Универзитет у Крагујевцу

Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Рачунарска техника и софтверско инжењерство

Ниво студија: Основне академске студије

Предмет: Рачунарски алати

Број индекса: 626/2018

Ђорђе С. Гачић

Скенер за класификацију савијених алуминијумских и PVC профила коришћењем 3D камере

Дипломски рад

Комисија за преглед и одбрану:

1. **Проф. Др. Владимир Миловановић - ментор**
2. _____
3. _____

Датум одбране: _____

Оцена: _____

У оквиру овог дипломског рада кандидат треба да, са снимка 3D камере, издвоји профил скалирањем по дубини пиксела. Интензитет пиксела пропорционалан је растојању објекта од камере, а не интензитету свјетлости. Профил може бити окренут на било коју страну и имати различита закривљења. Класе у које треба сврстати профил разликују се по начину закривљења лука на профилу. Раван дио на профилу треба што више занемарити, а класификацију заснивати углавном на закривљеним дијеловима профила тј. луковима.

Препоручена литература:

[1] .....

[2] .....

[3] .....

Крагујевац, 13.08.2022.

Ментор:

Проф. Др. Владимир Миловановић

_____

Резиме

Циљ овог пројекта јесте развити софтвер који ће служити као скенер за класификацију већ савијених алуминијумских и PVC профила. Класификација се врши на основу облика лука тј. закривљеног дијела профила. Почетна слика профила се добија преко 3D камере код које је интензитет пиксела пропорционалан растојању објекта од камере, а не интензитету свјетлости. Профили могу имати различите облике и величине као и различите дужине несавијеног дијела који је потребно што више занемарити при класификацији. Врло битне ставке пројекта су издвајање конуре профила, њена припрема за што боље препознавање одбирака на ивици профила, као и препознавање оријентације јер профил може бити окренут на било коју страну. Главни алати за претходно наведене операције су функције из OpenCV библиотеке. Читав пројекат је имплементиран у програмском језику C#, с тим да је основа пројекта која садржи дио за повезивање апликације са 3D камером и приказ слике преузета већ готова под називом DepthBasics-WPF.

Кључне ријечи:
класификација, профил, лук, пиксел, 3D камера, контура, ивица, препознавање, OpenCv, DepthBasics-WPF

Abstract

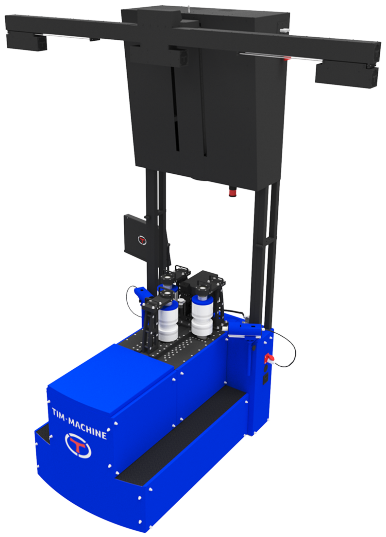
The goal of this project is to develop software that will serve as a scanner for the classification of already bent aluminum and PVC profiles. The classification should be done on the basis of the shape of the arch, i.e. curved part of the profile. The initial image of the profile is obtained through a 3D camera where the intensity of the pixels is proportional to the distance of the object from the camera, and not to the intensity of the light. Profiles can have different shapes and sizes, as well as different lengths of the unfolded part, which should be ignored as much as possible during classification. Very important items of the project are the extraction profile contour, its preparation for the best possible detection of samples on the edge of the profile, as well as the detection of the orientation because the profile can be turned to any side. The main tools for the aforementioned operations are functions from the OpenCV library. The entire project is implemented in the C# programming language, with the fact that the basis of the project, which contains the part for connecting the application to the 3D camera and displaying the image, is already downloaded under the name DepthBasics-WPF.

Keywords:
classification, profile, arc, pixel, 3D camera, contour, edge, detection, OpenCV, DepthBasics-WPF

Садржај:

1 Увод

У данашње вријеме све је већа потреба за аутоматизацијом разних задатака како у индустрији тако у осталим сферама живота. Све већи број фирми иде ка томе да машине које имају ширу намјену прилагоди сопственим потребама како би се процеси обављали много брже и прецизније. Задатак у оквиру овог пројекта се односи на скенирање и класификацију већ савијених алуминијумских и PVC профила коришћењем 3D камере. Пројекат се уједно ради за потребе фирме TIM-ING CENTAR у Крагујевцу, која се бави производњом најсавременијих машина за савијање алуминијумских и PVC профила. Машине посједују управљачке 3D камере и безбједносну 3D камеру које омогућавају аутоматско управљање, аутоматску контролу и безбједан рад на машини. Једна таква машина приказана је на следећој слици:



Слика 1: TIM-MACHINE MODEL M-CNC-3D [7]

Класификатор је дио пројекта 3D скенера профила који је тренутно у изради. Пројекат обухвата софтверски дио неопходан за класификацију и не улази у оквире процеса савијања. Основна намјена апликације је одредити коју врсту лука садржи већ савијен профил. Имплементација пројекта и врсте поменутих лукова биће детаљно објашњени у наставку.

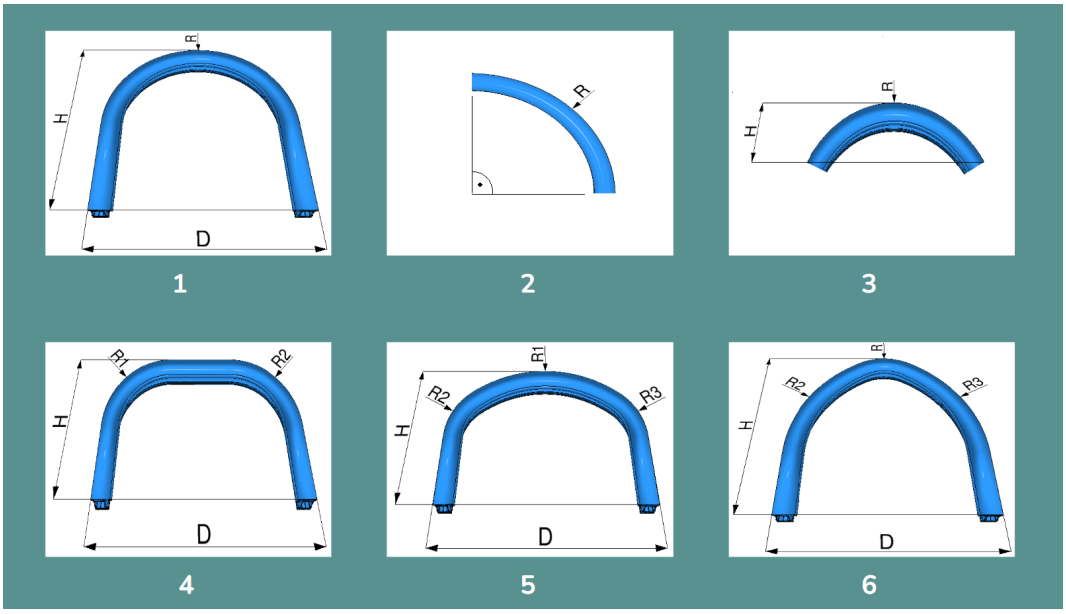
Овај рад не садржи дијелове програмског кода, али има јасно назначене смјернице ка функцијама и дијеловима кода у којима су поједини задаци имплементирани.

2 Детаљна поставка задатка

Улазни податак на основу кога се одређује врста лука коју профил садржи јесте снимак са 3D камере. Из тог снимка се издваја један фрејм који представља слику код које је интензитет пиксела пропорционалан растојању објекта од камере, а не интензитету свјетлости. Примјер једног тако издвојеног фрејма дат је на слици испод:

Слика 2: Слика са 3D камере

Примјећујемо да се на слици налази примјерак савијеног профила који је постављен на двије хоризонталне шипке и да је камера постављена изнад профила. Сваки пиксел има одређену дубину која представља удаљеност објекта, коме пиксел припада, од саме камере. Са слике видимо да, што су пиксели тамнији, то је објекат ближе камери, Оно што је главни задатак јесте класификација профила са слике у једну од следећих класа:



Слика 3: Класе профила

Да би се боље разумјеле врсте лукова које представљају класе у које се сврстава профил потребно је навести неке карактеристике лукова сваке класе означене бројевима као на претходној слици. Те карактеристике су (називи у заградама су скраћени називи класа који ће се користити у наставку):

- 1 - заклапа 180° (правилан лук)
- 2 - заклапа 90° (L лук)
- 3 - кружни исјечак који заклапа угао мањи од 180°, при чему различито од 90° (исјечак)
- 4 - заклапа два угла по 90° (n - лук)
- 5 - доња половина хоризонтално постављене елипсе (хоризонтална елипса)
- 6 - доња половина вертикално постављене елипсе (вертикална елипса)

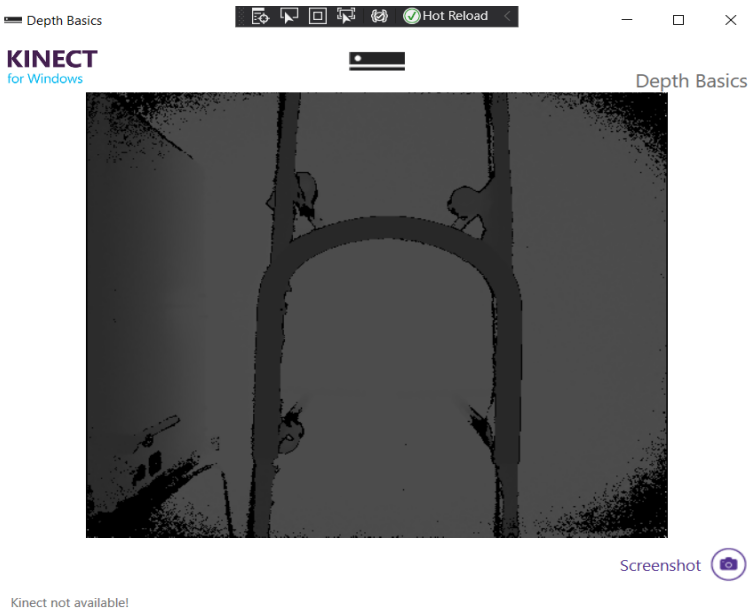
Очигледно је да примјерак профила са улазне слике приказане изнад (слика 2) припада класи означеној са 1 (правилан лук - заклапа 180°). Међутим, битно је напоменути да примјерци са претходне слике (слика 3) и примјерци у пракси не морају имати баш исти облик. На пример, раван дио на савијеним профилима не треба узимати у разматрање него само дио профила који представља лук, јер равни дио може у неким случајевима изостати, а у неким бити доста дужи, а да при томе угао лука остане исти. Такође, профил може бити окренут на било коју страну, тј. не мора бити окренут крацима ка горе. Главни фактор на основу кога се одређује припадност некој врсти (класи) профила је лучни дио профила и његов облик.

У оквиру овог дипломског рада, може се усвојити да су краци профила увијек исте дужине и да су у симетричном положају.

3 Коришћени ресурси за израду пројекта

За израду пројекта коришћени су следећи ресурси:

- 3D камера која даје дубинску матрицу слике
- Већ савијени профили за тестирање
- C# програмски језик [4][5]
- Microsoft Visual Studio 2019 развојно окружење
- OpenCvSharp - OpenCv [2][3] библиотека за обраду слике прилагођена развоју у C# програмском језику
- DepthBasics-WPF - преузет пројекат који нуди основне функционалности преузимања фрејмова са 3D камере и приказивање слике на основу дубинске матрице у оквиру WPF апликације израђене у C#. Преузет је преко апликације SDK Browser (Kinect for Windows) v2.0 [8]. Изглед прозора који је доступан у оквиру те апликације је следећи:



Слика 4: Depth Basics WPF Window

Види се да графички интерфејс нуди могућност приказа са снимка 3D камере тако да "удаљенији" пиксели од камере буду свјетлији, а ближи камери да буду тамнији.

Треба напоменути да су фрејмови у току израде учитавани из .csv фајла због немогућности свакодневног приступа камери и снимању профила уживо, па је томе прилагођен већ постојећи код.

4 Обрада слике и припрема за класификацију

Да би класификација профила била обављена прецизно и успјешно, неоходно је правилно издвојити профил са фрејма (слике) који је издвојен из снимка 3D камере. То је веома битан и сложен процес од којег касније зависи прецизност са којом се може рећи да неки профил припада одређеној класи. Обрада слике и припрема за класификацију пролази кроз следеће фазе које ће у наставку бити детаљно образложене:

- Скалирање по дубини пиксела
- Бинаризација
- Отклањање шума
- Издвајање контуре профила
- Одређивање оријентације профила
- Свођење на фиксну ширину слике
- Издвајање одбирака лука и њихово усредњавање

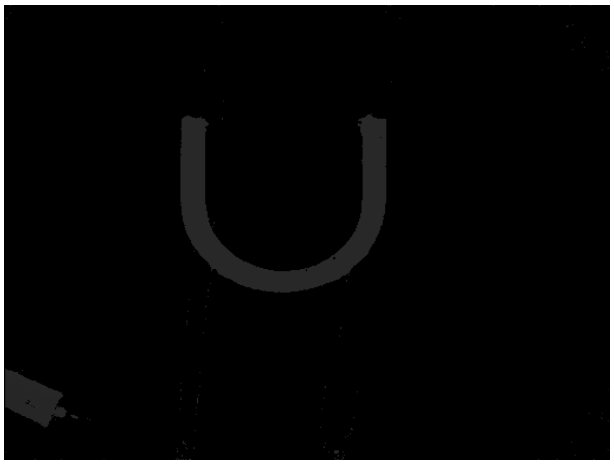
Битно је напоменути да све протходне фазе треба да се обављају редослиједом како су написане изнад.

4.1 Скалирање по дубини пиксела

Представља ограничавање дубине пиксела у дубинској матрици на неку вриједност. У нашем случају потребно је одбацити све што се налази физички испод профила. Вриједност дубине изнад које ће бити одбачени пиксели уноси се ручно, јер та вриједност зависи од висине платформе на којој се налази профил као и висине на којој се налази камера. То значи да ће поменута вриједност дубине бити једнака удаљености камере од платформе на коју је постављен профил. Такође је идеја да се камера подиже и спушта у зависности од величине профила, али то није предмет разматрања у оквиру овог пројекта.

У овом случају, "одбацивање" пиксела значи обојити их у црно у приказу тј. поставити им вриједност на 0 у `grayscale` метрици приказа слике.

