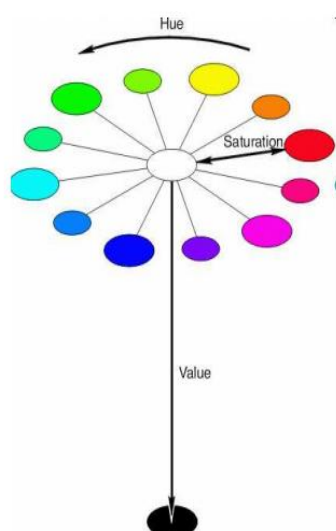
*Saturation* – атрибут визуелног осећаја чистоће боје (засићена = чиста, живахна).

*Value/Brightness* – вредност/сјај је атрибут визуелног осећаја по којем нека област изгледа да емитује више или мање светлости, односно субјективан интензитет светла. [9][6]

Засићење[0-100] % за вредност/сјај = 100%



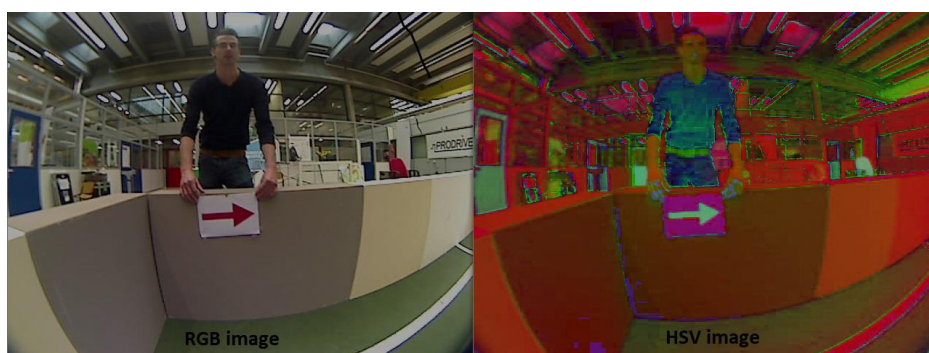
Вредност/сјај[0-100]% за засићење = 100%



Слика 4. Однос *Hue/Saturation/Value*

### 2.1.5. Избор модела боја

*RGB* модел није перцепцијски униформан по питању осветљености да би се постигла добра представа боје. Разлог за то је што су *R, G, B* компоненте боје неког објекта на дигиталној слици у колерацији са интезитетом светлости којом је слика обасјана, односно нису сви делови слике униформно обасјани истом количином светлости. Код *HSV* модела количина светлости је униформна на целој слици, стога је лакше раздвојити боје а самим тим и објекте, па се због тога у издвајању објеката на слици више користи *HSV* модел боја.[4][3][8]



Слика 5. *RGB* модел боја и *HSV* модел боја

### 2.1.6. Сегментација слике

Термин сегментација слике се односи на групу поступака за поделу слике на регионе са сличним атрибутима. Од атрибута се најчешће користи осветљеност - код монохроматских слика (слика у једној боји), или боја - код слика у боји. У процесу сегментације могу се користити и друга обележја: ивице, мере текстуре, итд. Иако је сегментација најважнија фаза у анализи слике, до данас не постоји теоријска основа сегментације. Већина поступака сегментације који су прихваћени у пракси је хеуристичког карактера. Осим тога, не постоји ни начин за квантитативну процену колико је поступак сегментације добар. Најчешће се користи описна квалитативна дефиниција добре сегментације у којој се каже да региони добијени поступком сегментације треба да буду униформни и хомогени у односу на неке карактеристике, као што су ниво сивог (тон) или текстура. Унутрашњост региона треба да буде једноставна и без малих рупа, а границе региона једноставне, глатке и просторно тачне. Суседни региони треба да буду довољно различити по оном атрибуту по коме су униформни. Неопходно је поседовати могућност раздвајања региона на слици који одговарају објектима од позадине. У најједноставнијој имплементацији сегментације(трешхолдовање) добија се бинарна(трешхолдована) слика, која задржава све важне информације у погледу

броја, облика и позиције објекта, а која је касније погодна за анализу и примену морфолошких операција. Обично, црни пиксели (вредност 0) представљају позадину а бели пиксели (вредност 1) представљају објекте. Сегментација је уствари одређивање параметра, који ће представљати праг за класификацију пиксела слике на објекте и позадину. Сви пиксели чија је вредност испод прага придружују се позадини, а они чија је вредност већа или једнака вредности прага придружују се објектима. Сегментирати је могуће сиве слике и мултиспектралне слике(слике у боји). За слике у боји се одређују прагови за сваку компоненту боје, тако да резултујући праг дели простор могућих комбинација боја (рецимо *HSV* простор) у два простора. Одређени алгоритми за сегментацију слика могу дати задовољавајуће резултате само на одређеним класама слика.[3][2]

## 2.2. Математичка морфологија

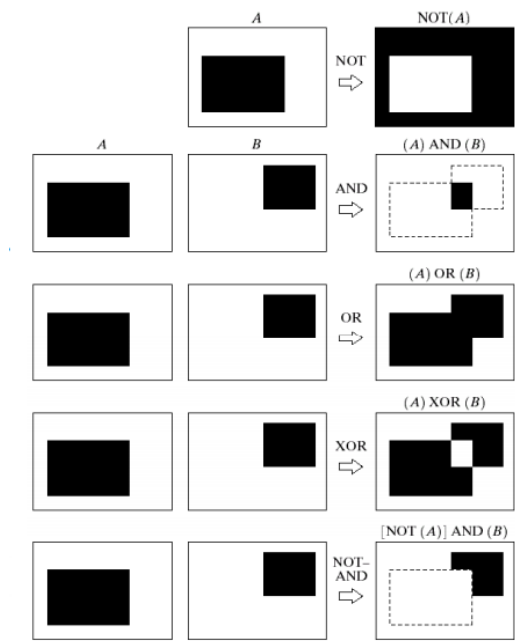
Математичка морфологија је метода издвајања компонената са слике у циљу описивања и репрезентације региона слике. Теорија скупова је основ математичке морфологије. Скупови представљају објекте у слици. У бинарној слици скупови су подскупови простора  $Z_2$ , а чланови су  $2D$  вектори са координатама  $(x,y)$ . [2]

### 2.2.1. Логичке операције

Дефинисане су између пиксела бинарних слика. Омогућавају реализацију морфолошких операција. Три основне логичке операције су *AND*, *OR*, *NOT*. Било која логичка операција може се реализовати помоћу ове три операције.[2]

$p$	$q$	$p \text{ AND } q$ (also $p \cdot q$ )	$p \text{ OR } q$ (also $p + q$ )	$\text{NOT } (p)$ (also $\bar{p}$ )
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Табела 1. Логичке операције



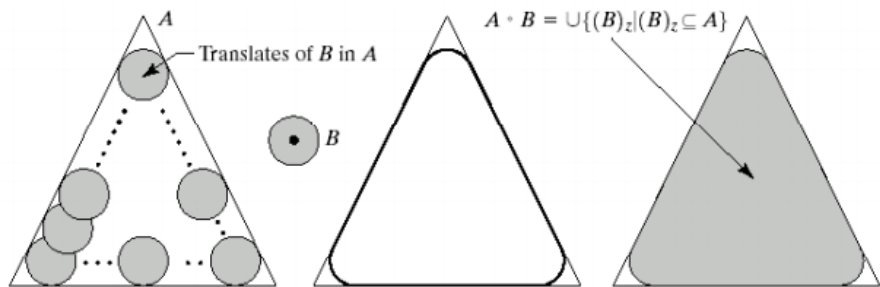
Слика 6. Логичке операције на бинарној слици

## 2.2.2. Морфолошке операције

Реч морфологија значи форма и структура објекта. Морфологијом се описује распоред и дају међусобне релације између објеката. Дигитална морфологија даје начин да се опишу или анализирају облици дигиталних објеката, па према томе и дигиталних слика. Дигитална морфологија посматра слику у контексту теорије скупова. Дакле, слика је скуп елемената (пиксела) који груписани у дводимензионалну структуру дају одређене облике. Математичке операције над скуповима се користе за анализу облика, пребројавање, препознавање и слично. Основне морфолошке операције су ерозија која са слике брише групе пиксела које одговарају задатом и дилатација која мало подручје око пиксела попуњава по задатом моделу. Зависно од типа слике (бинарна, *grayscale* - нијансе сиве, *color* - у боји), дефиниције дилатације и ерозије се разликују и морају се разматрати одвојено.[2][3]