Ако улазну слику (слика 1) скалирамо по дубини са одређеном вриједношћу која је потребна да се исправно скалира, добијамо следећи приказ:



Слика 5: Скалирано по дубини

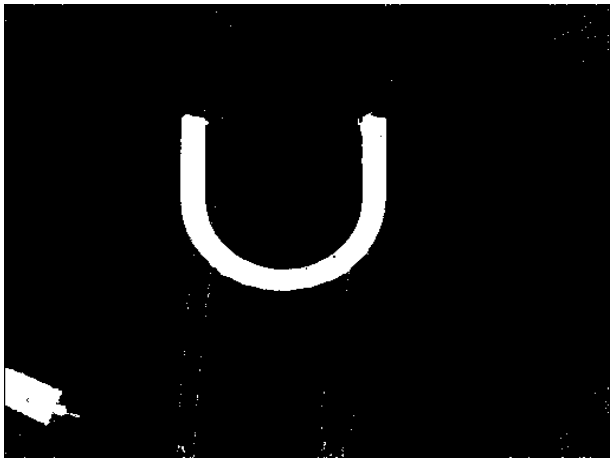
Са слике се види да је резултат баш онакав какав треба бити и да је скалирање исправно одрађено. Вриједност максималне дубине пиксела уноси оператер и добија се експериментално или мјерењем удаљености камере од платформе на којој се налази профил.

Имплементација скалирања по дубини, у оквиру овог пројекта, налази се унутар функције:

- `MainWindow.ProcessDepthFrameData`

4.2 Бинаризација

Бинаризација се односи на постојање само црне или бијеле боје на слици. У нашем случају, потребно је да бијелом бојом буде обојено све оно што није обојено црном након скалирања по дубини. Бинаризација се врши због једноставније и прецизније обраде у наредним корацима. У сврху поменуте бинаризације користи се уграђена функција `OpenCvSharp.Cv2.Threshold`. Од претходне слике (слика 5) добија се следећа:



Слика 6: Бинаризована слика

Имплементација жељене бинаризације, у оквиру овог пројекта, налази се унутар функције:

- `TimingScanner.ScannerUtils.ToBlackWhiteImage`

4.3 Отклањање шума на слици

Односи се на занемаривање ситних дијелова тј. пиксела који нису дијелови профила, а који су препознати на истој дубини као профил, или мањој. Такође помаже при усредњавању ивица профила, као и попуњавању тзв. шупљина на прифилу који су такође посљедица шума или, у неким случајевима, сировог материјала од кога је профил направљен.

Отклањање шума, у оквиру овог пројекта, обавља се примјеном морфолошких операција над бинаризованом сликом. Морфологија у процесирању дигиталних слика представља облик и структуру објекта. Постоје двије основне морфолошке операције и то:

- Ерозија (сужавање): користи се када је потребно отклоноти шум, сузити површину објекта, или раздвојити скуп објеката.
- Дилатација (проширивање): користи се када је потребно попунити шупљине објекта или спојити више објеката или дијелове објеката у један.

Често се користи и узастопна примјена дилатације и ерозије. У зависности од редослиједа примјене, одговарајуће трансформације су:

- Отварање: ерозија праћена дилатацијом - корисно при уклањању шума
- Затварање: дилатација праћена ерозијом - корисно при затварању малих рупа унутар објекта у првом плану или малих црних тачака на објекту

Начин на који се обављају операције ерозије и дилатације заснован је на постојању филтера (језгра, кернела) који представља матрицу бинарних вриједности (0 и 1). Облик филтера може бити различит у зависности од распореда нула и јединица. Најчешћи облици филтера су правоугаони, елиптички и укрштени. Примјер филтера величине 5x5 од све три поменуте врсте приказан је на слици испод:

```
[[1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1]]
```

а) Правоугаони филтер

```
[[0, 0, 1, 0, 0],
 [1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1],
 [0, 0, 1, 0, 0]]
```

б) Елиптички филтер

```
[[0, 0, 1, 0, 0],
 [0, 0, 1, 0, 0],
 [1, 1, 1, 1, 1],
 [0, 0, 1, 0, 0],
 [0, 0, 1, 0, 0]]
```

в) Укрштени филтер

Слика 7: Најчешће коришћени облици филтера

Филтер се креће кроз слику (као у 2D конволуцији). Нека је објекат на слици обојен бијелом бојом а позадина црном и нека су вриједности пиксела бијеле боје једнаки 1, а црне 0. Ако се ради о ерозији, пиксел у оригиналној слици (било 1 или 0) ће се подесити на 1 само ако сви пиксели оригиналне слике испод филтера имају вриједност 1. У супротном, ако се ради о дилатацији, пиксел ће бити подешен на 1 ако је бар један пиксел оригиналне слике испод филтера једнак 1. На тај начин се постиже сужавање објекта у случају ерозије, односно проширивање објекта у случају дилатације.

Морфолошке операције су садржане у `OpenCV` библиотеци. У оквиру овог пројекта, за морфолошке операције, коришћене су функције из библиотеке `OpenCvSharp`:

- Ерозија: `OpenCvSharp.Cv2.Erode`
- Дилатација: `OpenCvSharp.Cv2.Dilate`
- Отварање и затварање: `OpenCvSharp.Cv2.MorphologyEx`

Резултат отклањања шума са бинаризоване слике је приказан испод:

Слика 8: Отклоњен шум

Ако се упореди претходна слика са сликом 6, примјећује се да је претходна слика "прочишћена" од шума тј. од ситних контура које нису припадале објекту, и да су шупљине које су постојале по објекту сада попуњене. Такође, примјећује се да оригинална ширина профила није нарушена. То се постигло тако што су редом извршаване следеће морфолошке операције:

- Ерозија са елиптичким филтером величине 3x3
- Дилатација са елиптичким филтером величине 7x7
- Морфолошко затварање са елиптичким филтером 7x7
- Ерозија са елиптичким филтером величине 7x7

- Дилатација са елиптичким филтером величине 3x3

До претходног начина имплементације отклањања шума дошло се експериментално. Имплементација отклањања шума, у овом пројекту, налази се унутар функције:

- `TimingScanner.ScannerUtils.RemoveNoise`

4.4 Издвајање контуре профила

Ако се погледа слика након отклањања шума (слика 8), примјећује се да само отклањање шума није довољно да би на слици била само контура од интереса тј. контура профила. Разлог је што на истој дубини могу постојати и други објекти који су довољно велики да остану на слици и након одрађених морфолошких операција. На слици 8, то је случај са нежељеним објектом у доњем лијевом углу. Претпоставка је да је, по димензијама, профил највећа контура на слици. С обзиром да је профил једина контура од интереса и да на слици могу постојати и друге, мање контуре које су безначајне, као што је случај на слици 8, потребно је издвојити само контуру профила. Издвајање пролази кроз неколико корака:

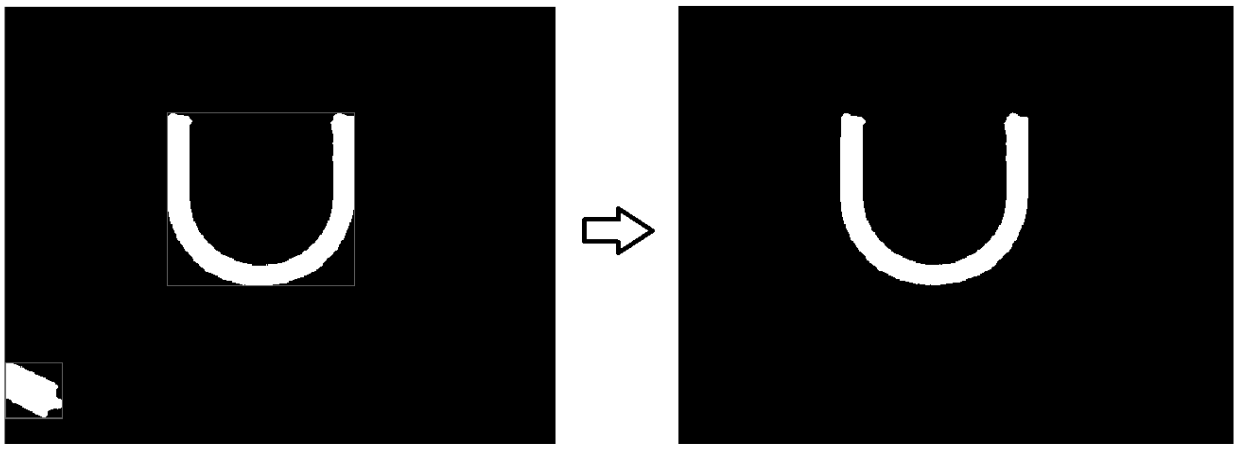
- Прво се издвајају све контуре на слици. За то користимо уграђену функцију `OpenCvSharp.Cv2.FindContours` која ради на принципу да се из улазне слике коју прима као аргумент, примјеном напредних алгоритама [1], издвоји низ вектора тачака које припадају свим контурама на слици. Сваки вектор представља вектор координата (X, Y) свих пиксела једне контуре. Алгоритам ради сасвим задовољавајуће над бинаризованим сликама.

- Пролази се кроз добијени низ контура са слике и за сваку контуру се проналази окружујући правоугаоник чије ивице додирују ивице контуре са сваке стране. Странице правоугаоника су паралелне са страницама слике. За проналажење поменутих окружујућих правоугаоника, у оквиру овог пројекта, користи се уграђена функција `OpenCvSharp.Cv2.BoundingRect` која на основу вектора тачака (координата пиксела) контуре враћа жељени правоугаоник око те контуре. Затим се провјерава да ли правоугаоник око контуре у тренутној итерацији има већу површину од правоугаоника који је до тада имао највећу површину, и ако има, поставља се за највећи правоугаоник, а индекс тренутне итерације се памти као индекс највеће контуре у низу. Овај начин омогућује да се пронађе највећа контура по димензијама, а не по броју пиксела, што је за овај пројекат пожељнији начин имплементације

- Након проналаска највеће контуре, генерише се потпуно црна слика димензија као улазна и на њу се, коришћењем уграђене функције `OpenCvSharp.Cv2.DrawContours`, преко вектора координата пиксела исцртава добијена највећа контура. Други начин би био да се само одсијече улазна слика до страница окружујућег правоугаоника највеће контуре. Међутим, тај начин није добар јер у том случају и даље можемо имати нежељену контуру или дио ње на слици, јер њен окружујући правоугаоник потенцијално може улазити у површину правоугаоника који окружује највећу контуру.

- Последњи корак је одсијецање слике на којој је само највећа контура до страница правоугаоника који ту контуру окружује, тако да ће излазна слика бити величине окружујућег правоугаоника највеће контуре.

На следећим сликама је илустративно приказан поступак издвајања контуре профила који је претходно детално описан:



Слика 9: Пронађена највећа контура на слици

Слика 10: Издвојена највећа контура (профил)

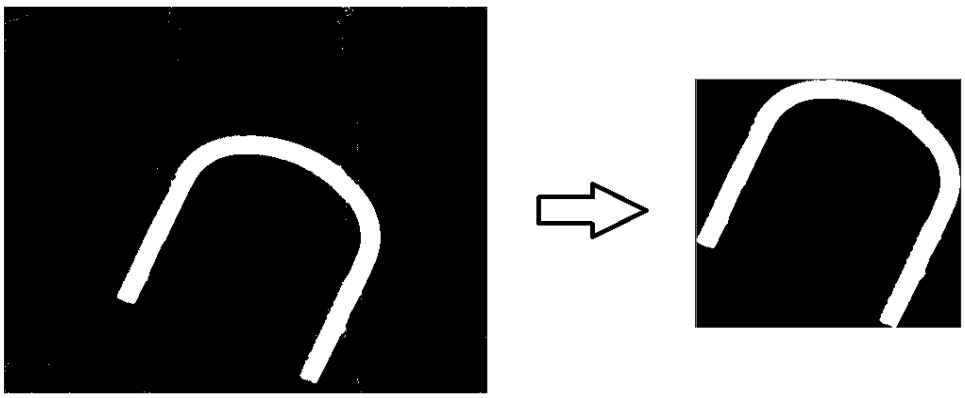
Имплементација издвајања највеће контуре (контуре профила), у овом пројекту, налази се унутар функције:

- `TimingScanner.ScannerUtils.ExtractLargestContour`

4.5 Одређивање оријентације профила

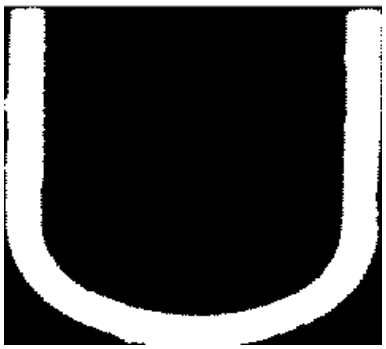
На претходним сликама профил је био окренут крацима ка горе што представља најпожељнију оријентацију профила у оквиру овог пројекта. Разлог за то је накнадно

издвајање одбирака лука пратећи удаљеност контуре од доње ивице слике. Међутим, исправна оријентација не мора увијек бити случај, јер профил може бити окренут на било коју страну. На примјер, профил може бити окренут као на следећој слици:



Слика 11: Неправилно оријентисан профил

Циљ је да се одреди тренутна оријентација профила и ротира се тако да имамо следећи резултат:



Слика 12: Правилно оријентисан профил

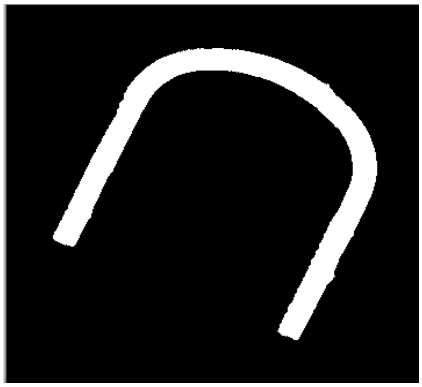
Да би се дошло до пожељно оријентисаног профила (профил окренут крацима ка горе) пролазе се редом следећи кораци:

- Скелетизација контуре
- Проналазак почетне и крајње тачке профила
- Одређивање угла на основу кога ротирамо
- Ротација за вриједност израчунатог угла
- Окретање профила крацима ка горе ако је потребно

4.5.1 Скелетизација контуре

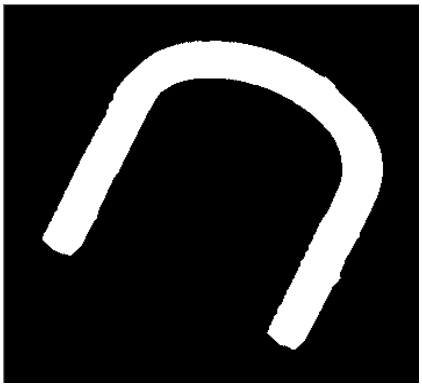
Скелетизација представља издвајање средње линије контуре. Скелетизација контуре, у овом случају, потребна је како би се пронашле почетна и завршна тачка профила, на основу којих ће се одредити оријентација профила на слици. Процес скелетизације пролази кроз следеће фазе:

- Корак 1: Простор око контуре се попуњава црним оквиром дебљине 40 пиксела са свих страна, тако да се слика повећава, а контура остаје исте величине у средини слике. Разлог што се ово ради јесте припрема за корак 2. Након корака 1 резултујућа слика има следећи изглед:



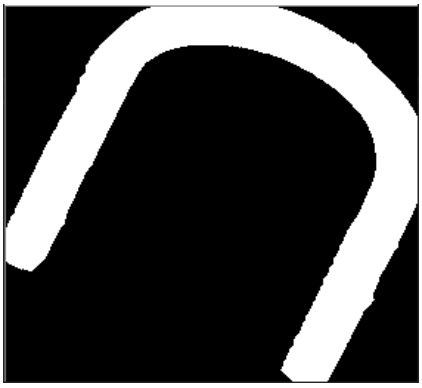
Слика 13: Слика проширена црним оквиром око контуре

- Корак 2: Проширује се контура профила морфолошком операцијом дилатације која је објашњена у поглављу које се бави отклањањем шума на слици. У овом случају, ради се проширивање са елиптичким филтером димензије 3x3 и то у седам итерација. Проширивање се сада ради како би се попуниле потенцијалне шупљине на профилу, које нису попуњене кроз процес отклањања шума. На слици 13 таквих шупљина нема, али се кроз праксу показало да их може бити и у том случају би потенцијално представљале сметњу за издвајање средње линије контуре профила. Након примјене ове операције, контура ће имати следећи изглед:



Слика 14: Проширена контура профила

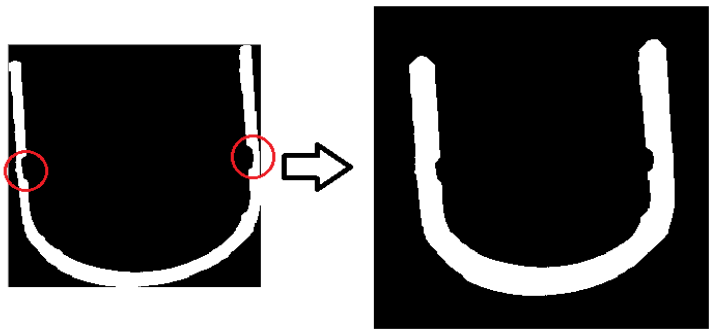
Ако се не би примјенио корак 1 (додавање црног оквира), након проширивања добио би се следећи резултат:



Слика 15: Проширена контура ако се не постави црни оквир

Са слике се види да је дио контуре профила нестао тј. облик је нарушен. Због тога би нам резултат на слици 15 био неупотребљив.

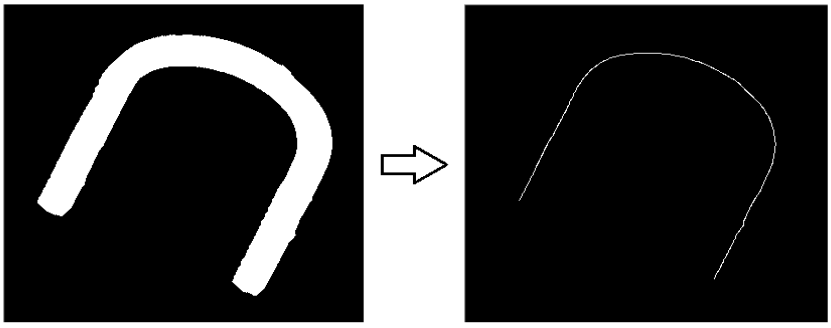
Примјер када је проширивање приликом процеса скелетизације корисно јесте следећи:



Слика 16: Случај корисног проширивања контуре

Са слике се види да на мјестима означеним црвеном бојом постоје одређени недостаци на контури, који су проширивањем прилично ублажени и представљаће мању сметњу при процесу скелетизације у следећем кораку. Такође је битно напоменути да проширивање у овој мјери занемарљиво утиче на облик лука.

- Корак 3: Вршимо процес скелетизације тј. издвајање средње линије профила. За ту операцију се користи Џанг-Сун (Zhang-Suen) алгоритам који представља брзи паралелни алгоритам за стањивање дигиталних шаблона [6]. Алгоритам се користи позивањем уграђене функције `OpenCvSharp.XImgProc.CvXImgProc.Thinning` са одговарајућим аргументом који означава да се користи поменути алгоритам. Када алгоритам примјенимо у случају примјера са слике 14, добија се следећи резултат:



Слика 17: Издвојен скелет контуре профила

У наставку је објашњење принципа на коме ради примјењени алгоритам за скелетизацију (Џанг-Сун алгоритам). Објашњење је прилагођено задатку који се ради у оквиру овог пројекта.

Нека је контура бијеле боје на слици, а подлога црне, и нека су црни и бијели пиксели означени редом са 0 и 1. Алгоритам ради над свим пикселима означеним са 1 (бијелим пикселима) који имају осам сусједних пиксела. То значи да ивични пиксели слике неће бити разматрани. Због тога се, прије примјене алгоритма, улазна слика проширује са сваке стране црним оквиром ширине једног пиксела. На тај начин смо сигурни да ће сви пиксели улазне слике бити разматрани у оквиру алгоритма.

Нека је $P1$ пиксел бијеле боје над којим алгоритам тренутно ради и нека има осам сусједних пиксела. Сусједни пиксели су организовани на следећи начин:

P9	P2	P3
P8	P1	P4
P7	P6	P5

- Дефинишимо $A(P1)$ = број прелаза са црне на бијелу боју (0 -> 1) у секвенци $P2, P3, P4, P5, P6, P7, P8, P9, P2$. Треба запазити додатно $P2$ на крају, што значи да се секвенца кружна.
- Дефинишимо $B(P1)$ = број пиксела сусједних са $P1$ који су бијеле боје (= $sum(P2, ..., P9)$).

Фаза 1
Тестирају се сви пиксели слике, и они који задовољавају следеће услове се на почетку ове фазе само означе:

1. Пиксел је бијеле боје и има осам сусједних пиксела

2. $2 <= B(P1) <= 6$
3. $A(P1) = 1$
4. Барем један од $P2$, $P4$ и $P6$ је црне боје (има вриједност 0)
5. Барем један од $P4$, $P6$ и $P8$ је црне боје (има вриједност 0)

Након проласка кроз све пикселе слике и означавања свих пиксела који задовољавају услове из фазе 1, сви означени пиксели се подешавају на 0 (постају црни).

Фаза 2
Тестирају се поново сви пиксели слике, и они који задовољавају следеће услове се на почетку ове фазе само означе:

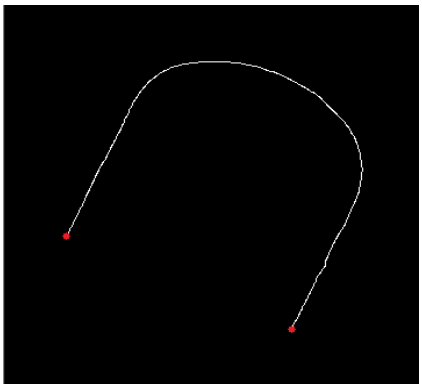
1. Пиксел је бијеле боје и има осам сусједних пиксела
2. $2 <= B(P1) <= 6$
3. $A(P1) = 1$
4. Барем један од $P2$, $P4$ и $P8$ је црне боје (има вриједност 0)
5. Барем један од $P2$, $P6$ и $P8$ је црне боје (има вриједност 0)

Након проласка кроз све пикселе слике и означавања свих пиксела који задовољавају услове из фазе 2, сви означени пиксели се подешавају на 0 (постају црни).

Фазе 1 и 2 се понављају све док не престане да постоји било каква промјена у вриједности пиксела слике.

4.5.2 Проналазак почетне и крајње тачке профила

Процес скелетизације из претходног корака одрађен је у сврху лакшег проналаска почетне и крајње тачке средње линије контуре профила. Тачке које тражимо приказане су на следећој слици:



Слика 18: Тачке почетка и краја средње линије контуре

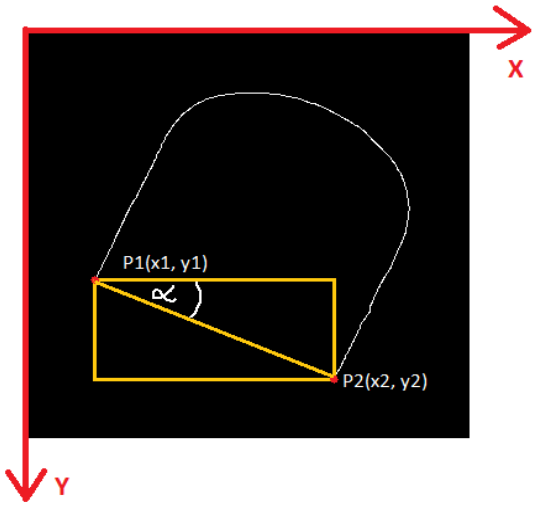
Да бисмо дошли до жељених тачака, пролази се кроз све пикселе слике која је резултат скелетизације. Нека је тренутни пиксел означен са p. За пиксел p се врши провјера да ли је бијеле боје (да ли је пиксел контуре). Ако јесте, провјеравају се сви његови сусједни пиксели. Ако има тачно један сусједни пиксел бијеле боје, то значи да је пиксел p крајња тачка средње линије контуре. По правилу, ако је скелетизација дала очекивани резултат, постојаће тачно двије крајње тачке средње линије контуре профила. У оквиру пројекта, ова операција је имплементирана унутар функције:

- `TimingScanner.ScannerUtils.GetContourPoints`

Примарна намјена функције јесте да се издвоје све тачке средње линије контуре профила, али као излазни параметар има и промјенљиву у којој ће бити садржане само крајње тачке средње линије контуре. Треба напоменути да ако издвојимо све тачке средње линије контуре профила користећи ову функцију, те тачке неће бити распоређене тако да иду редом онако како граде линију контуре. За одређивање оријентације профила довољне су нам само крајње тачке средње линије профила. Разлог зашто је обоје имплементирано у једној функцији јесте тај што за оба излаза пролазимо кроз исту петљу, а на неким мјестима су нам потребна оба излаза (и све тачке и крајње тачке линије), па се у том случају удвостручи брзина обављања обје операције.

4.5.3 Одређивање угла на основу кога се ротира слика

Када су познате крајње тачке средње линије контуре, потребно је одредити угао (нагиб ка X оси) праве која пролази кроз те двије тачке. Нека су тачке означене са $P1$ и $P2$, а угао са α. На следећој слици је детаљан приказ позиција тачака и угла, као и усвојеног координатног система:



Слика 19: Позиција тачака и угла за ротацију

Алгоритам за одређивање угла α пролази редом кроз три услова:

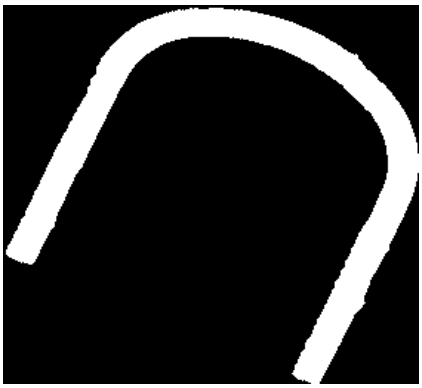
- Ако је $x1 = x2$, тада је $\alpha = 90°$

- Ако је $y1 = y2$, тада је $\alpha = 0^\circ$
- Ако не важи ни један од претходна два услова, тада је $\alpha = \arctan\left(\frac{y1-y2}{x1-x2}\right)$

Ако права коју формирају тачке $P1$ и $P2$ заклапа оштар угао са X-осом, тада ће α имати позитивну вриједност, а ако заклапа туп угао тада ће α имати негативну вриједност.

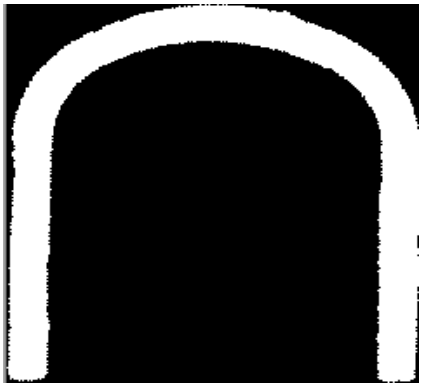
4.5.4 Ротација за вриједност израчунатог угла

Када је угао α израчунат, потребно је улазну слику, коју имамо на почетку процеса одређивања оријентације контуре, ротирати за израчунати угао α.



Слика 20: Улазна слика у процес одређивања оријентације профила

Када претходну слику ротирамо за добијени угао α, добија се следећи резултат:



Слика 21: Резултат након ротације за угао α

Имплементација приказане ротације, у овом пројекту се налази унутар функције:

- `TimingScanner.ScannerUtils.MatRotate`

Поступак ротације пролази кроз следеће кораке:

1. Рачуна се ротациона матрица за дводимензионалну слику преко уграђене функције `OpenCvSharp.Cv2.GetRotationMatrix2D`. Функцији се прослијеђују координате центра слике, вриједност угла и скалирајући фактор (*scale*) који ће у случају саме ротације слике бити 1. Излазна матрица има следећи облик:

$$\begin{bmatrix} a & b & (1-a)\cdot c_x - b\cdot c_y \\ -b & a & b\cdot c_x + (1-a)\cdot c_y \end{bmatrix}$$

гдје су:

- $a = scale \cdot \cos(\alpha)$
- $b = scale \cdot \sin(\alpha)$
- c_x, c_y - координате центра слике

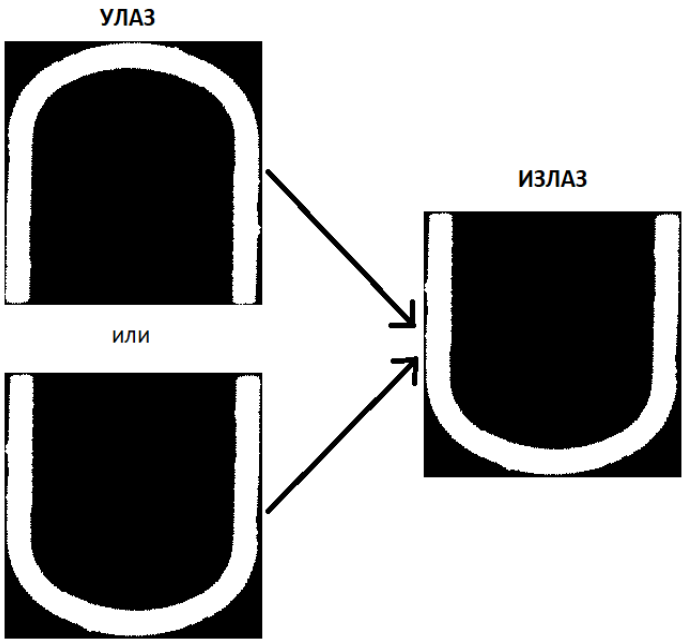
Ако угао α има позитивну вриједност, ротација ће се обављати у смјеру супротном кретању казаљке на сату, а ако угао α има негативну вриједност, ротација ће се обављати у смјеру кретања казаљке на сату.

2. Потребно је одредити које димензије ће имати излазна слика након ротације за угао α, и у зависности од тога, помјерити центар ротације. То се ради тако што се правоугаоник $R1$, који има димензије улазне слике, заротира за угао α и затим се око њега опише правоугаоник $R2$ са страницама паралелним X и Y осама. Димензије описаног правоугаоника $R2$ биће димензије излазне слике након ротације. Центар ротације у матрици ротације се помјера за половину разлике ширина правоугаоника $R2$ и $R1$ ($\frac{R2.Width - R1.Width}{2}$) по X-оси, и за половину разлике висина правоугаоника $R2$ и $R1$ ($\frac{R2.Height - R1.Height}{2}$) по Y-оси. Одређивање димензија излазне слике и помјерање центра ротације спрјечава потенцијални губитак информација о контурама са улазне слике приликом ротације.