### 2.2.2.1. Операција отварање

Ублажава контуру објекта, укида танке везе између делова објекта и елиминише мале оштре делове објекта. Отварање кружним структурним елементом представља скуп тачака које додирује елемент уписан у објекат.[2]



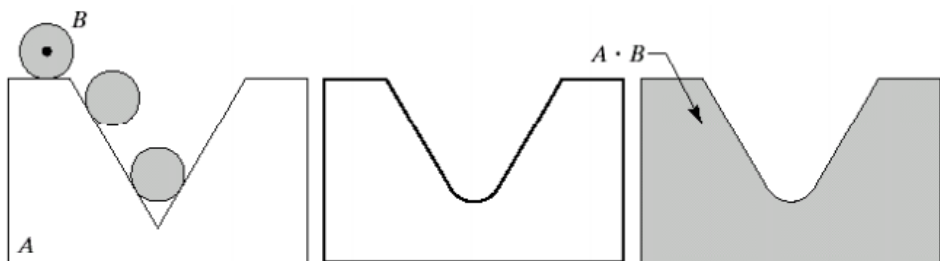
Слика 7. Операција отварања

Особине отварања:

1.  $(A \circ B) \subseteq A$ ,
2.  $C \subseteq D \Rightarrow (C \circ B) \subseteq (D \circ B)$
3.  $(A \circ B) \circ B = A \circ B$

#### 2.2.2.2. Операција затварања

Ублажава контуру објекта, стапа уске прекиде и дуге танке увале, елиминише мале рупе и попуњава процепе у контури. Затварање кружним структурним елементом представља скуп тачака које додирује елемент када клизи око објекта.[2]



Слика 8. Операција затварања

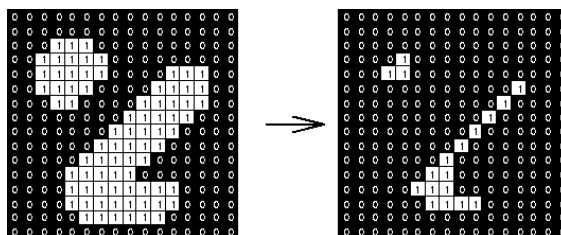
Особине затварања:

1.  $A \subseteq (A \bullet B)$ ,
2.  $C \subseteq D \Rightarrow (C \bullet B) \subseteq (D \bullet B)$
3.  $(A \bullet B) \bullet B = A \bullet B$

У неким сличајевима у бинарној слици може да се јави шум у виду црних пиксела унутар белих површина које представљају издвојене објекте или белих пиксела унутар црних површина које представљају позадину. Када до тога дође, рачунару објекти постају нејасни због чега треба уклонити шум. То се може решити морфолошким операцијама ерозијом и дилатацијом. [3][1]

### 2.2.2.3. Ерозија

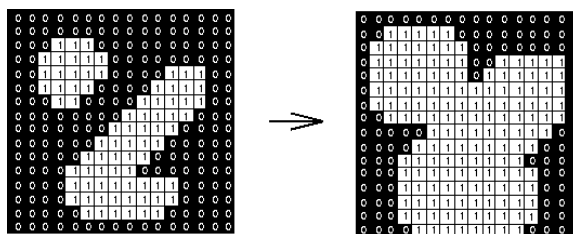
Ерозија у морфолошкој обради слике је отварајућа логичка операција  $AND$  између елемената околине, односно ако су све тачке из околине (црна или бела) тачка ће бити (црна или бела).[1]



Слика 9. Ерозија

### 2.2.2.4. Дилатација

Дилатација у морфолошкој обради слике је затварајућа логичка операција између елемената околине, односно довољно је да бар једна тачка из околине буде (црна или бела) па да тачка буде (црна или бела).[1]



Слика 10. Дилатација

## 2.3. Гринова теорема

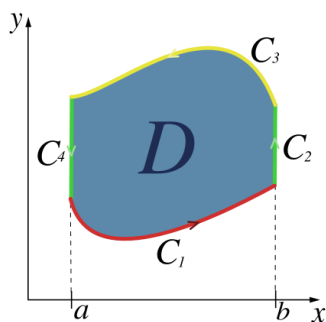
Након добијања бинарне слике и издвајања објекта из позадине, потребно је рачунару објаснити где се дводимензионални објекат на слици налази и које су његове границе. У ове сврхе погодно је применити Гринову теорему. У физици и математици, Гринова теорема даје однос између криволинијског интеграла око просте затворене криве  $C$  и двоструког интеграла над области  $D$  ограниченом са  $C$ . Добила је име по британском научнику Џорџу Грину. Ако је  $C$  позитивно оријентисана, део по део глатка, проста затворена

крива у равни и ако је  $D$  област ограничена кривом  $C$ . Ако  $L$  и  $M$  имају непрекидне парцијалне изводе на отвореној области која садржи  $D$ , онда

$$\int_C L dx + M dy = \iint_D \left( \frac{\partial M}{\partial x} - \frac{\partial L}{\partial y} \right) dA$$

Некада се црта кружић на симболу за интеграл ( $\oint_C$ ) да се означи да је крива  $C$  затворена (тада се интеграл назива циркулацијом). За позитивну оријентацију, на овом кругу се може нацртати стрелица у смеру супротног смеру казаљке на сату.

Ако је  $D$  проста област чије се границе састоје од кривих  $C_1, C_2, C_3, C_4$ , може се демонстрирати Гринова теорема. (Слика 11.).



**Слика 11.** Гринова теорема над области  $D$

Следи доказ теореме за поједностављену област  $D$ , област типа I где су  $C_2$  и  $C_4$  вертикалне линије. Сличан доказ постоји када је  $D$  област типа II, где су  $C_1$  и  $C_3$  праве линије.

Ако се може показати да су искази

$$\int_C L dx = \iint_D \left( -\frac{\partial L}{\partial y} \right) dA \quad (1)$$

и

$$\int_C M dy = \iint_D \left( \frac{\partial M}{\partial x} \right) dA \quad (2)$$

тачни, онда се може доказати Гринова теорема у првом случају.

Област типа I,  $D$  на Слици 11. дефинисана са:

$$D = \{(x, y) | a \leq x \leq b, g_1(x) \leq y \leq g_2(x)\}$$

где су  $g_1$  и  $g_2$  непрекидне функције. Израчуна се двоструки интеграл из (1):



$$\begin{aligned}
& \iint_D \left( \frac{\partial L}{\partial y} \right) dA \\
&= \int_a^b \int_{g_1(x)}^{g_2(x)} \left[ \frac{\partial L}{\partial y}(x, y) dy dx \right] \\
&= \int_a^b \left\{ L[x, g_2(x)] - L[x, g_1(x)] \right\} dx \quad (3)
\end{aligned}$$

$C$  се може записати као унија четири криве:  $C_1, C_2, C_3, C_4$ .

Код  $C_1$ , користе се параметарске једначине:  $x = x, y = g_1(x), a \leq x \leq b$ . Тада

$$\int_{C_1} L(x, y) dx = \int_a^b \left\{ L[x, g_1(x)] \right\} dx$$

Код  $C_3$ , користе се параметарске једначине:  $x = x, y = g_2(x), a \leq x \leq b$ . Тада

$$\int_{C_3} L(x, y) dx = - \int_{-C_3} L(x, y) dx = - \int_a^b [L(x, g_2(x))] dx$$

Интеграл над  $C_3$  се негира, јер иде у негативном правцу од  $b$  до  $a$ , јер је  $C$  оријентисана позитивно (у смеру супротном смеру казаљке на сату). На  $C_2$  и  $C_4$ ,  $x$  остаје константно, што значи да

$$\int_{C_4} L(x, y) dx = \int_{C_2} L(x, y) dx = 0$$

Стога,

$$\begin{aligned}
\int_C L dx &= \int_{C_1} L(x, y) dx + \int_{C_2} L(x, y) dx + \int_{C_3} L(x, y) dx + \int_{C_4} L(x, y) dx \\
&= - \int_a^b [L(x, g_2(x))] dx + \int_a^b [L(x, g_1(x))] dx \quad (4)
\end{aligned}$$

Комбиновањем (3) са (4), добија се (1). На сличан начин добија се (2). [7]

## 2.4. Класификација, фази логика, фази скупови

Развојем рачунарства, а самим тим и развојем система који се користе за решавање проблема у реалном свету, већ постојећи математички модели нису више били довољни да се у потпуности опишу непрецизне информације и вишезначни појмови који су бивали све комплекснији. Проблеми из реалног живота које је постало могуће решити су постајали све компликованији, те традиционални системи за моделовање и технике за анализу су постали сувише прецизни, крути. Да би се смањила комплексност таквих проблема почеле су да се уводе разна упрошћења, претпоставке и различите врсте ограничења. На тај

начин се обезбеђивао задовољавајући компромис између количине информација које имамо са једне стране и њихове несигурности са друге стране.

Овај компромис је постигнут увођењем фази логики и фази скупова. Фази логика се може поделити на :

**-оштру логику**, где пропозиције могу бити или тачне или нетачне, односно степен тачности је или 0 или 1. У оштру логику се убрајају класични или оштри скупови, којима елемент може да припада или не припада. У овом случају функција припадности је једна бинарна функција:

$$\mu_A(x) = \begin{cases} 0 & x \notin A \\ 1 & x \in A \end{cases}$$

У обради и процесирању слике увек се тежи да се неодређеност сведе на минимум, из тог разлога слику из реалног света трешхолдовањем сводимо на бинарну.

**-фази логику**, где је степен тачности у интервалу између 0 и 1. Теорија фази скупова проширује концепт припадности дефинисањем парцијалне припадности. Ако је  $U$  колекција објеката  $x$ , тада је фази скуп  $A$  над  $U$  одређен као скуп парова (објекат, функција припадности) где функција припадности узима реалне вредности из интервала  $[0, 1]$ . Логично је да веће вредности функције означавају већи степен припадности. Из тога се може закључити да фази скуп садржи елементе са различитим степенима припадности том скупу. Та идеја је у супротности са идејом припадности у класичним скуповима, јер тамо елемент није могао припадати скупу ако његова припадност том скупу није потпуна. Треба нагласити да елементи фази скупова могу бити елементи не само једног, већ и више фази скупова одједном. Фази логика за класификацију објеката на слици може да се употреби тако што се направе узорни скупови дефинисањем функције припадности и сваки објекат од интереса ће у одређеној мери припадати неком од узорних скупова. [11][24][1]

$$A = \left\{ (x, \underbrace{\mu_A(x)}_{\text{функција припадности}}) \mid x \in U \right\}$$

$$\mu_A : U \rightarrow [0, 1]$$

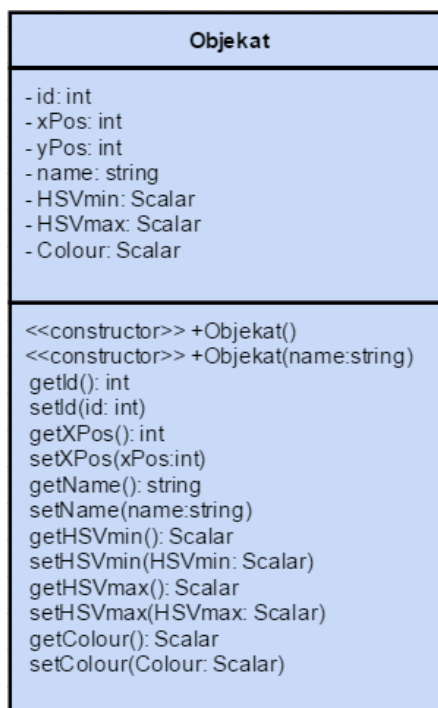
$U$  : универзални скуп

У обради слике након издвајања објеката (сегментације) потребно је одредити којим скуповима објекти припадају и на тај начин се врши њихова класификација.

### 3. Спецификација и имплементација софтвера за препознавање и класификацију објекта

#### 3.1. Модел за класификацију објекта

Модел за класификацију објекта репрезентује објекат који се прати и евидентира. Модел треба да садржи атрибуте објекта и методе које ће се користити у препознавању и класификацији. Обзиром на то да софтвер треба да врши класификацију на основу боје један од атрибута ће сигурно бити боја тог објекта. Пожељно је да сваки објекат носи неко јединствено обележје и да има одређени назив који представља класу којој припада, односно боју којој припада. Објекат се налази на слици која је дводимензионална па би један од атрибута могла бити позиција објекта у односу на координатни почетак.



Слика 14. UML дијаграм класе Објекат

Сваки објекат је описан атрибутима:

*id* – идентификациони број објекта.

*xPos* – x позиција објекта у односу на кадар

*yPos* – y позиција објекта у односу на кадар

*type* – тип (назив објекта)

*HSVmin* – минимална *HSV* вредност под којом је објекат издвојен из позадине у бинарној(трешхолдованој слици), вектор/листа која садржи 3 вредности(*H, S, V*)  
*HSVmax* – максимална *HSV* вредност под којом је објекат издвојен из позадине у бинарној(трешхолдованој слици), вектор/листа која садржи 3 вредности(*H, S, V*)  
*Colour* – боја објекта, вектор/листа која садржи 3 вредности(*R, G, B*)

Објекти који се налазе на покретној траци су произвољно распоређени. Због тога је цео систем неодређен и непредвидив у великој мери. Обзиром на то да би софтвер требао динамички и у реалном времену да обрађује видео, односно секвенцу слика и даје резултате класификације и праћења објеката кориснику у исто време задатак постаје знатно компликованији. Корисник би требао да има увид у све шта се дешава на покретној траци док она ради, односно док се креће. Софтвер треба да обезбеди кориснику увид у тренутно бројчано стање објеката сличне боје који се налазе у кадру камере, обрађује добијени видео у реалном времену и прати их у виду виртуелне (аугментоване) реалности исписујући њихову позицију и назив који их прати све време док су у кадру. Софтвер треба да води евиденцију о укупном бројчаном стању објеката који су прошли кроз кадар као и позицију у односу на кадар и даје јединствен назив сваком објекту. Податке о објектима који пролазе кроз кадар софтвер смешта у текстуални фајл. Корисник треба да има потпуну контролу над креирањем видео снимка и то у режимима :

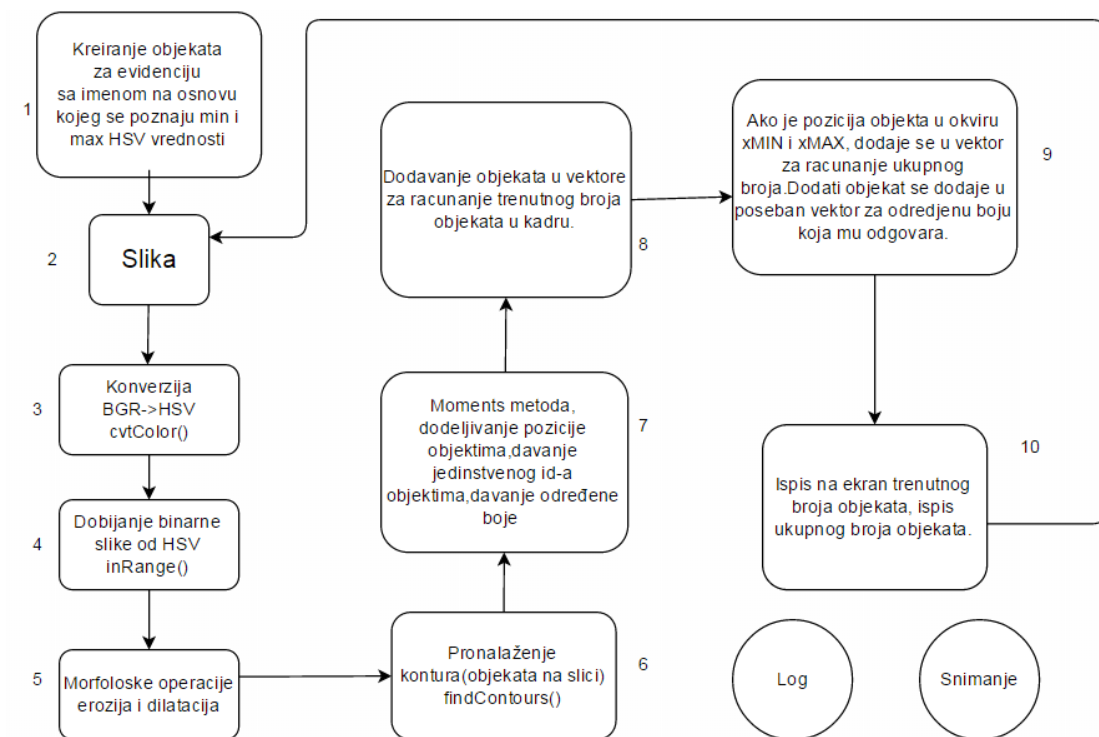
Мануелно – по жељи корисник прави видео снимак онога што се дешава на покретној траци у различитим временским периодима.

Аутоматски – снима се када се објекти налазе у кадру

Комбиновано – корисник може да укључи аутоматско снимање, и у сваком моменту да пређе на мануелно и обрнуто.