3. Последњи корак јесте ротација улазне слике на основу матрице ротације и израчунатих димензија излазне слике. За ту операцију користи се уграђена функција `OpenCvSharp.Cv2.WarpAffine` која се иначе користи за трансформације дигиталних слика. У овом случају се користи за ротацију (линеарна трансформација).

4.5.5 Окретање профила крацима ка горе ако је потребно

Резултат ротације из претходног корака не гарантује да ће на излазној слици профил бити окренут крацима ка горе или ка доле. Због тога је потребно одредити у ком положају се налази профил након прве ротације. Могућа су два положаја: окренут крацима ка доле и крацима ка горе. Циљ је да профил буде окренут крацима ка горе:



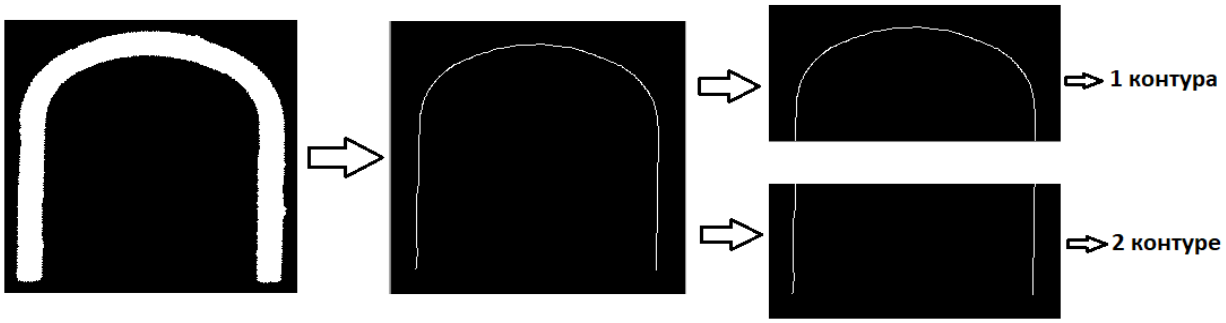
Слика 22: Коначно исправљање положаја профила

Принцип на коме ради крајње исправљање положаја профила на слици је следећи:

1. Додаје се црни оквир дебљине 10 пиксела на улазну слику
2. Врши се процес скелетизације (Цанг-Сун алгоритам)
3. Раздвајају се горња и доња половина слике
4. Одређује се број контура на горњем и на доњем дијелу примјеном уграђене `OpenCvSharp.Cv2.FindContours` која је већ поменута.
5. Упоређује се број контура у доњем и горњем дијелу. По правилу, у овој фази, требало би да дио слике у коме се завршавају краци профила има двије контуре, а дио који садржи већи дио лука да има једну контуру. Ако су у доњем дијелу двије контуре, а у горњем једна, тада је потребно ротирати слику за 180° тј. окренути краке ка горе. Ако је у доњем дијелу једна контура, а у горњем двије, то значи да је профил већ исправно оријентисан па се задржава улазна слика и на излазу (не ради се ротација).

Ако је потребна, ротација за 180°, у оквиру овог пројкта, се ради примјеном функције `TimingScanner.ScannerUtils.MatRotate`, чији је принцип рада већ описан.

Претходне операције могу се сликовито приказати на следећи начин:



Слика 23: Процес коначног исправљања положаја профила

Читава описана операција окретања профила крацима ка горе имплементирана је унутар функције `TimingScanner.ScannerUtils.TurnUpwards`

4.6 Свођење на фиксну ширину слике

Након одређивања оријентације профила, потребно је свести слику на фиксну ширину. У оквиру овог задатка, усвојено је да се сведе на ширину од 300 пиксела, што значи да ћемо у наредном кораку имати 300 одбирака који представљају лук. Разлог за свођење на фиксну ширину јесте што величина лукова може да буде врло промјенљива. То значи да ћемо за "мале" или удаљене лукове имати прилично мало одбирака на основу којих одређујемо облик лука. С друге стране, ако је улазна слика шира од 300 пиксела, број одбирака биће смањен на 300, али се експериментално показало да је 300 одбирака сасвим довољно за приично прецизну класификацију. Битно је напоменути и да фиксна ширина омогућава да после, у току класификације, толеранције одступања одређених вриједности од претпостављених могу бити примјенљиве на све величине лукова. О толеранцијама ће више бити ријечи у опису самог процеса класификације лукова.

Слика 24: Свођење на фиксну ширину профила

Свођење на фиксну ширину је имплементирано у оквиру функције:

- `TimingScanner.ScannerUtils.ScaledResize`

Функција као аргументе прима улазну слику и ширину коју ће имати излазна слика. На основу тих података потрбно је израчунати висину коју ће имати излазна слика, а то се ради коришћењем пропорције. Нека је `inputImage` улазна слика, а `outputImage` излазна слика. Тада треба да буде задовољено:

$$\frac{outputImage.Height}{outputImage.Width} = \frac{inputImage.Height}{inputImage.Width}$$

Одатле можемо израчунати висину излазне слике као:

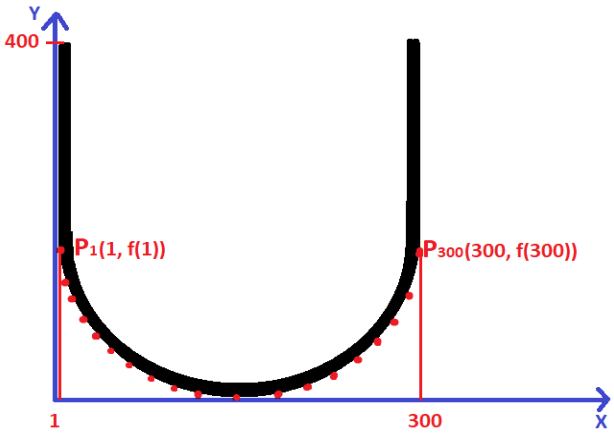
$$outputImage.Height = \frac{inputImage.Height \cdot outputImage.Width}{inputImage.Width}$$

Сада, када су познате и висина и ширина излазне слике, потребно је формирати излазну слику примјеном уграђене функције `OpenCvSharp.Mat.Resize` над улазном сликом.

Треба напоменути да се свођење може радити и на већу ширину, али то касније смањује рачунску ефикасност алгоритма за класификацију, а не даје бољи резултат сразмјерно повећању ширине слике.

4.7 Издвајање одбирака лука и њихово усредњавање

Да бисмо издвојили одбирке лука потребно је окренути лук крацима ка горе и координатни систем се поставља у доњи лијеви угао издвојене слике:



Слика 25: Принцип издвајања одбирака лука

Ширина улазне слике је 300 пиксела и толико имамо одбирака. Одбирци по X-оси су цјелобројне вриједности од 1 до 300, а по Y-оси су израчунате вриједности на основу облика лука полазећи с лијеве стране матрице слике на десно. Тачке $P_{1...300}$ представљају координате спољашње ивице лука. Битно је напоменути да је препорука да се увијек прати спољашња ивица профила јер садржи више пиксела лука, а самим тим и више информација о облику лука.

Ако издвајање одбирака лука на описани начин примјенимо на примјерку профила са слике:

Слика 26: Профил за издвајање одбирака лука

добијају се одбирци који имају следећи облик:

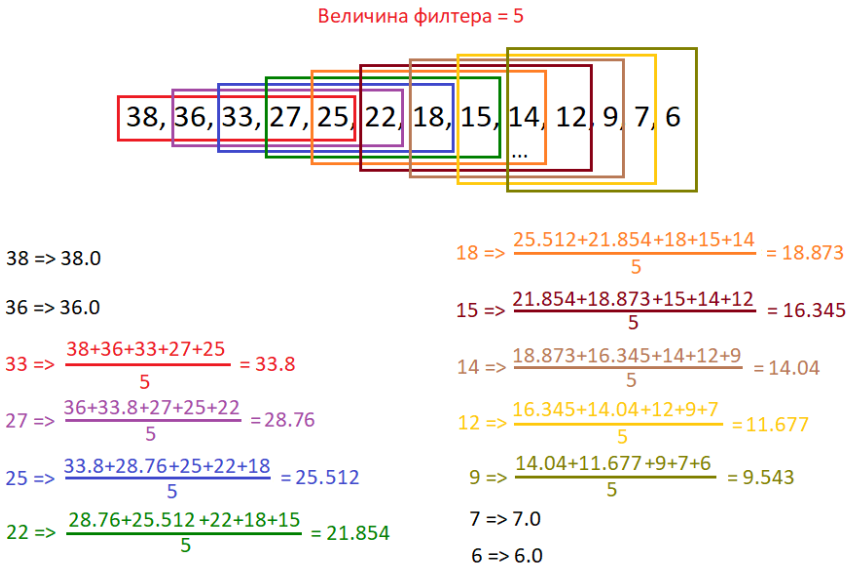
138, 122, 122, 114, 107, 107, 102, 102, 95, 92, 91, 88, 88, 86, 86, 79, 76, 76,
74, 74, 71, 71, 69, 67, 67, 64, 64, 62, 59, 59, 57, 57, 55, 55, 54, 52, 52, 50, 50,
48, 48, 45, 42, 41, 40, 40, 38, 38, 38, 36, 36, 35, 35, 33, 31, 31, 29, 29, 28, 26,
26, 24, 24, 24, 24, 24, 23, 23, 23, 23, 21, 21, 21, 19, 19, 19, 19, 17, 17, 17, 16,
16, 16, 16, 14, 14, 14, 12, 12, 10, 10, 10, 10, 10, 9, 9, 9, 9, 9, 9, 9, 9, 7, 7, 7, 7,
7, 7, 6, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 7, 7, 7, 7, 9, 9, 9, 9, 10, 10, 10,
12, 12, 12, 12, 12, 14, 14, 14, 14, 16, 16, 16, 16, 16, 16, 16, 17, 17, 17, 19, 19,
19, 19, 21, 21, 21, 23, 23, 24, 24, 24, 26, 26, 26, 26, 28, 28, 28, 29, 29, 31, 31,
33, 35, 35, 38, 38, 40, 40, 41, 43, 43, 45, 45, 47, 48, 48, 50, 50, 52, 52, 54, 55,
55, 57, 57, 59, 59, 60, 64, 64, 69, 69, 71, 74, 74, 76, 76, 79, 79, 83, 88, 88, 91,
91, 95, 95, 100, 105, 105, 110, 110, 117, 126, 126, 145,

Слика 27: Одбирци лука

Добијање претходних одбирака из слике на којој је контура профила окренута крацима ка горе је имплементирано унутар функције

- `TimingScanner.ScannerUtils.GetArrayY`

Да бисмо од претходних цјелобројних одбирака са слике 27 добили усредњене одбирке који прецизније описују облик лука, примјењује се метод замјене одбирака са просјечном вриједношћу његових околних одбирака. Операција се врши проласком једнодимензионалног филтера, непарне дужине веће или једнаке 3, кроз низ одбирака са слике изнад, и то по следећем принципу:



Слика 28: Принцип усредњавања одбирака лука

Нека је n величина (дужина) филтера. Првих и последњих $\frac{n-1}{2}$ вриједности у низу остају непромјењене. Дужина филтера мора бити непаран број већи или једнак 3. Имплементација описаног принципа усредњавања налази се унутар функције:

- `TimingScanner.ScannerUtils.MovingAverageSmooth1D`

Усредњавање одбирака лука се, у оквиру овог пројекта, врши у двије итерације. У првој се усредњавају одбирци коришћењем филтера дужине 5, а у другој коришћењем филтера дужине 3. Ако овај метод примјенимо на одбирке са слике 27, добијају се следећи одбирци:

```
138, 126.8667, 120.5289, 114.931, 110.7093, 107.0005, 103.3661, 99.8147, 96.48328, 93.74197, 91.47251, 89.65206, 87.82796, 85.49702, 82.79867,
80.17036, 78.09755, 76.35483, 74.79089, 73.32928, 71.85741, 70.44847, 68.99822, 67.60168, 66.2241, 64.7332, 63.18081, 61.57925, 60.15083, 58.87896,
57.71693, 56.6895, 55.6714, 54.66888, 53.60029, 52.56458, 51.53656, 50.52142, 49.44533, 48.07126, 46.39595, 44.59099, 43.0299, 41.73502, 40.62326,
39.69508, 38.87148, 38.11591, 37.33168, 36.60092, 35.89032, 34.99951, 33.94505, 32.77794, 31.67428, 30.667, 29.66044, 28.66288, 27.59713, 26.56283,
25.66893, 24.992, 24.58073, 24.26406, 23.95213, 23.63763, 23.38275, 23.09234, 22.65777, 22.13761, 21.55494, 20.93713, 20.29795, 19.7802, 19.32669,
18.79173, 18.2138, 17.66448, 17.19678, 16.77929, 16.46423, 16.13837, 15.68374, 15.15217, 14.5631, 13.94169, 13.16717, 12.31051, 11.52512, 10.91032,
10.53482, 10.23827, 9.937698, 9.629554, 9.378238, 9.223155, 9.127477, 9.072526, 9.040801, 9.022931, 8.879501, 8.536072, 8.06831, 7.649275, 7.386317,
7.221385, 7.059629, 6.768899, 6.336605, 5.878067, 5.52859, 5.309883, 5.110545, 4.865036, 4.588766, 4.35531, 4.210332, 3.986967, 3.597406, 3.102857,
2.668713, 2.397201, 2.227479, 2.129701, 2.073025, 2.041075, 2.022996, 2.012873, 2.007193, 2.00402, 2.002245, 2.001253, 1.934033, 1.764835, 1.532366,
1.323638, 1.1926, 1.110381, 1.062974, 1.035464, 1.019952, 1.011171, 1.006254, 1.003495, 1.001953, 1.001091, 1.000609, 1.000291, 1.000097, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1.066667, 1.235556, 1.467852, 1.676484, 1.807468, 1.889657, 1.937047,
1.964548, 1.980055, 2.122166, 2.464859, 2.93221, 3.351015, 3.613846, 3.778706, 3.873755, 3.928907, 3.960004, 3.977606, 4.05413, 4.22855, 4.463937,
4.674298, 4.806247, 4.888976, 4.936666, 4.964336, 4.979936, 4.988766, 4.993711, 4.996486, 5.131369, 5.470015, 5.935091, 6.352626, 6.748079, 7.250319,
7.809739, 8.282031, 8.575027, 8.823636, 9.097141, 9.389958, 9.766021, 10.25406, 10.81164, 11.28235, 11.5752, 11.89031, 12.3327, 12.8578, 13.30917,
13.72375, 14.2367, 14.80213, 15.27778, 15.57265, 15.75564, 15.86084, 15.98836, 16.19153, 16.44321, 16.79602, 17.27087, 17.82105, 18.28761, 18.71148,
19.22973, 19.79821, 20.40891, 21.04253, 21.75733, 22.44898, 23.00427, 23.54498, 24.12796, 24.74063, 25.2424, 25.68616, 26.21557, 26.7903, 27.33782,
27.80451, 28.35476, 29.00729, 29.79755, 30.80268, 31.93395, 33.21863, 34.52095, 35.84473, 37.1054, 38.20078, 39.26482, 40.29432, 41.37889, 42.42366,
43.45685, 44.54155, 45.58817, 46.6221, 47.57411, 48.54993, 49.52837, 50.51685, 51.57611, 52.60762, 53.63309, 54.58025, 55.55337, 56.53029, 57.45125,
58.54115, 59.84676, 61.56034, 63.50901, 65.48973, 67.49171, 69.36028, 71.15839, 72.69973, 74.10363, 75.47134, 76.95188, 78.89693, 81.24052, 83.78442,
86.0573, 88.11912, 90.08554, 92.12435, 94.60312, 97.39921, 100.4184, 103.2361, 106.0884, 109.6508, 113.8765, 119.3483, 123.7147, 131.5716, 145,
```