Видео се снима све време док је укључен или појединачан режим снимања или комбинован режим снимања, све док корисник не изда команду за снимање новог видеа.

Софтвер треба кориснику давати увид у бројчано стање објеката у тренутном кадру које је ажурно у смислу да сваки пут када нови објекат уђе у кадар или када објекат изађе из кадра(фрејма) бројчано стање се мења, као и укупно бројчано стање свих предмета који су прошли кроз кадар. На Слици 15. приказан је дијаграм рада софтвера.



Слика 15. Дијаграм рада софтвера

## 3.2. OpenCV

Софтвер за класификацију и праћење објеката на покретној траци се може имплементирати у различитим програмским језицима, који подржавају рад са *OpenCV* библиотekom. *OpenCV* (*Open Source Computer Vision*) је библиотека функција које се користе у анализи и обради слике. *OpenCV* библиотека је подржана у великом броју програмских језика на различитим платформама и оперативним системима због чега се често користи у имплементацији софтвера у којима је потребна употреба рачунарске визије.

### 3.2.1. Добијање слике

Слика се може добити помоћу камере или неког другог мултимедијалног уређаја који има својство камере. Сliku је потребно репрезентовати рачунару, ради њене даље обраде. *OpenCV* има уграђене методе за учитавање слике у оквиру класе *VideoCapture*. Да би слика могла да се обради и сачува, односно да би се над њом примењивале методе *OpenCV* библиотеке пракса је да се слика стави у матрицу. Најбоље би било направити три различите матрице, једну за *RGB* слику, једну за *HSV* слику, једну за бинарну слику.

### 3.2.2. Издвајање објекта из позадине

Као што је раније речено, слика добијена са камере на којој се објекат налази је најчешће репрезентована рачунару у *RGB* моделу боја. Потребно је употребити одређене методе које ће слику упростити тако што ће слика бити конвертована у бинарну слику на којој је објекат издвојен из позадине.

#### 3.2.2.1. Конверзија слике из *RGB* у *HSV* модел боја

Конверзијом у *HSV* модел се постиже униформност у погледу осветљења слике. Слика из *RGB* модела се може конвертовати у *HSV* модел помоћу методе *cvtColor()* *OpenCV*-а.

Метода прима три параметра. Први параметар је улазни низ односно матрица у коју је стављена *RGB* слика. Други параметар је излазни низ односно матрица у коју се ставља *HSV* слика. Трећи параметар је код који одређује који простор боја конвертујемо. У овом случају биће потребан код *COLOR\_BGR2HSV*. Конвенционално уведени распони нијанси за *R*, *G* и *B* компоненте су: 0-255 за 8-битне слике, 0-65535 за 16-битне слике, што значи да је за 8-битне слике могуће добити комбинацијом три компоненте до 16,8 милиона нијанси ( $256 R * 256 G * 256 B$ ), а за 16-битне слике до 281 трилион могућих нијанси ( $65536 R * 65536 G * 65536 B$ ). Методом *cvtColor()* у случају 8 и 16-битних слика, *R*, *G*, и *B* компоненте се скалирају на вредности од 0-1 при чему се добијају *HSV* вредности на следећи начин:

$$\begin{aligned} V &\leftarrow \max(R, G, B) \\ S &\leftarrow \begin{cases} \frac{V - \min(R, G, B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases} \\ H &\leftarrow \begin{cases} 60(G - B)/(V - \min(R, G, B)) & \text{if } V = R \\ 120 + 60(B - R)/(V - \min(R, G, B)) & \text{if } V = G \\ 240 + 60(R - G)/(V - \min(R, G, B)) & \text{if } V = B \end{cases} \end{aligned}$$

*H* може да буде негативно, у том случају се узима вредност  $H = H + 360$ . Тако да је:  $0 \leq V \leq 1$ ,  $0 \leq S \leq 1$ ,  $0 \leq H \leq 360$ . За 8-битне слике *H*, *S*, *V* вредности се конвертују на следећи начин  $V = 255 * V$ ,  $S = 255 * S$ ,  $H = H/2$  (да би се добио број 0-255). За 16-битне слике конверзија у *HSV* простор још увек није омогућена у *OpenCV* библиотеци.

Од фрејма(слике) у *HSV* моделу је лакше добити бинарну слику(урадити трешхолд) због уједначене осветљености него од *RGB* слике. Конверзијом у бинарну слику постиже се знатно смањење неодређености слике, односно

издвајање оног а што је битно и занемаривање остатка слике. Бинарна слика се добија методом *inRange()*. [19]

Метода *inRange()* проверава да ли се вредности елемената низа(матрице) се налазе између вредности елемената друга два низа(матрице). Метода прима 4 параметра. Први параметар - *src* би требао да буде *HSV* матрица, други параметар - *lowerb* су минималне *HSV* вредности под којима се елемент види, трећи - *upperb* су максималне *HSV* вредности под којима се елемент види, четврти - *dst* параметар је матрица у коју се смешта бинарна слика. За сваки елемент *I* једнодимензионог низа је:

$$dst(I) = lowerb(I)_0 \leq src(I)_0 \leq upperb(I)_0$$

$$dst(I) = lowerb(I)_0 \leq src(I)_0 \leq upperb(I)_0 \wedge lowerb(I)_1 \leq src(I)_1 \leq upperb(I)_1$$

Након примењивања методе *inRange()* добија се бинарна слика где пиксели који се налазе у оквиру *min* и *max HSV* вредности постају бели, остали пиксели су црни. Као што је речено, може се појавити шум на слици, што се може елиминисати морфолошким операцијама ерозијом и дилатацијом. *OpenCV* има уграђене алгоритме за ерозију и дилатацију, *erode()* и *dilate()*. За успешније уклањање шума, потребно је одредити димензије елемента за ерозију и елемента за дилатацију, што опет зависи од количине шума и димензија саме слике. [18]

### 3.2.2.2. Проналажење контура (детекција ивица)

Контуре односно ивице објекта се могу пронаћи помоћу методе *findContours()* која као улазни параметар прима бинарну слику(трешхолдовану слику), излазни параметар је скуп контура(скуп скупова свих тачака спољашње линије објекта - бели простори у бинарној слици). [16]

### 3.2.2.3. *Moments* метода

Моменти слике у рачунарској визији представљају одређени део слике који садржи пикселе сличног интезитета који описују објекте. Моменти слике се користе за описивање објекта након сегментације слике. У математици за континуалну *2D* функцију  $f(x,y)$  моменти се рачунају формулом:

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy$$

За  $p, q = 0, 1, 2, \dots$

Прилагођавајући ову формулу бинарној слици са интезитетима пиксела  $(x,y)$  моменти  $M_{ij}$  се могу израчунати као:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

Централни моменти се рачунају по формули:

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y) dx dy$$

где су  $\bar{x} = \frac{M_{10}}{M_{00}}$  и  $\bar{y} = \frac{M_{01}}{M_{00}}$  координате центра објекта, односно тежиште.

Ако је  $f(x,y)$  дигитална слика, онда претходна једначина постаје:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y)$$

Пример за централне моменте трећег реда:

$$\mu_{00} = M_{00},$$

$$\mu_{01} = 0,$$

$$\mu_{10} = 0,$$

$$\mu_{11} = M_{11} - \bar{x}M_{01} = M_{11} - \bar{y}M_{10},$$

$$\mu_{20} = M_{20} - \bar{x}M_{10},$$

$$\mu_{02} = M_{02} - \bar{y}M_{01},$$

$$\mu_{21} = M_{21} - 2\bar{x}M_{11} - \bar{y}M_{20} + 2\bar{x}^2M_{01},$$

$$\mu_{12} = M_{12} - 2\bar{y}M_{11} - \bar{x}M_{02} + 2\bar{y}^2M_{10},$$

$$\mu_{30} = M_{30} - 3\bar{x}M_{20} + 2\bar{x}^2M_{10},$$

$$\mu_{03} = M_{03} - 3\bar{y}M_{02} + 2\bar{y}^2M_{01}.$$

*Moments* – класа *OpenCV*-а, која има методе које за улазни параметар примају скуп контура(ивица слике) добијен методом *findContours()*. Као резултат метода даје  $x, y$  координате тачака контуре објекта које су дефинисане својим унутрашњим простором - моментима. Моменти контуре у *OpenCV* библиотеци се рачунају на принципу Гринове формуле. [16][10]

Централни моменти представљају центар масе односно тежиште белих пиксела, добијају се на основу формуле:



$$\mu_{ji} = \sum_{x,y} (\text{array}(x, y) \cdot (x - \bar{x})^j \cdot (y - \bar{y})^i)$$

Где је  $(\bar{x}, \bar{y})$  центар масе односно тежиште, а

$$\bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}}$$

Централни моменти се нормализују степеном  $(i+1)/2 + 1$ .

$$\nu_{ji} = \frac{\mu_{ji}}{m_{00}^{(i+j)/2+1}}$$

$$\mu_{00} = m_{00}, \nu_{00} = 1 \quad \nu_{10} = \mu_{10} = \mu_{01} = \nu_{10} = 0$$

### 3.2.3. Одређивање узорака, калибрација

Одређивање узорака заправо представља одређивање фази скупова и одређивање функција припадности за објекте фази скупова, чиме се заправо одређују критеријуми за класификацију. Сваки објекат чине пиксели који припадају у одређеној мери распону између минималних и максималних  $H, S, V$  вредности односно пиксели су елементи фази скупа који је заправо тај распон а минималне и максималне  $H, S, V$  вредности су функције припадности. На пример објекти црвених нијанси ће припадати у одређеној мери фази скупу црвених објеката, одређеним минималним и максималним  $H, S, V$  вредностима црвене функције припадности. За сваку боју ће постојати посебан фази скуп и посебне функције припадности. Одређивање функције припадности може се урадити помоћу калибрације.

Пошто софтвер за класификацију треба да опслужује динамички систем попут покретне траке у реалном времену, због велике неодређености самог система потребно је осмислити добар начин за одређивање функције припадности. У оваквом систему то је могуће урадити узимајући мануелно узорак одређеног објекта са покретне траке, а узорак се одређује по боји објекта, па је потребно издвојити боје од интереса. Тај процес одређивања функције припадности се назива калибрација.

### 3.2.3.1. Калибрација боја

Калибрација боја представља одређивање минималних и максималних *HSV* вредности под којима се објекат издваја из позадине. У зависности колико боја се прати, за сваку би требало одредити које су то вредности и на неки начин их уписати у модел за класификацију.

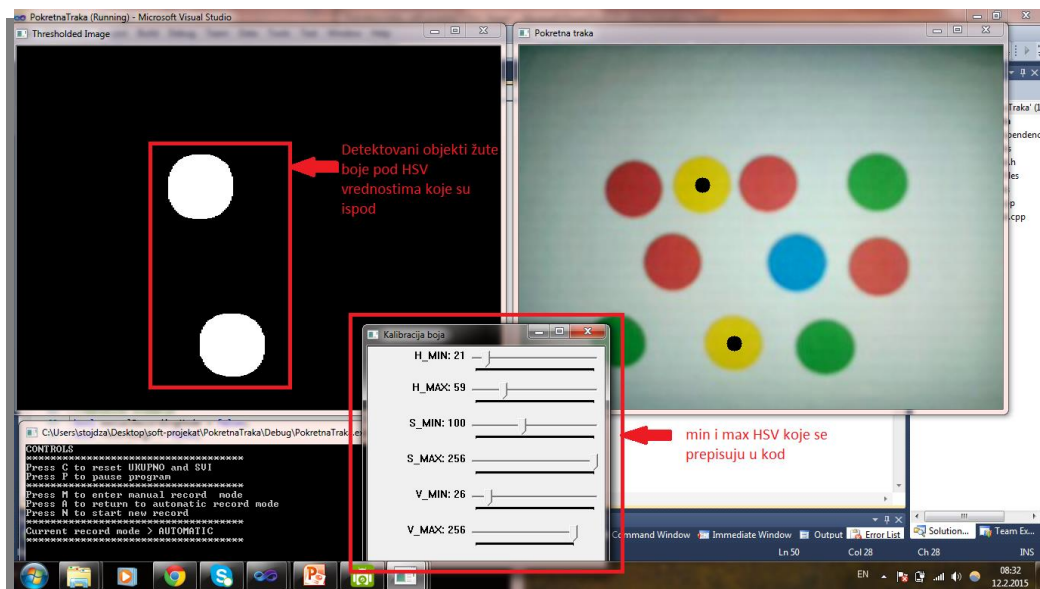
Под одређеним *min* и *max HSV* вредностима одређене боје, пиксели са *HSV* слике под тим вредностима у бинарној слици постају бели, односно објекат је издвојен из позадине, скуп груписаних белих пиксела представља објекат који се прати. Тешко је претпоставити које су то вредности минималне и максималне *HSV* вредности под којима ће објекти бити бели у бинарној слици. У *OpenCV*-у је опсег *HSV* вредности 0-255 за сваку компоненту (*H,S,V*). Одређивање *min* и *max HSV* вредности је заправо одређивање функције припадности објеката њиховим фази скуповима. Након одређивања вредности компоненти, оне се уписују у модел за класификацију.

Пример:

```
//одређивање min и max HSV вредности из HSV слике при калибрацији боја
inRange(HSV,Scalar(H_MIN,S_MIN,V_MIN),Scalar(H_MAX,S_MAX,V_MAX),threshold);
//уписивање вредности у модел за класификацију
//конструктор са параметром name
Objekat(string name){
    setType(name);
    if(name=="zut"){
        setHSVmin(Scalar(12,137,123));
        setHSVmax(Scalar(39,256,235));
        setColour(Scalar(0,255,255));
    }
}
//након калибрације одређују се min и max HSV вредности за сваку боју, на пример за жуту боју

Objekat("zut");
inRange(HSV,zut.getHSVmin(),zut.getHSVmax(),threshold);
//овај поступак би требао да се одради за све боје од интереса
```

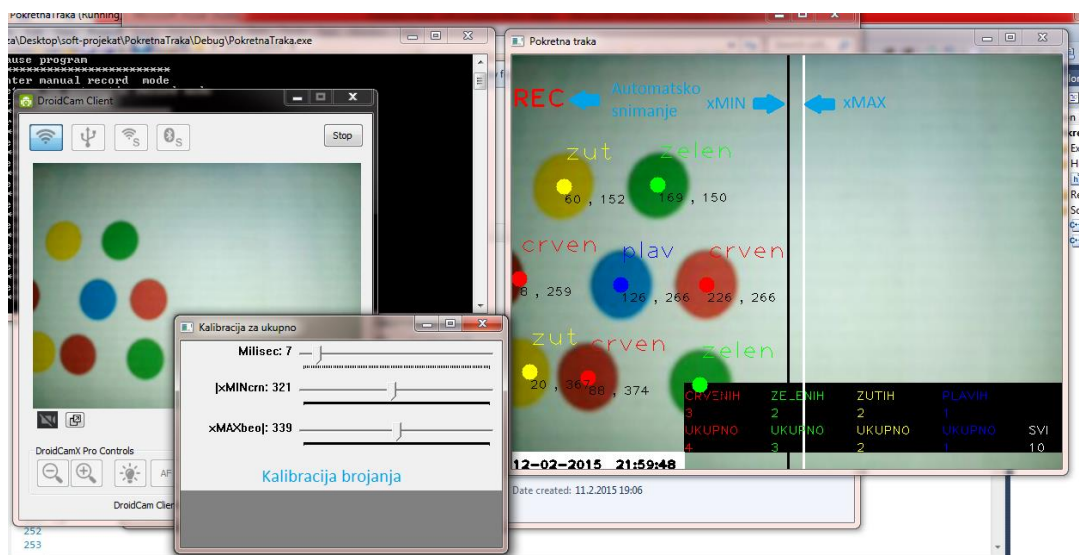
Потребно је направити функционалност којом ће лако моћи да се провере све ове вредности за објекте на слици. Један од начина је да се направе слајдери (*trackbars*) са поменутиим распоном за сваку компоненту посебно, *Hmin* и *Hmax*, *Smin* и *Smax*, *Vmin* и *Vmax*. Померањем слајдера за *min* и *max HSV* вредности пиксели који не припадају опсегу у бинарној слици постају црни, а пиксели који припадају постају бели. Када се добије изоштрено одвајање за одређени објекат, одређена је његова функција припадности, и он представља узорак за све објекте чија је боја слична његовој који ће припадати једном фази скупу. На Слици 16. је приказана калибрација боја, односно *HSV* вредности за жуту боју.