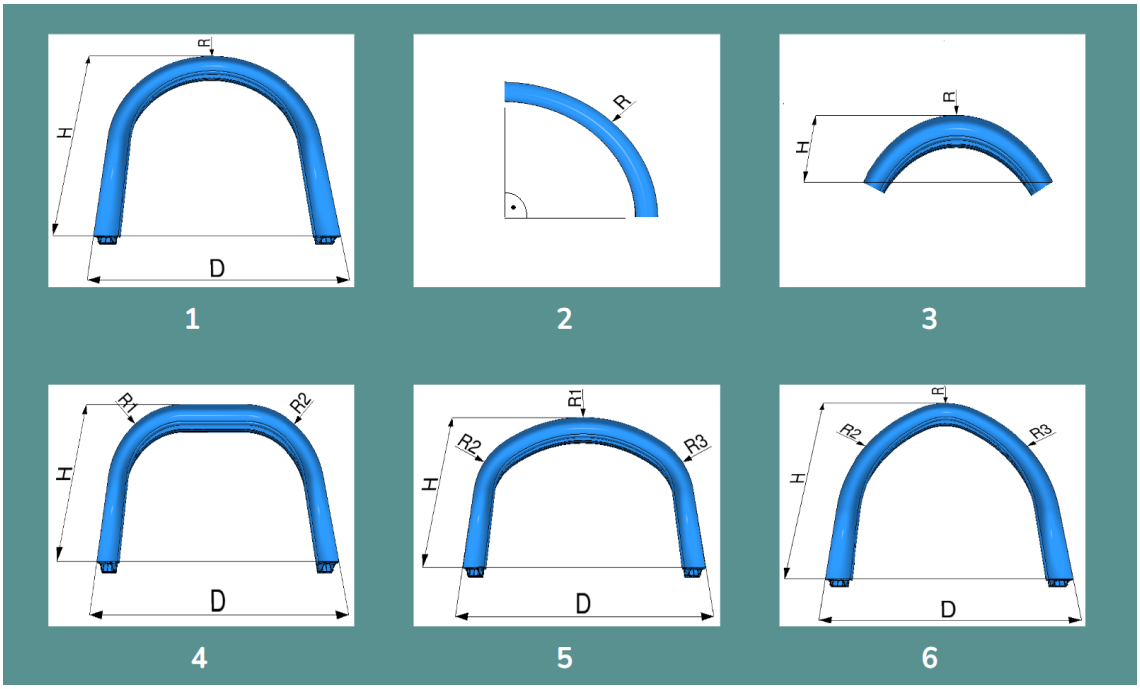
Слика 29: Усредњени одбирци лука

Процес издвајања одбирака и њиховог усредњавања, само на основу улазне слике и према поменутим правилима, имплементиран је у оквиру функције:

- `TimingScanner.BendClassifier.GetAverageSamplesArray`

5 Процес класификације

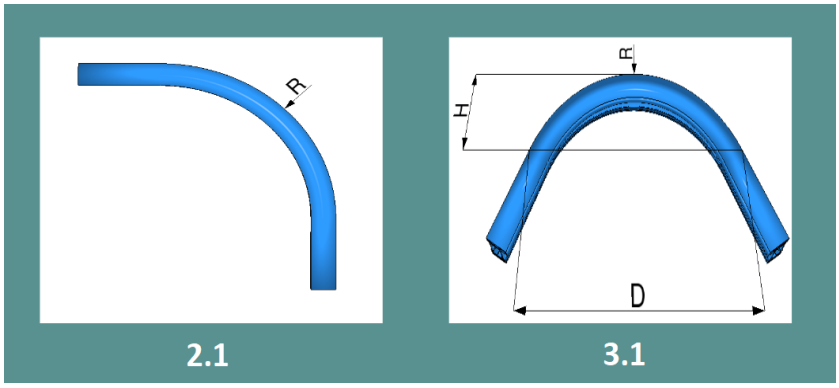
Процес класификације највише се заснива на усредњеним одбирцима добијеним на начин описан у претходном поглављу. У овом поглављу биће описан поступак класификације заснован на подацима од једног фрејма (слике) који је издвојен из снимка добијеног са 3D камере. Класификација се одвија методом елиминације. То значи да ако је први услов задовољен, не иде се даље са провјером, а ако први услов није задовољен редом се провјеравају услови све док неки не буде задовољен, а могуће је да ни један услов не буде задовољен што значи да је облик непознат. За потребе једноставнијег означавања, класе лукова су означене бројевима, а присјећања ради, облици из поставке задатка су приказани и на слици испод:



Слика 30: Класе профила

Битно је напоменути да L - лук и исјечак (класе 2 и 3), када су окренути крацима ка горе, међу одбирцима лука могу имати и одбирке равних дијелова крака профила, што представља отежавајућу околност јер те равне дијелове треба отклонити како би одбирци представљали само лук. Стога се усваја да L - лук и исјечак без равног дијела представљају класе 2 и 3, а L - лук и исјечак са равним дијеловима да представљају

класе 2.1 и 3.1 и тако ће бити означене у наставку описа процеса класификације. Примјер тих подкласа је на слици испод:



Слика 31: Подкласе класа профила 2 и 3

Поступак елиминације се обавља следећим редослиједом:

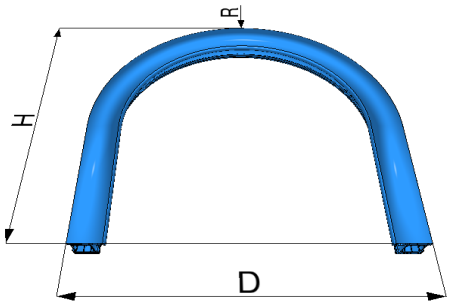
1. Провјера за правилан лук (заклапа 180°) - класа 1
2. Провјера за L лук (заклапа 90°) - класа 2
3. Провјера за исјечак (кружни исјечак који заклапа угао мањи од 180°, при чему различито од 90°) - класа 3
4. Провјера за n - лук (заклапа два угла по 90°) - класа 4
5. Провјера за хоризонталну и вертикалну елипсу - класе 5 и 6
6. Провјера за L лук и кружни исјечак са равним крацима - класе 2.1 и 3.1

Функција у којој је смјештена претходно описана логика класификације је:

- `TimingScanner.BendClassifier.Classify`

Услови које је потребно задовољити у сваком од претходно наведених корака описани су детаљно у наставку. Улазни податак у све функције за провјеру јесте усредњени низ одбирака лука који добијамо поступком описаним у претходном поглављу, а у неким случајевима и матрица слике правилно оријентисаног профила. Провјере се врше на принципу претпоставки да у току провјере на слици јесте лук који припада класи за коју се тренутно врши провјера.

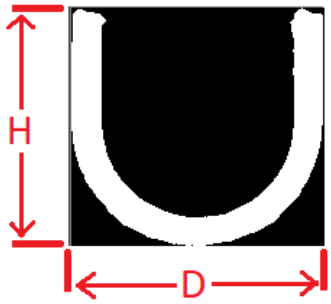
5.1 Провјера за правилан лук - заклапа 180°



Слика 32: Правилан лук - класа 1

Провјера за правилан лук врши се кроз више корака:

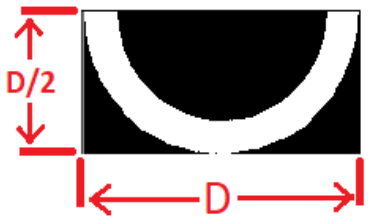
1. Посматра се слика профила окренутог крацима ка горе као на следећој слици:



Слика 33: Правилан лук - положај за одбирке 1

Ако је половина ширине претходне слике већа од висине слике са толеранцијом од пет пиксела ($\frac{D}{2} - H > 5$), у том случају лук не може бити правилан и не иде се на корак 2. Ако претходни услов није испуњен прелази се на корак 2.

2. Ако је висина слике већа од половине ширине слике ($H > \frac{D}{2}$), тада се слика по висини одсијеца тако да остане само доњи дио који има висину која је једнака половини ширине слике као на следећем приказу:

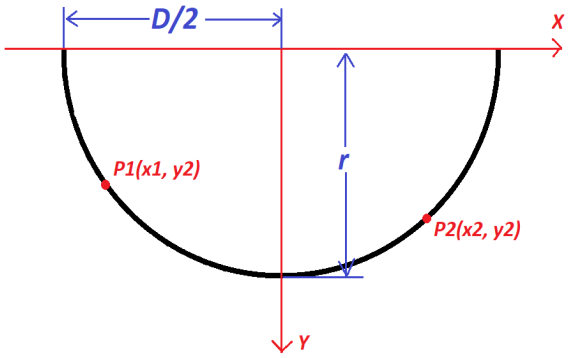


Слика 34: Правилан лук - положај за одбирке 2

Сада се врши издвајање и усредњавање одбирака лука са слике 34. За такве усредњене одбирке врши се повјера да ли припадају кружници полупречника који је једнак половини ширине улазне слике (слика 33). Провјера се врши уз дозвољено укупно одступање које износи 15 (5% од 300) узастопних одбирака који одступају за вриједност већу од 3.5 (пиксела) од стварних вриједности одбирака кружнице. Ако се дозвољено одступање прекорачи, профил са слике не испуњава услов да буде правилан лук. Математички, провјерава се да ли је задовољена једначина кружнице која гласи:

$$x^2 + y^2 = r^2$$

Нека је i индекс одбирка у низу и креће од 0, D ширина слике из које се издвајају одбирци (једнака броју одбирака), $P1$, $P2$ су неке тачке ивице лука, а r вриједност полупречника кружнице којој провјеравамо припадност. Нека је координатни почетак смјештен на средини ширине слике, удаљен за вриједност r од доње ивице слике:



Слика 35: Правилан лук - провјера 2 - скица

Може се написати следеће:

- Ако је одбирак у првој половини низа одбирака (тачка $P1$), тада важи:

$$x_1^2 = \left(\frac{D}{2} - i\right)^2$$

$$y_1^2 = (r - odbirci[i])^2$$

- Ако је одбирак у другој половини низа одбирака (тачка $P2$), тада важи:

$$x_2^2 = \left(i - \frac{D}{2}\right)^2$$

$$y_2^2 = (r - odbirci[i])^2$$

Ако је дозвољено одступање једног одбирка 3.5, за било које x и y се провјерава следећи услов припадности кружници:

$$|\sqrt{x^2 + y^2} - r| < 3.5$$

Ако више од 15 узастопних вриједности (x, y) не задовољи претходни услов, лук се не класификује као правилан. Функција која је, у овом пројекту, задужена за провјеру да ли одбирци неког лука задовољавају једначину кружнице са дефинисаним полупречником и вриједностима дозвољених одступања је:

- `TimingScanner.BendClassifier.IsSection`

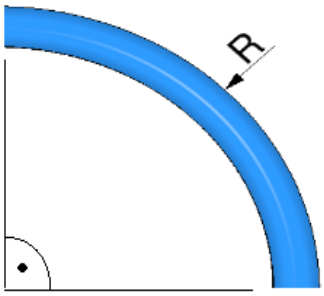
Функција је веома битна и за остатак класификације, јер се користи и у провјерама у наставку.

3. У овај корак се улази ако половина ширине улазне слике и њена висина нису ни у једном од односа из корака 1 и 2. У овом случају не врши се никакво одсијецање слике. Одбирци над којима се врши провјера су усредњени одбирци са улазне слике. Даља провјера је идентична као у кораку 2.

Провјера да ли је лук правилан је имплементирана унутар функције:

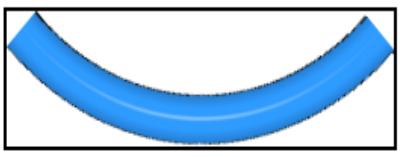
- `TimingScanner.BendClassifier.IsRegularArc`

5.2 Провјера за L лук - заклапа 90°



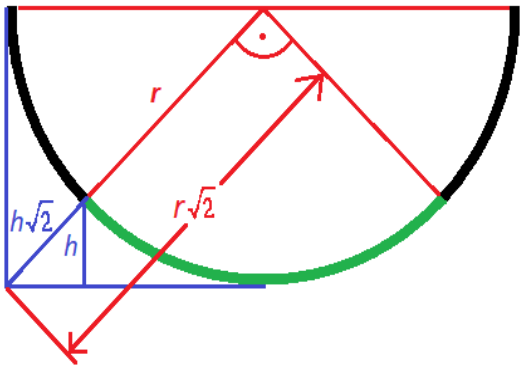
Слика 36: L лук - класа 2

Провјера се врши тако што се издвоје и усредње одбирци лука са профила у следећем положају:



Слика 37: L лук - положај за одбирке

Након тога се само врши провјера да ли одбирци припадају кружници на истом принципу као за правилан лук позивањем функције `TimingScanner.BendClassifier.IsSection`. Међутим, сада не можемо тако брзо рећи која је вриједност полупречника r кружнице са којом ће се упоређивати одбирци. Потребно је математички одредити вриједност полупречника на основу усредњених одбирака и доступних података са улазне слике. Нека је на следећој слици приказан зеленом бојом дио кружнице који представља L лук:



Слика 38: L лук - одеђивање полупречника

Податак који је познат и на основу кога одређујемо полупречник јесте вриједност h и једнака је вриједности првог (или последњег) одбирка у низу. До вриједности r на основу h долазимо на следећи начин:

$$h\sqrt{2} = r\sqrt{2} - r$$

$$r(\sqrt{2} - 1) = h\sqrt{2}$$

$$r = \frac{h\sqrt{2}}{\sqrt{2}-1}$$

$$r = \frac{h\sqrt{2}}{\sqrt{2}-1} \cdot \frac{\sqrt{2}+1}{\sqrt{2}+1}$$

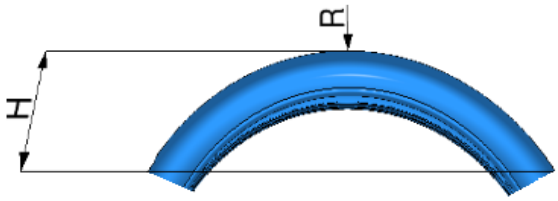
$$r = \frac{h\sqrt{2}(\sqrt{2}+1)}{\sqrt{2}^2-1^2}$$

$$r = 2h + h\sqrt{2}$$

Провјера се врши уз дозвољено укупно одступање које износи 15 (5% од 300) узастопних одбирака који одступају за вриједност већу од 3.3 (пиксела) од стварних вриједности одбирака дијела кружнице са израчунатим полупречником. Ако се дозвољено одступање прекорачи, профил са слике не испуњава услов да буде L лук. Функција која се позива при провјери да ли се задовољавају услови за L лук (класу 2) је:

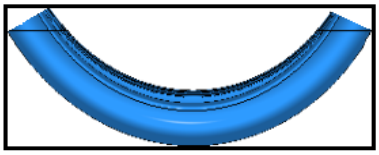
- `TimingScanner.BendClassifier.IsArcL`

5.3 Провјера за кружни исјечак - заклапа угао мањи од 180°, при чему различито од 90°



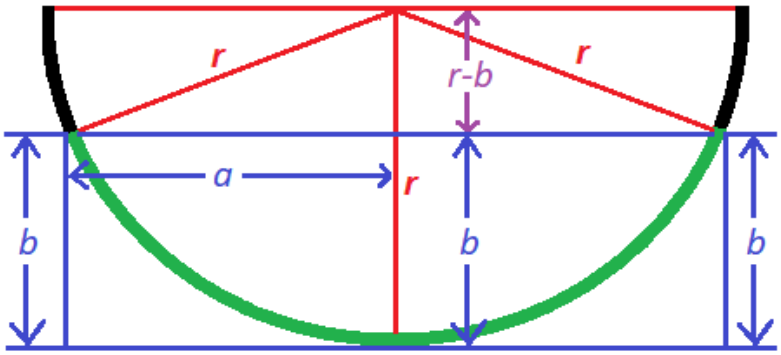
Слика 39: Кружни исјечак - класа 3

Провјера се врши тако што се издвоје и усредње одбирци лука са профила у следећем положају:



Слика 40: Кружни исјечак - положај за одбирке

Након тога се само врши провјера да ли одбирци припадају кружници на истом принципу као за правилан лук и L лук позивањем функције `TimingScanner.BendClassifier.IsSection`. Међутим, ни сада не можемо тако брзо рећи која је вриједност полупречника r кружнице са којом ће се упоређивати одбирци. Потребно је математички одредити вриједност полупречника на основу усредњених одбирака и доступних података са улазне слике. Нека је на следећој слици приказан зеленом бојом дио кружнице који представља неки кружни исјечак:



Слика 41: Кружни исјечак - одређивање полупречника

Подаци који су познати и на основу којих одређујемо полупречник јесу вриједност b (вриједност првог (или последњег) одбирка у низу) и a (вриједност половине ширине улазне слике тј. половине дужине низа одбирака). До вриједности r на основу a и b долазимо на следећи начин:

$$r^2 = a^2 + (r - b)^2$$

$$r^2 = a^2 + r^2 - 2rb + b^2$$

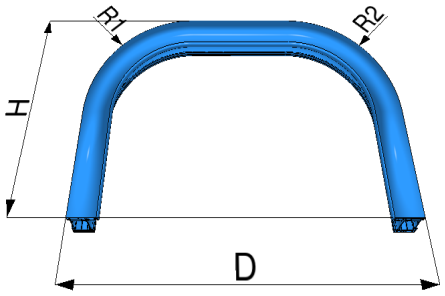
$$2br = a^2 + b^2$$

$$r = \frac{a^2+b^2}{2b}$$

Провјера се врши уз дозвољено укупно одступање које износи 15 (5% од 300) узастопних одбирака који одступају за вриједност већу од 3.3 (пиксела) од стварних вриједности одбирака дијела кружнице са израчунатим полупречником. Ако се дозвољено одступање прекорачи, профил са слике не испуњава услов да буде кружни исјечак. Функција која се позива при провјери да ли се задовољавају услови за кружни исјечак (калсу 3) је:

- `TimingScanner.BendClassifier.IsAnySection`

5.4 Провјера за n - лук (заклапа два угла по 90°)

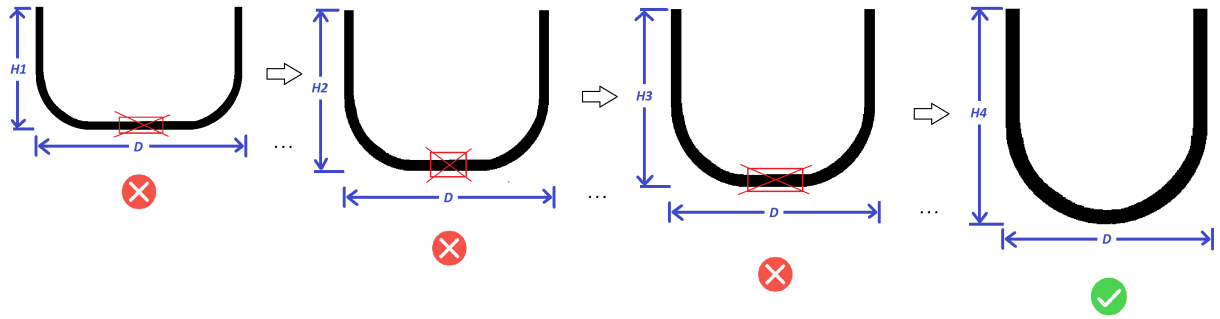


Слика 42: n лук - класа 4

Очекивана улазна слика у процес провјере има следећи облик:

Слика 43: n лук - почетни положај

Принцип на коме се заснива провјера јесте сужавање контуре профила одсијецањем средине у којој се очекивано налази раван дио. Ако се одсијецањем дође до правилног лука, значи да је на улазу профил који је савијен у n - лук. Ако се процес итерације заврши без препознатог правилног лука, тада се сматра да на слици није n - лук, и наставља се даље са провјером припадности другим класама лукова. У свакој итерацији се отклањају по двије колоне пиксела у средини тренутне слике а остаци слике се споје. Процес се прекида или кад се препозна правилан лук (са дозвољеним одступањем), или када удаљеност тренутне контуре од доње ивице слике пређе вриједност од осам пиксела. На следећој слици је приказан дио процеса провјере у коме је препознат правилан лук након одређеног броја итерација:



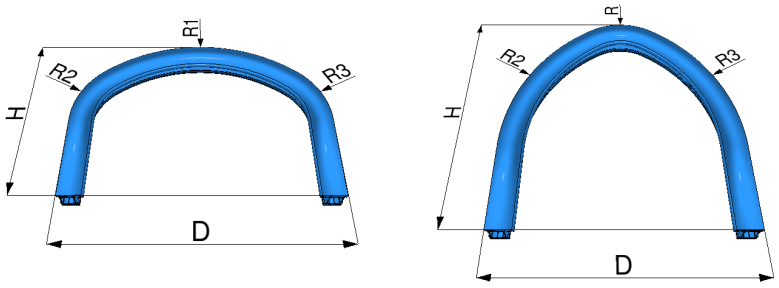
Слика 44: n лук - дио процеса провјере

Треба уочити да је ширина слике увијек једнака ширини улазне слике кроз итерације, док је висина промјењљива. На тај начин се у свакој итерацији врши провјера за једнак број одбирака лука. Имплементација провјере за n - лук је смештена унутар функције:

- `TimingScanner.BendClassifier.IsArcN`

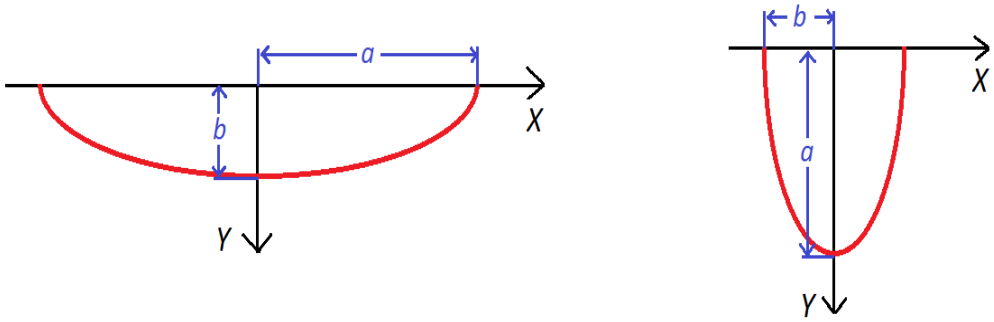
С обзиром да се врши провјера за правилан лук у свакој итерацији, унутар претходне функције се позива већ поменута функција за провјеру припадности одбирака правилном луку `TimingScanner.BendClassifier.IsRegularArc`.

5.5 Провјера за хоризонталну и вертикалну елипсу



Слика 45: Хоризонтална и вертикална елипса - класе 5 и 6

Провјера се заснива на томе да ли усредњени одбирци лука, издвојени са слике на којој је профил окренут крацима ка горе, задовољавају једначину елипсе са дефинисам дозвољеним одступањем. Нека имамо профил окренут крацима ка горе и нека је координатни систем постављен као на следећој слици:



Слика 46: Хоризонтална и вертикална елипса у координатном систему

Тада ће за хоризонталну елипсу важити следећа једначина у канонском облику:

$$b^2x^2 + a^2y^2 = a^2b^2,$$

док ће за вертикалну елипсу важити:

$$a^2x^2 + b^2y^2 = a^2b^2$$

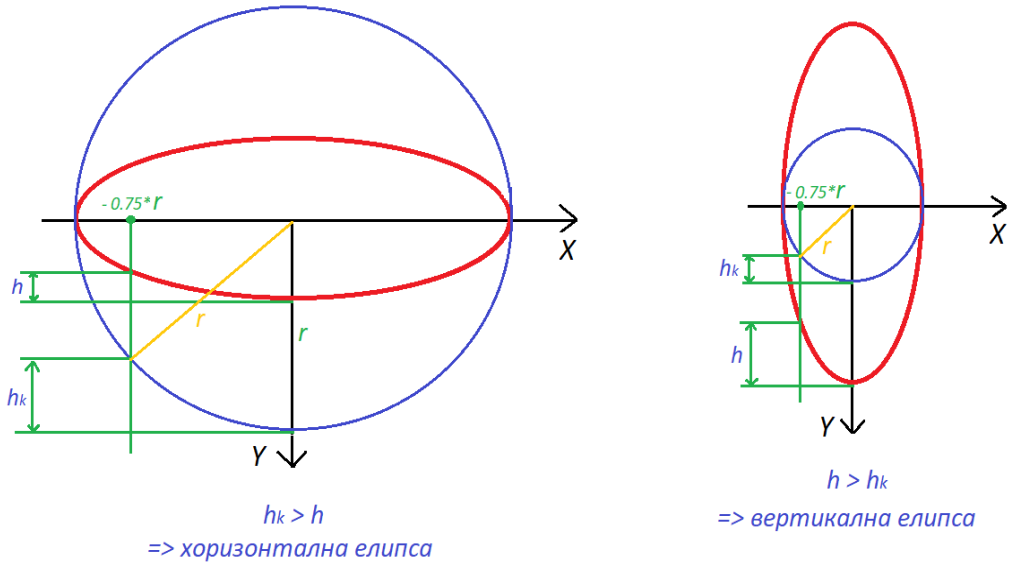
Да бисмо знали коју једначину користити, потребно је наћи начин разликовања хоризонталне и вертикалне елипсе.

5.5.1 Разликовање хоризонталне и вертикалне елипсе

Гранични случај између хоризонталне и вертикалне елипсе јесте кружница. Рецимо да имамо слику једног профила који представља хоризонталну и једног који представља вертикалну елипсу као у следећем приказу:

Слика 47: Хоризонтална и вертикална елипса - примјерци

Половина ширине слике на којој је хоризонтална елипса представљаће полупречник описане кружнице око елипсе, док ће половина ширине слике на којој је вертикална елипса представљати полупречник кружнице уписане у елипсу. Оба случаја су приказана на следећој слици:



Слика 48: Хоризонтална и вертикална елипса разликовање

Вриједност полупречника r је ширина половине слике. До вриједности h_k у оба случаја долази се на следећи начин:

$$h_k = r - \sqrt{r^2 - (0.75 \cdot r)^2}$$

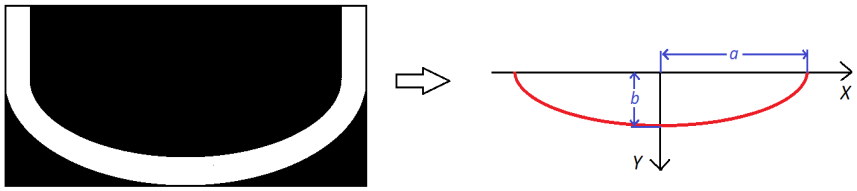
$$h_k = r - \frac{r\sqrt{7}}{4}$$

Вриједност h је заправо одбирак у низу усредњених одбирака чији је индекс оквирно једнак једној осмини дужине низа одбирака. Треба напоменути да се на основу услова са слике 48 не може коначно рећи да ли имамо хоризонталну или вертикалну елипсу. Они служе као смјернице за даљи процес препознавања хоризонталне и вертикалне елипсе. Имплементација процеса разликовања хоризонталне и вертикалне елипсе, који је претходно описан, налази се унутар функције:

- `TimingScanner.BendClassifier.CheckHorizontalVerticalEllipse`

5.5.2 Провјера за хоризонталну елипсу

Ако се као резултат претходне провјере добије да је $h_k > h$ то нам даље сугерише да лук профила на слици потенцијално има облик хоризонталне елипсе:



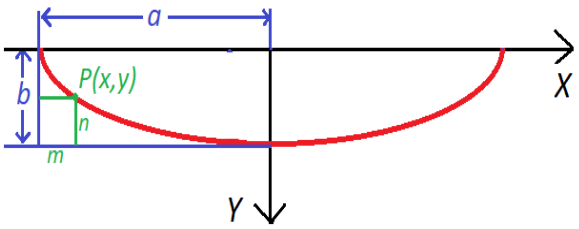
Слика 49: Хоризонтална елипса

Следећи корак јесте провјера да ли одбирци задовољавају једначину елипсе:

$$b^2x^2 + a^2y^2 = a^2b^2,$$

гдје су a и b редом већа и мања полуоса елипсе. Отежавајућа околност у односу на провјеру припадности кружници је то што елипса има двије полуосе. Једна од њих (полуосу a) може се одредити као вриједност половине ширине улазне слике. Међутим, проблем представља полуоса b која нам није позната. Није поуздано поставити вриједност b на вриједност висине слике јер профил може имати и равне краке промјењивих дужина. Такође, није поуздано поставити вриједност b на вриједност првог или последњег одбирка у низу јер ти одбирци у пракси могу да "проклизају" на раван дио профила и да ни приближно немају везе са вриједношћу која нам је потребна.