Слика 16. Калибрација боја

### 3.2.3.2. Калибрација бројања објеката

Покретна трака се креће одређеном брзином, а софтвер у одређеном интервалу прима слику са камере и обрађује је. Један од начина да се успешно врши укупно бројање објеката који су прошли кроз кадар, односно објеката на видео снимку јесте да се усклади калибрација толеранције за бројање ( $x_{MIN}$  i  $x_{MAX}$ ) и калибрација временског интервала добављања слике са камере. Да би вођење евиденције о објектима који су прошли кроз кадар, односно бројање објеката било што прецизније, потребно је ускладити – ширину опсега  $x_{MIN}$  и  $x_{MAX}$  вредности у којем се објекат детектује и броји и временски интервал добављања слике са камере са брзином кретања објеката. На Слици 17. је приказана калибрација бројања.  $x_{MIN}$  је представљен линијом црне боје, а  $x_{MAX}$  линијом беле боје. У интервалу између  $x_{MIN}$  и  $x_{MAX}$  се врши укупно бројање свих објеката који су прошли на траци. *Milisc* представља временски интервал добављања слике са камере. Уколико се добро усклади калибрација бројања са брзином траке бројање би било прецизно али при великом броју објеката прецизност је мања.



Слика 17. Калибрација бројања

Други начин за укупно бројање објеката који су прошли кроз кадар који би елиминисао употребу калибрације бројања је компликованији, али није немогућ за имплементацију. Компликованији је јер имплементација софтвера на начин описан у раду се базира на обради сваког фрејма(слике) видеа појединачно у реалном времену где је непредвидиво где ће се објекти налазити пре него што уђу у кадар. Потребно је одабрати одговарајући критеријум који ће сваки објекат чинити јединственим а у исто време у сваком наредном фрејму препознати тај објекат који још није прошао кроз кадар камере као објекат из претходног фрејма, или на неки начин упамтити тај објекат и заборавити на њега у сваком наредном кадру, али тиме би изгубили могућност праћења путање објекта кроз кадар камере. Иако компликованији за имплементацију, овакав начин би пребројавао објекте са већом тачношћу.

### 3.2.4. Исцртавање облика на видеу

*OpenCV* садржи методе за исцртавање разних облика на слици. Ове методе се могу употребити у циљу побољшања слика из реалног света (примена аугментоване или побољшане реалности.) Након калибрације боја и калибрације бројања може се пратити тренутно стање објеката на видеу са Сlike 20 на ком су у сваком моменту док се снима:

- објекти обележени одговарајућом бојом
- испод обележја су координате објекта у односу на координатни почетак видеа
- изнад сваког објекта је његов назив
- исцртане су црна и бела линија које представљају  $xMIN$  и  $xMAX$  између којих се евидентира и врши укупно бројање објеката

- у доњем десном углу videa се налази број објеката у тренутном кадру, а испод њега је укупан број објеката који су прошли кроз кадар камере
- у доњем левом углу videa се налази датум и време снимања videa

Обележавање објекта је урађено на следећи начин:

```
//иницијализација објекта
Objekat objekat;
//исцртавање круга боје објекта у центру сваког објекта
circle(cameraFeed,cv::Point(objekat.getXPos(),objekat.getYPos()),9,objekat.getColour(),-1);
// исписивање координата објекта и назива објекта методом putText
putText(cameraFeed,intToString(objekat.getXPos())+ " , " +
intToString(objekat.getYPos()),cv::Point(objekat.getXPos(),objekat.getYPos()+20),1,
1,Scalar(0,0,0));

putText(cameraFeed,objekat.getType(),cv::Point(objekat.getXPos(),objekat.getYPos()-
30),1,2,objekat.getColour());
```

Исцртавање линија за калибрацију бројања  $x_{MIN}$  и  $x_{MAX}$ :

```
line(cameraFeed,Point(xMIN,0),Point(xMIN,480),crBoja,2,8,0);
line(cameraFeed,Point(xMAX,0),Point(xMAX,480),beBoja,2,8,0);
```

Испис укупног броја објеката који су прошли кроз кадар:

```
//пример исписа укупног броја жутих објеката, zuVector је вектор(листа) који садржи
пронађене објекте жуте боје
putText(cameraFeed,"UKUPNO",cv::Point(400,440),1,1,zuBoja);
putText(cameraFeed,intToString(zuVector.size()),cv::Point(400,460),1,1,zuBoja);
//слично за све остале објекте

//испис укупног броја свих објеката(збир свих вектора(листа) који садрже објекте)
putText(cameraFeed,"SVI",cv::Point(600,440),1,1,beBoja);
putText(cameraFeed,intToString(plVector.size()+crVector.size()+zeVector.size()+zuVector.size()),cv::Point(600,460),1,1,beBoja);
```

Испис бројчаног стања објеката у тренутном кадру:

```
//испис тренутног броја жутих објеката

if(zutiTren.size()>0) {
    putText(cameraFeed,"ZUTIИ",cv::Point(400,400),1,1,zuBoja);
    putText(cameraFeed,intToString(zuVectorTren.size()),cv::Point(400,420),1,1,zuBoja);
}
//слично за све остале објекте
```

Датум и време су исписани на следећи начин:

```
//испис датума и времена getDateTIme() је функција која добавља системско
време са рачунара
rectangle(cameraFeed,Point(0,460),Point(200,480),beBoja,-1);
putText(cameraFeed,getDateTIme(),Point(0,480),1,1,Scalar(0,0,0),2);
```

### 3.2.5. Снимање видеа

Софтвер треба да омогући кориснику да снима видео покретне траке у жељеним временским тренутцима. На видео се могу у виду виртуалне (аугментоване) реалности додати подаци као што су боја објекта, тренутна позиција објекта, тренутно време тако да се може добити положај објекта у односу на одређен временски тренутак, подаци о трајекторији објекта итд. Потребно је покренути снимање командом, и потом одабрати режим снимања.

Снимање би требало омогућити у два режима: аутоматски и мануелно. Треба омогућити у сваком моменту из аутоматског режима снимања прећи у мануелни и обрнуто, као и искључити и укључити снимање или паузирати.

#### 3.2.5.1. Аутоматски режим снимања видеа

Видео се снима аутоматски док се у кадру налазе објекти, када на траци нема објеката снимање се аутоматски зауставља све док не наиђе неки нови објекат потом се наставља снимање и цео процес се снима у исти фајл.

### 3.2.5.2. Мануелни режим снимања видеа

Мануелно снимање би требало да се може иницирати у било ком моменту рада софтвера помоћу команди. Могуће је у било ком моменту паузирати снимање видеа и поново наставити са снимањем, или укључити аутоматски режим снимања.

## 3.3. Вођење евиденције о објектима

За време проласка кроз кадар подаци о објектима (боја објекта, *id* објекта, *x* и *y* позиција у односу на видео, време када је детектован) се уписују у текстуални фајл на хард диску (*Log*).

## 3.4. Команде

Команде за манипулацију софтвером се изводе путем тастатуре. Команде се могу имплементирати на разне начине, зависно од програмског језика у којем се софтвер имплементира. Најједноставније решење за имплементацију команди би била употреба *switch* петље која је активна сво време рада софтвера.

## 4. Верификација

### 4.1. Симулација траке

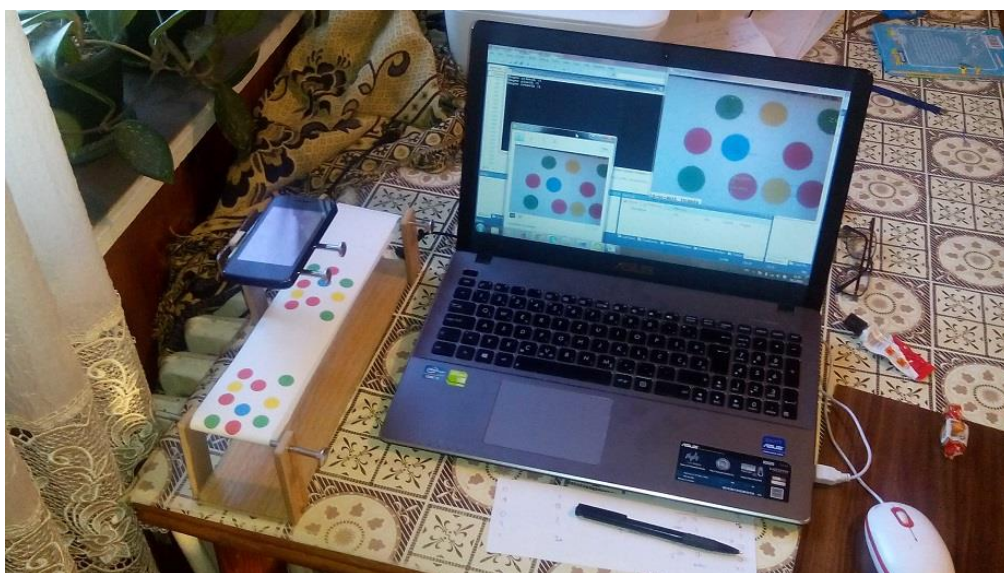
За потребе тестирања софтвера може бити коришћена симулација покретне траке на ручни погон (Слика 18.). Трака је бела и на њој се налазе скупови објеката (тачке) различите боје (црвена, плава, зелена и жута). Ради веће изолованости система пожељно је укључити LED светло телефона како би осветљење било једнако у свим деловима слике и боје јасно дефинисане. Систем за класификовање би требао да ради независно од облика објеката који се налазе на траци, класификација ће се вршити на основу боје.





**Слика 18.** Симулација покретне траке на ручни погон

Слика ће бити добијена помоћу камере паметног телефона са андроид оперативним системом који је фиксиран на симулацији покретне траке(Слика 19.). Слика ће бити послата рачунару у реалном времену помоћу програма *Droid cam* који се може инсталирати на андроид телефон са *Google app store*. Потребно је такође инсталирати и сервер за апликацију. Сервер се инсталира на рачунар на који се имплементира софтвер за праћење и класификацију објеката.

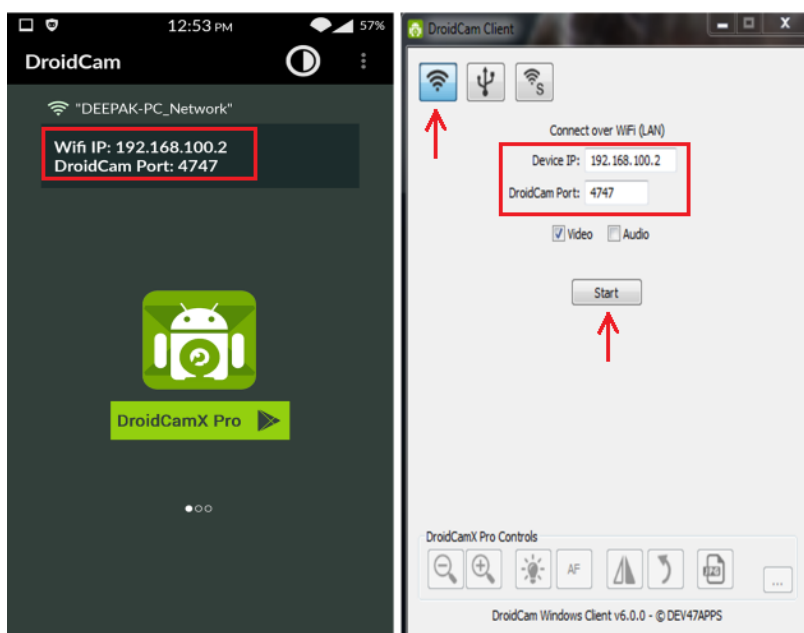


**Слика 19.** Слика са камере андроид телефона на рачунару



## 4.2. Повезивање рачунара и камере телефона

Телефон са андроид системом и рачунар требају бити умрежени на исти *Wi-Fi* (*wireless*) рутер путем којег ће размењивати слику са камере. На Слици 20. приказан је са леве стране интерфејс *DroidCam* апликације за андроид телефон а десно је интерфејс *DroidCam Client* апликације за рачунар. У *DroidCam Client* апликацији потребно је обележити дугме за повезивање преко *Wi-Fi* мреже, прекуцати вредности “*Wi-Fi IP*” и “*DroidCam Port*” из андроид апликације и кликнути на дугме *Start*. [21][22][23]



Слика 20. Повезивање рачунара и камере телефона

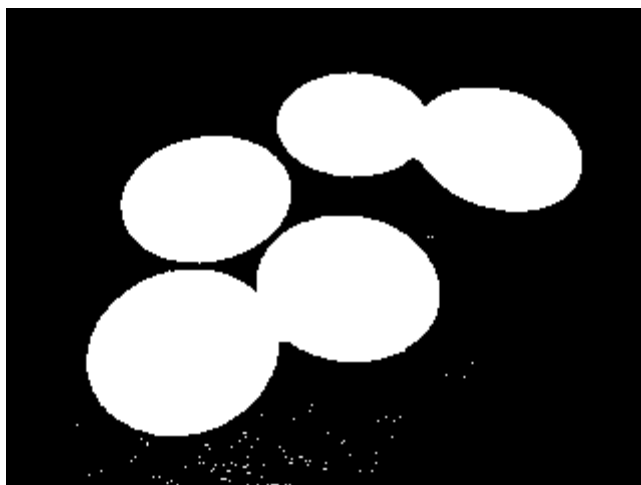
## 4.3. Специјални случај код бројања

У идеалним условима објекти су на траци одвојени и начин њиховог препознавања који је описан у овом систему одлично ради, међутим у реалном свету објекти се често налазе једни преко других (Слика 21). Начин препознавања који је описан у овом раду би два спојена објекта препознавао као један објекат.



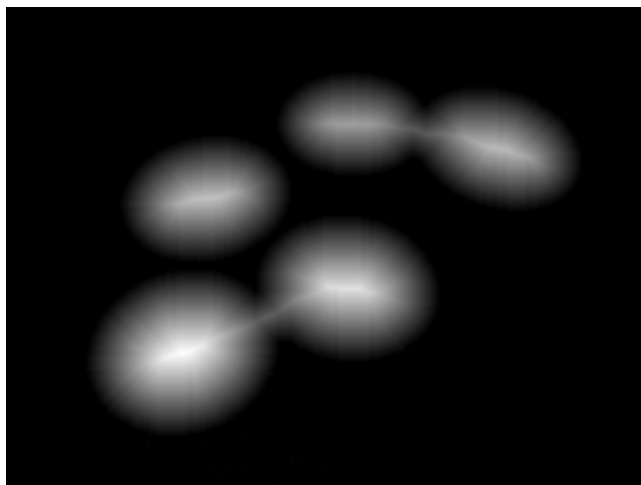
**Слика 21.** Спојени објекти (новчићи)

Потребно је слику конвертовати у бинарну(Слика 22.).



**Слика 22.** Бинарна(трешхолдована) слика

Трансформација дистанце се примењује на бинарне слике која резултује добијањем слике са нијансама сиве која је слична бинарној слици, с тим да је интензитет сиве боје јачи ка ободу објеката, односно ка позадини и он слаби ка централном делу објекта. Након трансформације потребно је нормализовати добијену слику на вредности  $[0,1]$  како би се смањили бели пиксели који представљају објекте, да буду одвојени црним пикселима односно позадином.(Слика 23.)



**Слика 23.** Нормализована слика

Нормализована слика се опет конвертује у бинарну при чему се добија резултујућа слика где су објекти потпуно одвојени једни од других(Слика 24.).



**Слика 24.** Потпуно одвојени објекти

Од овако добијене слике је лако пронаћи контуре користећи претходно описану методу *OpenCV-a findContours()*. Након овог корака већ може да се добије увид у бројчано стање објеката на слици, и изврши њихова класификација методама описаним у овом раду.[13]

## 5. Закључак

У раду је представљен модел и имплементиран је софтвер за праћење и класификацију објеката на основу боје који се налазе на покретној траци. Представљен је опис функционалности за снимање целокупног процеса рада покретне траке и опис система за праћење објеката и вођење евиденције о испраћеним објектима. У даљем развоју овог решења потребно је омогућити да калибрација бројања ради независно од брзине кретања траке. У даљем развоју софтвера могуће је проширити његове функционалности са:

- вођењем комплетне евиденције свих позиција објекта
- исцртавањем путање кретања за сваки предмет
- одређивањем величине објекта на траци
- одређивањем растојања између објеката
- рачунањем брзине кретања објекта
- аутоматским одређивањем калибрације боја обележавањем помоћу миша
- препознавањем боја помоћу неуронске мреже
- праћењем објекта независно од боја
- генерисањем графика учесталости објеката у односу на временски период

Осим коришћења за надзор покретне траке у индустријске сврхе софтвер би могао да се користи у праћењу и евиденцији микро процеса, у медицини (праћење ћелија и процеса у ћелијама), биологији, физици, математици и другим наукама.