Рјешење проблема које је примјењено у овом пројекту односи се на претпостављање вриједности b помоћу вриједности a и вриједности одбирака лука јер је претпоставка да задовољавају једначину елипсе. Како се не би ослањали на један одбирак, вриједност полуосе b се рачуна као просјечна вриједност вриједности b које су добијене од отприлике једне четвртине узастопних одбирака лука који се налазе у првој половини низа одбирака. Формула по којој се добија вриједност b од вриједности a и једног одбирка лука чије су координате (x, y) изводи се на следећи начин:



Слика 50: Хоризонтална елипса - добијање мање полуосе

$$x = -(a - m) \text{ -> познато}$$

$$y = b - n \text{ -> непознато}$$

Ако претходне вриједности уврстимо у једначину елипсе:

$$b^2x^2 + a^2y^2 = a^2b^2$$

добија се:

$$b^2x^2 + a^2(b-n)^2 = a^2b^2$$

$$b^2x^2 + a^2(b^2 - 2bn + n^2) = a^2b^2$$

$$b^2x^2 + a^2b^2 - 2ba^2n + a^2n^2 = a^2b^2$$

$$b^2x^2 - 2ba^2n + a^2n^2 = 0$$

Рјешење се своди на рјешење квадратне једначине гдје увијек узимамо већи резултат, па је вриједност b за један одбирак једнако:

$$b_i = \frac{2a^2n_i + \sqrt{4a^4n_i^2 - 4x_i^2a^2n_i^2}}{2x_i^2}$$

Нека је N број одбирака преко којих рачунамо b, тада ће коначно b имати вриједност:

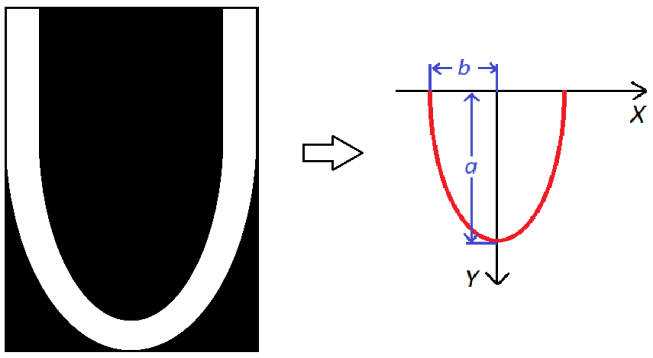
$$b = \frac{\sum_{i=1}^{N} b_i}{N}$$

Поступак провјере да ли одбирци задовољавају једначину хоризонталне елипсе са дефинисаним дозвољеним одступањима имплементиран је унутар функције:

- `TimingScanner.BendClassifier.IsHorizontalEllipse`

5.5.3 Провјера за вертикалну елипсу

Ако се као резултат претходне провјере добије да је $h_k < h$ то нам даље сугерише да лук профила на слици потенцијално има облик вертикалне елипсе.

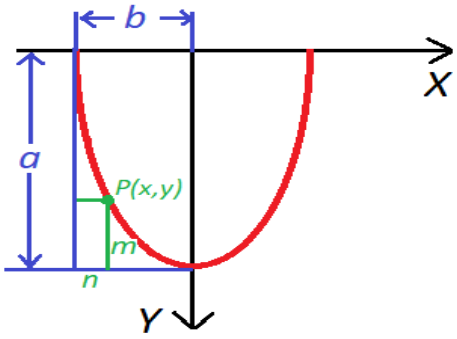


Слика 51: Вертикална елипса

Следећи корак јесте провјера да ли одбирци задовољавају једначину елипсе:

$$a^2x^2 + b^2y^2 = a^2b^2,$$

гдје су a и b редом већа и мања полуоса елипсе. Као и код хоризонталне елипсе, проблем је што не можемо директно израчунати вриједност једне полуосе. У овом случају то је већа полуоса a. Принцип добијања вриједности веће полуосе идентичан је као и принцип добијања мање полуосе код хоризонталне елипсе:



Слика 52: Вертикална елипса - добијање веће полуосе

$$x = -(b - n) \text{ -> познато}$$

$$y = a - m \text{ -> непознато}$$

Ако претходне вриједности уврстимо у једначину елипсе:

$$b^2y^2 + a^2x^2 = a^2b^2$$

добија се:

$$b^2(a - m)^2 + a^2x^2 = a^2b^2$$

$$b^2(a^2 - 2am + m^2) + a^2x^2 = a^2b^2$$

$$a^2b^2 - 2b^2am + b^2m^2 + a^2x^2 = a^2b^2$$

$$a^2x^2 - 2b^2ma + b^2m^2 = 0$$

Рјешење се своди на рјешење квадратне једначине гдје увијек узимамо већи резултат па је вриједност a за један одбирак једнако:

$$a_i = \frac{2b^2m_i + \sqrt{4b^4m_i^2 - 4x_i^2b^2m_i^2}}{2x_i^2}$$

Нека је N број одбирака преко којих рачунамо a, тада ће коначно a имати вриједност:

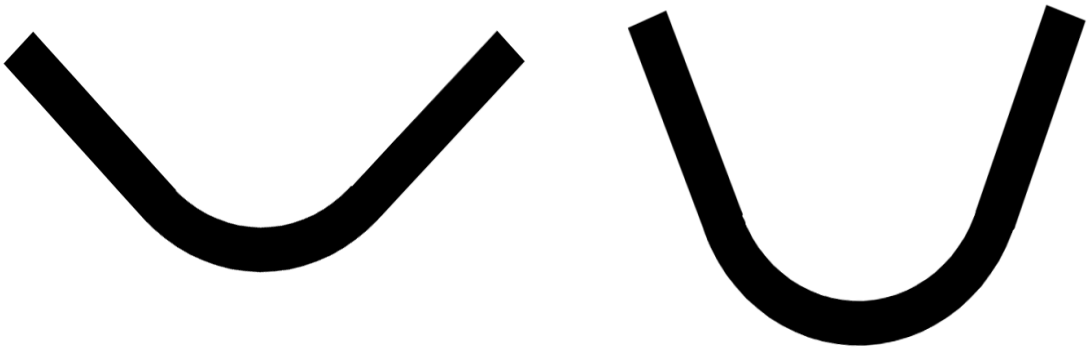
$$a = \frac{\sum_{i=1}^{N} a_i}{N}$$

Поступак провјере да ли одбирци задовољавају једначину вертикалне елипсе са дефинисаним дозвољеним одступањима имплементиран је унутар функције:

- `TimingScanner.BendClassifier.IsVerticalEllipse`

5.6 Провјера за L лук и кружни исјечак са равним крацима

Од класа које су од интереса, остала је могућност за L лук и кружни исјечак са равним крацима одређене дужине као на слици испод.



Слика 53: L лук и кружни исјечак са равним дијеловима - класе 2.1 и 3.1

Први корак јесте поступак препознавања и одсијецања дужине равног дијела профила.

5.6.1 Процес одсијецања равног дијела профила на слици

Циљ је што прецизније одсијећи раван дио профила на слици тако да остане само лучни дио профила. Имплементација је смјештена унутар функције:

- `TimingScanner.ScannerUtils.FindFlatParts`

Поступак пролази редом кроз следеће кораке:

1. Издвајање средње линије профила и координата тачака које чине ту линију на слици. Поступак којим се ово обавља је исти као и приликом одређивања оријентације профила на слици и детаљно је описан у том поглављу.

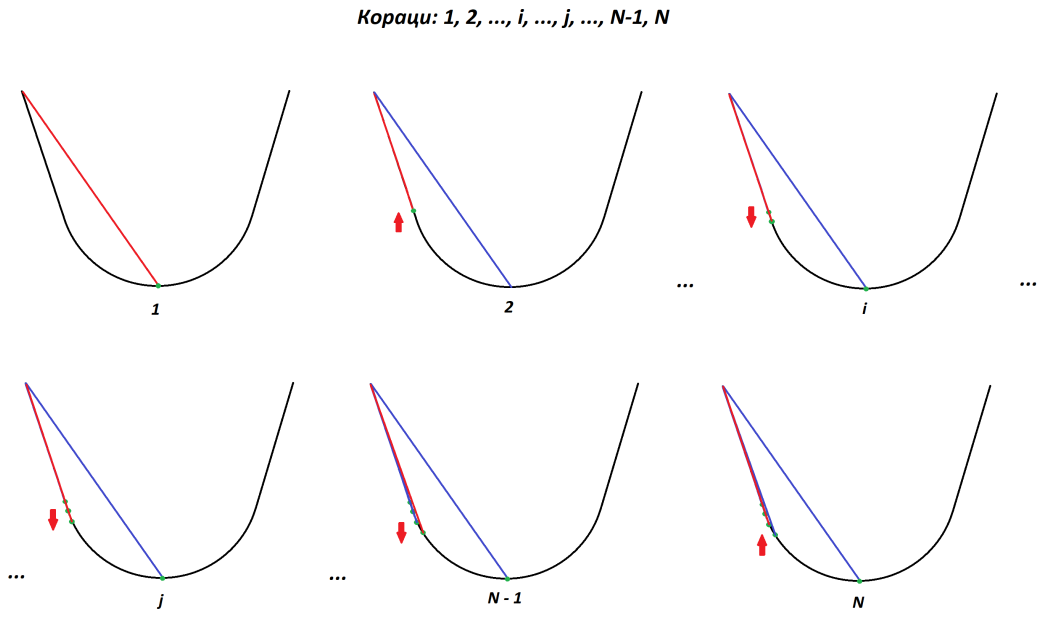
2. Тачке контуре у низу који добијамо у претходном кораку нису распоређене тако да редом чине тачке средње линије профила из које су издвојени. У овом кораку врши се њихово сортирање. Познато је које су почетна и крајња тачка линије. Сортирање се врши мјерењем удаљености између тачака. Алгоритам за сортирање смјештен је унутар функције:

   - `TimingScanner.ScannerUtils.GetSortedContourPoints`

3. Након сортирања врши се усредњавање координата тачака средње линије по истом принципу као усредњавање одбирака лука. Једина разлика је што је сада потребно усредњавати и x и y координату тачака. Усредњавање се врши у двије итерације. У првој итерацији се користи филтер дужине 5 тачака, а у другој филтер дужине 3 тачке. За усредњавање тачака се користи функција:

   - `TimingScanner.ScannerUtils.MovingAverageSmooth1DPointArr`

4. Идеја је отклонити онолико тачака средње линије са почетка и краја низа колико их представља раван дио контуре профила. На основу координата тачака које су на индексима на којима се одсијеца низ, одсијеца се и улазна слика профила. За одређивање колико тачака припада равном дијелу користи се метод бисекције који је дјелимично промијењен за потребе овог пројекта. Метод функционише тако што се повуче права између прве и средишње тачке низа, а затим се провјерава да ли све тачке између њих припадају тој правој са неким дозвољеним одступањем. То је усвојено као почетни корак јер тачке сигурно неће задовољити услов припадности правој ако су краци профила једнаке дужине. Следећи корак је да се преполови број тачака који је претходно провјераван, тако да се испитује прва четвртина низа свих тачака. Ако услов опет није задовољен провјерава се осмина и тако редом док услов не буде задовољен. Ако је услов задовољен крећемо се од тренутне тачке у супротном смјеру кроз низ са кораком од једне тачке док не наиђе на секвенцу која не задовољава једначину праве од прве до тренутно разматране тачке. Када се дође до те секвенце тачака, поново се мијења смијер кретања кроз низ и помјерање се врши за онолико тачака колико износи максимална дозвољена секвенца тачака које одступају више од дозвољене вриједности. На индексу на коме је последња тачка се одсијеца низ. С обзиром да су оба крака једнаке дужине, други крак се отклања тако што се са краја низа одсијече онолико тачака колико се одсијеца са почетка. Дио описаног процеса је приказан на следећој слици:



Слика 54: Принцип отклањања равног дијела

5. Следећи корак јесте одсијецање равног дијела профила са улазне слике и формирање слике на којој је само контура лука. Рецимо да је потребно одсијећи раван дио профила на следећој слици:

Слика 55: Улазна слика за одсијецање равног дијела

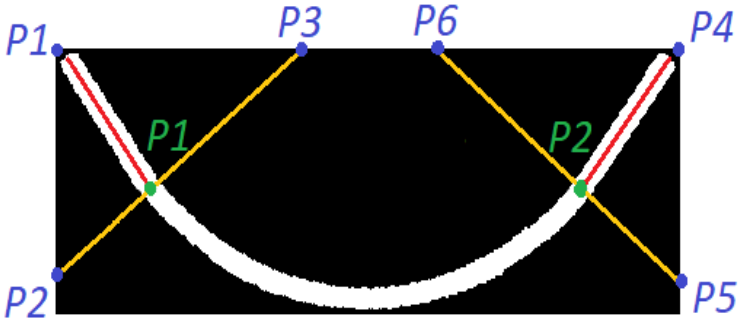
Нека је позната тачка средње линије на којој одсијецамо и права која пролази кроз ту тачку и почетну тачку средње линије профила.

- Једно рјешење је одсијећи слику одозго до праве која је на следећој слици означена жутом бојом и коју чине тачке $P1$ и $P2$:

Слика 56: Одсијецање равног дијела - први начин

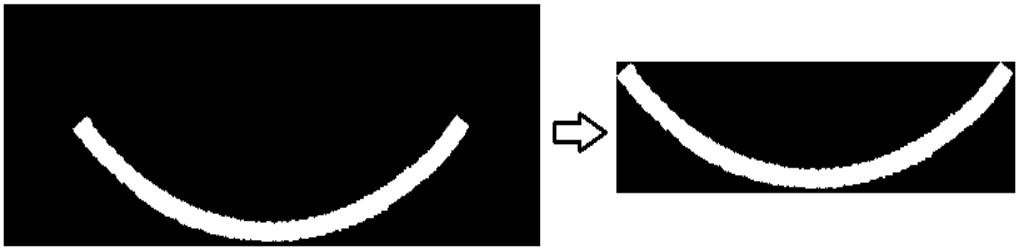
Међутим, овај начин није добар ако је профил веће дебљине. У том случају, одсијецањем као на слици изнад, низ одбирака који се касније издвајају садржаће и даље један дио одбирака који чине раван дио профила. Због тога се, у овом пројекту, користи други начин.

- Други начин је да се провуку праве које су нормалне на праве означене црвеном бојом (примјењује се услов нормалности двије праве $k_1 = -\frac{1}{k_2}$) и да пролазе кроз тачке $P1$ и $P2$. Затим се пронађу тачке пресјека тих правих са ивицама слике. Поступак је приказан на следећој слици:



Слика 57: Одсијецање равног дијела - други начин

Сада се контуре (троуглови) ограничене тачкама $P_1P_2P_3$ и $P_4P_5P_6$ обоје у црну боју (боју супротну боји контуре профила). Након тога се слика одсијеца до преостале контуре профила. Добије се следећи резултат:



Слика 58: Одсијецање равног дијела - резултат

Битно је напоменути да се читав процес отклањања равног дијела профила на слици могао заснивати на одбирцима лука улазне слике, јер важи ограничење да су краци једнаке дужине и симетричног положаја. Међутим, ако такво ограничење не важи у пракси (планирано је да у потенцијалном будућем развоју пројекта не важи!), такав начин би био потпуно неупотребљив.