# Литература

- [1] *Soft computing* теорија проф. Др. Ђорђе Обрадовић
- [2] Морфолошка обрада слике, Владимир Црнојевић,  
[http://www.ktios.net/stari/images/stories/clanovi\\_katedre/vladimir\\_cрноjevic/DOS/Po%20glavlje%208%20-%20Morfoloska%20obrada%20slike.pdf](http://www.ktios.net/stari/images/stories/clanovi_katedre/vladimir_cрноjevic/DOS/Po%20glavlje%208%20-%20Morfoloska%20obrada%20slike.pdf)
- [3] *Morphological image processing*,  
<https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.html>
- [4] Компјутерска визија – *Content base image retrieval*  
<http://www.demijan.com/Prof%20site/projects/Kompjuterska%20vizija%20-%20Content%20based%20image%20retrieval.pdf>
- [5] *Wikipedia, Computer vision*,  
[https://en.wikipedia.org/wiki/Computer\\_vision](https://en.wikipedia.org/wiki/Computer_vision)
- [6] *Wikipedia, outline object tracking*,  
[https://en.wikipedia.org/wiki/Outline\\_of\\_object\\_recognition](https://en.wikipedia.org/wiki/Outline_of_object_recognition)
- [7] *Wikipedia, Green theorem*,  
[https://en.wikipedia.org/wiki/Green%27s\\_theorem](https://en.wikipedia.org/wiki/Green%27s_theorem)
- [8] *Wikipedia, RGB color model*,  
[https://en.wikipedia.org/wiki/RGB\\_color\\_model](https://en.wikipedia.org/wiki/RGB_color_model)
- [9] *Wikipedia, HSV and HSL*,  
[https://en.wikipedia.org/wiki/HSL\\_and\\_HSV](https://en.wikipedia.org/wiki/HSL_and_HSV)
- [10] *Wikipedia Image Moments*,  
[https://en.wikipedia.org/wiki/Image\\_moment](https://en.wikipedia.org/wiki/Image_moment)
- [11] *Wikipedia Fuzzy Logic*,  
[https://en.wikipedia.org/wiki/Fuzzy\\_logic](https://en.wikipedia.org/wiki/Fuzzy_logic)
- [12] *OpenCV documentation, structural analysis and shape detection*,  
[http://docs.opencv.org/2.4/modules/imgproc/doc/structural\\_analysis\\_and\\_shape\\_descriptors.html?highlight=moments#moments](http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=moments#moments)
- [13] *OpenCV Image Segmentation*,  
[http://docs.opencv.org/3.1.0/d3/db4/tutorial\\_py\\_watershed.html#gsc.tab=0](http://docs.opencv.org/3.1.0/d3/db4/tutorial_py_watershed.html#gsc.tab=0)

- [14] *OpenCV changing color spaces*,  
[http://docs.opencv.org/trunk/df/d9d/tutorial\\_py\\_colorspaces.html#gsc.tab=0](http://docs.opencv.org/trunk/df/d9d/tutorial_py_colorspaces.html#gsc.tab=0)
- [15] *OpenCV documentation, reading and writing image and video* ,  
[http://docs.opencv.org/2.4/modules/highgui/doc/reading\\_and\\_writing\\_images\\_and\\_video.html](http://docs.opencv.org/2.4/modules/highgui/doc/reading_and_writing_images_and_video.html)
- [16] *OpenCV Image Moments*,  
<http://docs.opencv.org/2.4/doc/tutorials/imgproc/shapedescriptors/moments/moments.html>
- [17] *OpenCV Overlapping Objects*,  
<https://opencv-code.com/tutorials/count-and-segment-overlapping-objects-with-watershed-and-distance-transform/>
- [18] *OpenCV Miscellaneous transformation cvtColor()*  
[http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous\\_transformations.html](http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html)
- [19] *OpenCV operations on arrays inRange()*  
[http://docs.opencv.org/2.4/modules/core/doc/operations\\_on\\_arrays.html](http://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html)
- [20] *Mashine vision software*,  
<http://machinevisionsoftware.com/>
- [21] *Droid cam software*,  
<https://www.dev47apps.com/>
- [22] *Droid Cam* инсталација са *google app store*,  
<https://play.google.com/store/apps/details?id=com.dev47apps.droidcam&hl=en>
- [23] Сервер за *Droid Cam* апликацију,  
<https://www.dev47apps.com/droidcam/windows/>
- [24] Структуре и методе у процесирању сигнала и слика, математичке инваријанте, Александар перовић, Един Долићанин, Александар Јовановић

## Биографија

Стојан Митрић је рођен 19.05.1992. године у Новом Саду. Основну школу „Иво Андрић“ у Будисави завршио је 2007. године. Гимназију „Јован Јовановић Змај“ у Новом Саду завршио је 2011. године. Исте године уписао се на Факултет техничких наука, одсек Рачунарство и аутоматика. Школске 2013/2014. године уписао се на смер Рачунарске науке и информатика. Положио је све испите предвиђене планом и програмом.

## Кључна документацијска информација

Редни број, <b>РБР:</b>	
Идентификациони број, <b>ИБР:</b>	
Тип документације, <b>ТД:</b>	монографска публикација
Тип записа, <b>ТЗ:</b>	текстуални штампани документ
Врста рада, <b>ВР:</b>	дипломски рад
Аутор, <b>АУ:</b>	Стојан Митрић
Ментор, <b>МН:</b>	Проф. Др. Ђорђе Обрадовић, ФТН Нови Сад
Наслов рада, <b>НР:</b>	Софтвер за класификацију објеката на покретној траци
Језик публикације, <b>ЈР:</b>	Српски
Језик извода, <b>ЈИ:</b>	Српски / Енглески
Земља публиковања, <b>ЗП:</b>	Србија
Уже географско подручје, <b>УГП:</b>	Војводина
Година, <b>ГО:</b>	2016
Издавач, <b>ИЗ:</b>	ауторски репринт
Место и адреса, <b>МА:</b>	Нови Сад, Факултет техничких наука, Трг Доситеја Обрадовића 6
Физички опис рада, <b>ФО:</b>	бр. поглавља: 16 / страница: 41 / цитата: 12 / табела: 1 / слика: 24 / фигура: 32 / прилога: 0
Научна област, <b>НО:</b>	Рачунарске науке и информатика
Научна дисциплина, <b>НД:</b>	Софт компјутинг
Предметна одредница / кључне речи, <b>ПО:</b>	Праћење, класификација, и бројање објеката на основу боје, Компјутерски вид, Анализа и обрада слике у реалном времену, Виртуелна и побољшана реалност
<b>УДК</b>	
Чува се, <b>ЧУ:</b>	Библиотека Факултета техничких наука, Трг Доситеја Обрадовића 6, Нови Сад
Важна напомена, <b>ВН:</b>	
Извод, <b>ИЗ:</b>	Опис и имплементација софтвера за анализу, класификацију, праћење, вођење евиденције и снимање видеа за објекте који пролазе на покретној траци у реалном времену.
Датум прихватања теме, <b>ДП:</b>	
Датум одбране, <b>ДО:</b>	
Чланови комисије, <b>КО:</b>	
председник	др Ђорђе Обрадовић, редовни професор, ФТН Нови Сад
члан	
ментор	
Потпис ментора	



## Key words documentation

Accession number, <b>ANO</b> :	
Identification number, <b>INO</b> :	
Document type, <b>DT</b> :	monographic publication
Type of record, <b>TR</b> :	textual material
Contents code, <b>CC</b> :	BSc thesis
Author, <b>AU</b> :	Stojan Mitrić
Mentor, <b>MN</b> :	Đorđe Obradović, PhD full prof., FTN Novi Sad
Title, <b>TI</b> :	Software for classification of objects on conveyer belt
Language of text, <b>LT</b> :	serbian
Language of abstract, <b>LA</b> :	serbian / english
Country of publication, <b>CP</b> :	Serbia
Locality of publication, <b>LP</b> :	Vojvodina
Publication year, <b>PY</b> :	2016
Publisher, <b>PB</b> :	author's reprint
Publication place, <b>PP</b> :	Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6
Physical description, <b>PD</b> :	chapters: 16 / pages:41 / references: 12 / tables: 1 / images: 24 / figures: 32
Scientific field, <b>SF</b> :	Computer science
Scientific discipline, <b>ND</b> :	Soft computing
Subject / Keywords, <b>S/KW</b> :	Motion tracking, classification and counting object depending on their color, Computer vision, Analysis and processing of image, Virtual and Augmented reality
<b>UDC</b>	
Holding data, <b>HD</b> :	Library of the Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad
Note, <b>N</b> :	
Abstract, <b>AB</b> :	This paper describes the working process and implementation of software for analyzing, motion tracking, logging and recording video for objects which are moving on conveyor belt in real time.
Accepted by sci. board on, <b>ASB</b> :	
Defended on, <b>DE</b> :	
Defense board, <b>DB</b> :	
president	Đorđe Obradović, PhD, full prof., FTN Novi Sad
member	
mentor	
Mentor's signature	