5.6.2 Провјера за L лук са равним крацима

Када се отклоне равни дијелови профила са слике, резултујућа слика се мијења тако да јој ширина буде 300 пиксела: Разлог за то је објашњен у поглављу 4.6: Свођење на фиксну ширину слике. Затим се из такве слике издваја низ одбирака лука и усредњава се по истом принципу као и до сада. Такви одбирци пролазе идентичну провјеру као за L лук без равног дијела. Провјера се врши са истим вриједностима дозвољених одступања и такође се позива функција:

- `TimingScanner.BendClassifier.IsArcL`

5.6.3 Провјера за кружни исјечак са равним крацима

Као и код претходне провјере за L лук, када се отклоне равни дијелови профила са слике, резултујућа слика се мијења тако да јој ширина буде 300 пиксела. Разлог за то је објашњен у поглављу 4.6: Свођење на фиксну ширину слике. Затим се из такве слике издваја низ одбирака лука и усредњава се по истом принципу као и до сада. Такви одбирци пролазе идентичну провјеру као за кружни исјечак без равног дијела. Провјера се врши са истим вриједностима дозвољених одступања и такође се позива функција:

- `TimingScanner.BendClassifier.IsAnySection`

6 Опис рада апликације

У наставку ће бити описан коначан принцип класификације, а биће приказан и начин рада саме апликације заједно са графичким корисничким интерфејсом.

6.1 Коначан принцип класификације

У претходним поглављима описан је поступак класификације профила са једног фрејма који је издвојен из снимка 3D камере. Међутим, због начина рада саме 3D камере сваки фрејм неће садржати потпуно исти распоред тачака контура, па због тога се ни након процеса отклањања шума неће добити увијек исти резултат. Један примјер је приказан на следећој слици гдје се могу уочити ситне разлике на фрејмовима истог снимка након процеса отклањања шума:

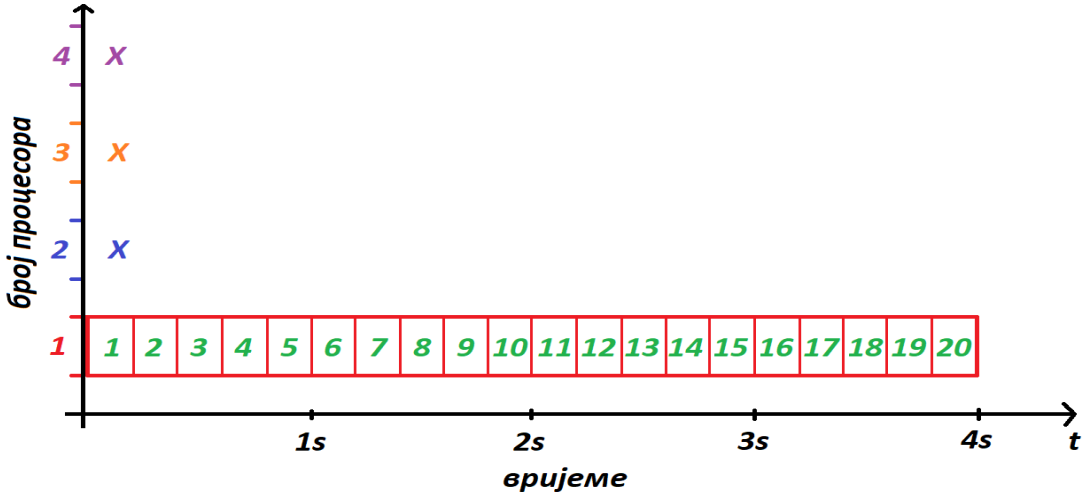
Слика 59: Разлика у фрејмовима истог снимка

Те разлике у пракси могу бити и веће. То значи да се за све фрејмове истог снимка неће увијек добити исти разултат класификације. Стога се не може издвојити било који фрејм и над њиме поуздано рећи која је класа у питању. Рјешење је узети више фрејмова и одрадити класификацију над сваким посебно. Затим се одреди која класа од претходно разматраних је препозната и на колико фрејмова. Она која је препозната на највише фрејмова представља крајњи резултат класификације. Кориснику је омогућено да има увид у то са коликом вјероватноћом класификатор тврди да је на снимку лук баш оне класе која је издвојена као резултат и приказана кориснику.

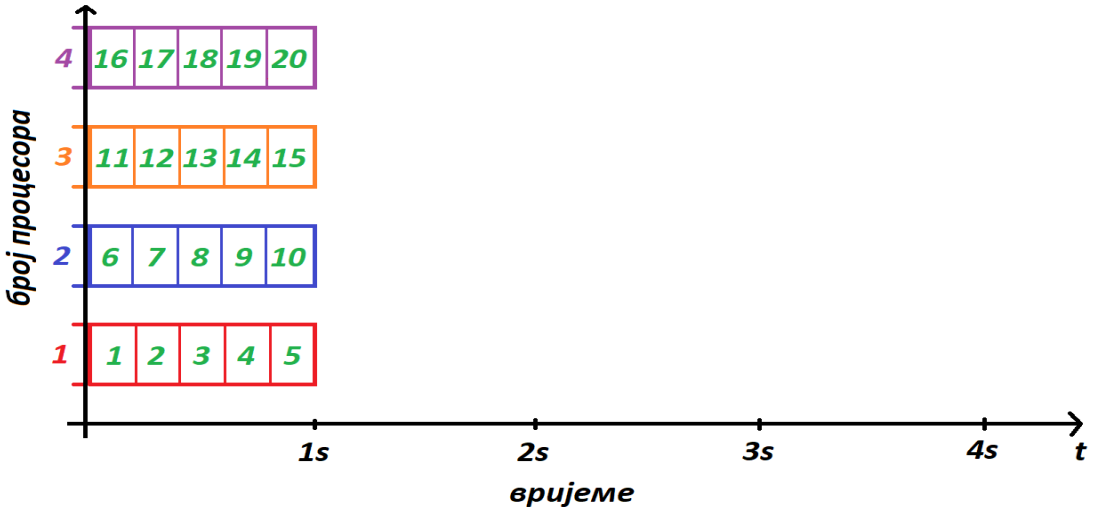
Да би се имплементирала таква класификација искоришћен је принцип рада линеарног бафера у коме се у сваком моменту налази неколико последњих фрејмова са снимка. Када дође нови фрејм, први фрејм у низу који представља бафер се уклања, а на крај се додаје фрејм који је тренутно дошао са камере. Кликом на дугме за класификацију пролази се кроз низ фрејмова који представља поменути бафер и за сваки фрејм се врши процес издвајања и класификације профила. Након класификације за сваки појединачни фрејм чува се разултат и на крају се провјерава која класа лукова је најзаступљенија међу добијеним резултатима.

6.2 Паралелизација

Идеја за паралализацију проистиче из чињенице да се из снимка издваја бафер са више фрејмова и да се над свим тим фрејмовима врши иста обрада, која у случају великог броја фрејмова може да потраје. У вези с тим, бафер се може подијелити на онолико дијелова колико постоји процесорских језгара на рачунару и да свако језгро обрађује свој дио бафера чиме се знатно убрзава рад апликације. На следећим сликама су приказани начин без паралелизације и начин на који може да функционише паралелизација ако се обрађује 20 фрејмова и ако је процесор садржи четири физичка језгра на које је извршење задатка подједнако распоређено:



Слика 60: Без паралелизације



Слика 61: Са паралелизацијом

Видимо да се укупно вријеме извршавања програма смањује са бројем процесора (физичких језгра процесора) на које се дијели извршавање укупног задатка.

У овом пројекту, паралелизација је извршена унутар функције:

- `MainWindow.ClassifyButton_Click`

Паралелизација се постиже коришћењем уграђене функције:

- `System.Threading.Tasks.Parallel.For`

која узима улогу `for` петље која се може паралелизовати. Функција као аргументе прима почетни индекс у итерацији, крајњи индекс (максимални број дијелова на који може бити подијељен укупан задатак) и делегат на функцију који ће се позивати једном по итерацији. Делегат је тип који представља референцу на методе са одређеном листом параметара и повратним типом. Када се инстанцира делегат, може се повезати његова инстанца са било којом методом са компатибилним потписом и повратним типом. Делегати се користе за прослеђивање метода као аргумената другим методама. Руковаоци догађајима (`EventHandler`-и) нису ништа друго до методе које се позивају преко делегата. Било која метода из било које доступне класе или структуре која одговара типу делегата може бити додјељена делегату. Метода може бити статичка или инстанца. Ова флексибилност значи да се може програмски промијенити позив метода или унијети нови код у постојеће класе. У случају из овог пројекта, делегату се додјељује функција која служи за класификацију фрејмова који јој се као аргумент прослијеђују унутар листе. Функцији се прослијеђује дио укупне листе фрејмова који ће бити обрађивани паралелно са другим дијеловима листе фрејмова. На колико дијелова ће се дијелити листа зависи од одабраног максималног броја итерација (дијелова на који може бити подијељен укупан задатак).

6.3 Графички кориснички интерфејс

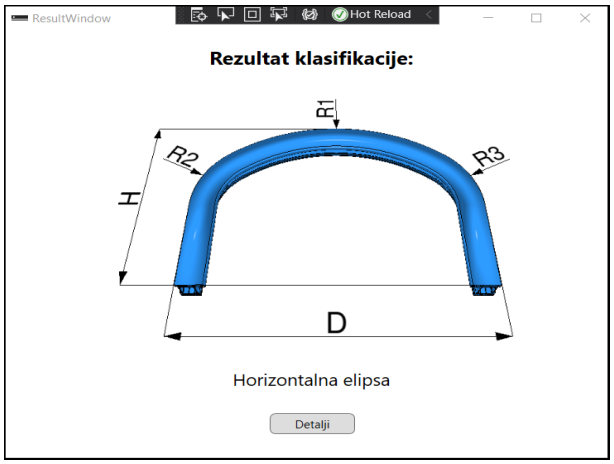
Графички кориснички интерфејс представља C# WPF апликацију. Функционалност и изглед почетног прозора дефинисани су унутар датотеке:

- `DepthBasics-WPF.MainWindow`

Почетни прозор који корисник види по покретању апликације има следећи изглед:

Слика 62: Почетни прозор апликације

Апликација је намјењена снимању уживо профила и могућности њихове класификације. Корисник има опцију да унесе максималну дубину пиксела који ће ући у разматрање. Та дубина би требало да представља удаљеност камере од платформе на којој лежи профил. Такође, постоји могућност чувања тренутне слике профила у меморију рачунара (Screenshot). Корисник има и опцију да освјежи бафер у коме су фрејмови. То би требало да се ради обично када на платформу тек стигне нови профил. Док се бафер не напуни дефинисаним бројем фрејмова (у нашем случају 20), дугме за класификацију неће бити доступно. Кликом на дугме "KLASIFIKUJ" отвара се прозор на коме је крајњи резултат класификације:



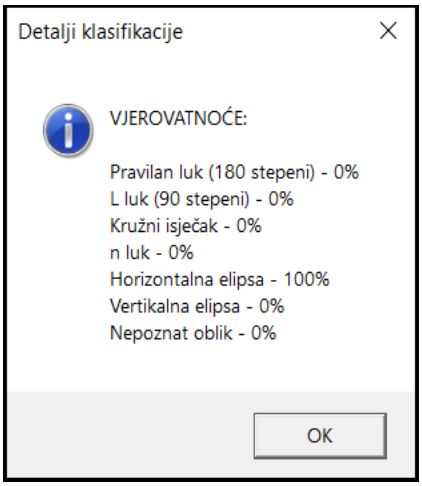
Слика 63: Прозор - резултат класификације

Крајњи резултат је хоризонтална елипса, а примјећује се да је то тачан резултат ако се погледа улазна слика профила на почетном прозору. Функционалност и изглед прозора

на коме се приказује резултат класификације дефинисани су унутар датотеке:

- `DepthBasics-WPF.TimingScanner.ResultWindow`

Корисник има могућност да погледа детаље класификације тј. са којом сигурношћу се може рећи да профил припада класи из крајњег разултата. Ако се кликне на дугме 'Detalji' отвара се следећи прозор:



Слика 64: Прозор - детаљи класификације

Из детаља класификације се види да класификатор препознаје са великом сигурношћу да је на улазном снимку хоризонтална елипса тј. резултат класификације са сваког фрејма из бафера је хоризонтална елипса.

7 Закључак

У оквиру овог рада обрађене су бројне технике дигиталне обраде слике, геометријски докази на којима се заснива класификација лукова као и имплементација поменутих метода кроз израду C# WPF апликације. Треба напоменути да се до имплементације многих дијелова овог пројекта дошло експериментално уз претходне мање успјешне покушаје рјешавања проблема. Поготово се то односи на постављање дозвољених одступања од идеалних вриједности.

С обзиром да је написани програм потребно још доста тестирати и провјеравати у пракси прије његове интеграције у неки систем, јасно је да ће бити простора за разна побољшања програма који је написан у оквиру овог пројекта. Такође је битно напоменути и нека ограничења која су узимана у обзир приликом израде пројекта као нпр. чињеница да профил мора имати краке једнаке дужине и симетричног положаја, претпоставка да је савијени профил највећа контура на слици скалираној по дубини, итд. Таква ограничења ће се у будућности настојати отклонити у зависности од потребе и значаја ограничења.

Могућа унапређења се односе и на прецизније отклањање шума, прецизније усредњавање одбирака као и прецизније отклањање равног дијела профила на слици. Када је у питању отклањање равног дијела профила на слици, треба напоменути разлог зашто се отклањање равног дијела не ради прије било каквог покушаја класификације, него тек после провјера за одређене класе. Разлог је тај што профил не мора уопште имати раван дио на крацима. У том случају би отклањање равног дијела са неким дозвољеном одступањем довело до отклањања корисног дијела лука. Иницијално препознавање да ли профил има раван дио или нема испоставља се као прилично тежак задатак, што свакако значи да има додатног простора за унапређење програма јер би иницијално отклањање равног дијела крака профила, уколико раван дио постоји, било од великог значаја и омогућило би прецизнији рад осталих дијелова програма, а самим тим и прецизнију класификацију.

Након потенцијалне успјешне имплементације класификатора, скенер чији је класификатор саставни дио, може се надоградити на функционалности прилично прецизног мјерења углова закривљености профила како би се детаљније могао описати сами профил са слике.

8 Литература

[1] Richard Szeliski: Computer Vision: Algorithms and Applications, 2. издање, септембар 2021.

[2] OpenCV Documentation, Image Processing, август 2022.
OpenCV Image Processing (docs.opencv.org)

[3] Tutorials Point: OpenCV Tutorial, август 2022.
OpenCV Tutorial (tutorialspoint.com)

[4] Microsoft Docs, C# Tutorial, август 2022.
C# Tutorial (docs.microsoft.com)

[5] Tutorials Point: WPF Tutorial, август 2022.
WPF Tutorial (tutorialspoint.com)

[6] T. Y. Zhang and C. Y. Suen: A Fast Parallel Algorithm for Thinning Digital Patterns, март 1984.
Џанг-Сун (Zhang-Suen) алгоритам (dl.acm.org)

[7] TIM-ING CENTAR, Машине за савијање, август 2022.
TIM-ING машина CNC 3D (tim-ing.com)

[8] Kinect for Windows SDK 2.0 Download, август 2022.
Kinect for Windows SDK 2.0 - преузимање (microsoft.com)