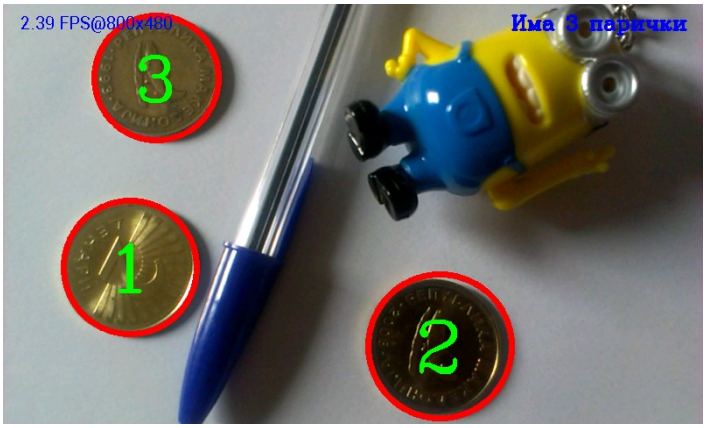
REAL TIME
COIN DETECTION



Предмет:
Обработка на слика (ОНС)

Изработил:
Мето Трајковски
121047

Ментор:
Проф. Д-р Ивица Димитриовски

Скопје, Ноември 2015 год.

Содржина:

1. Вовед

2. Опис на апликацијата

3. Објаснување на алгоритмот

4. Објаснување на други испробани алгоритми и експерименти во текот на изработка на апликацијата

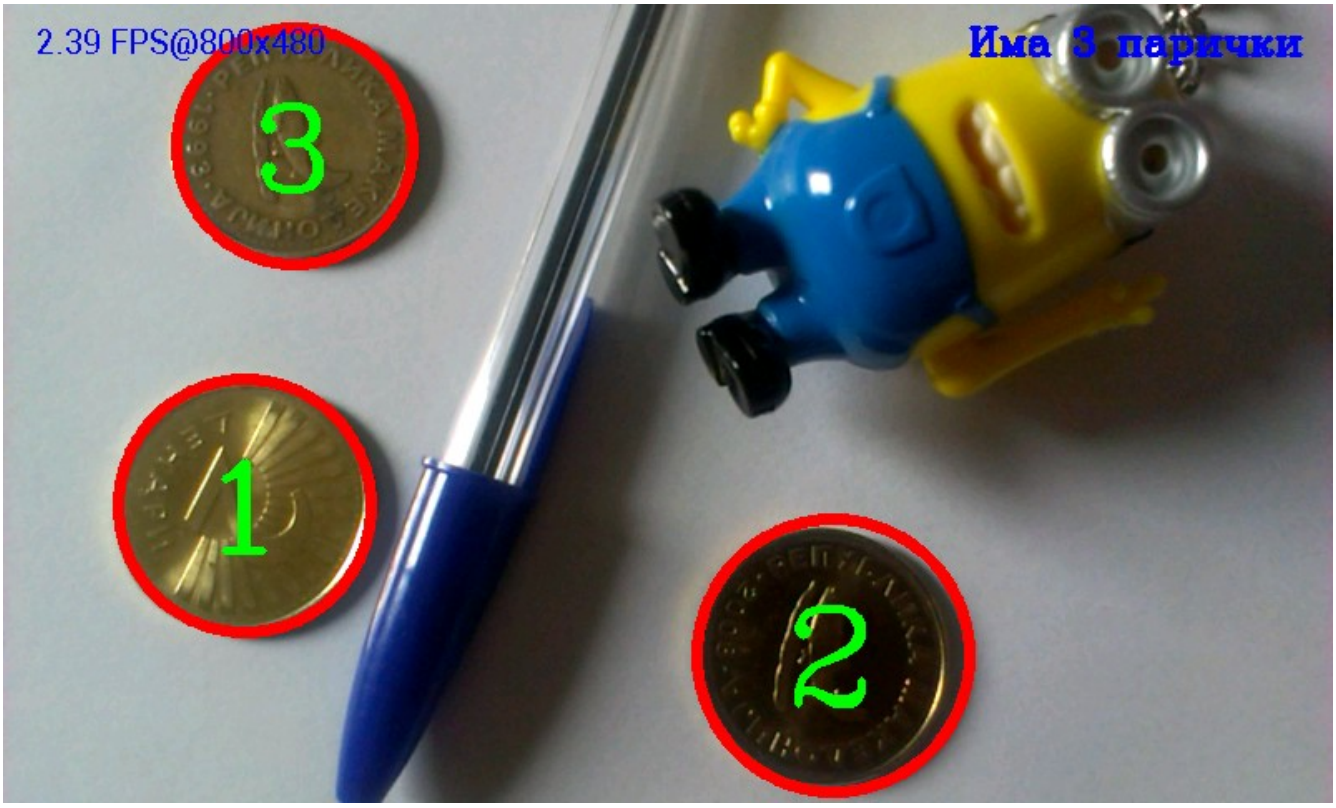
1. Вовед

За овој проект по предметот “Обработка на слика” изработив Android апликација која во реално време (додека камерата на Android телефонот снима) ги пронаоѓа сите монети во дадениот фрејм и ги означува.

Исто така за овој проект се послужив со open-source библиотеката OpenCV (*Open Source Computer Vision*) која е достапна во скоро сите од најпознатите јазици, која служи за компјутерска визија и ги поседува скоро сите алгоритми кои ги поминавме по овој курс. Но сепак не го искористив најновиот OpenCV (3.0) туку искористив постара верзија (2.4) бидејќи многу полесно можаат да се цртаат форми и испишува текст на екранот, а тоа ми беше многу потребно за оваа апликација.

Апликацијата работи на следниов начин: секој frame (слика од камерата, инаку андроид камерата е способна во 1 секунда да дава 30 слики, но бидејќи се користат многу комплексни и сложени алгоритми затоа во овој проект камерата враќа во просек по 3 слики во секунда и затоа човековото око може да примети дека оскокнуваат сликите од еден во друг кадар) се зема како влез и после неколку преработки се враќа истиот тој фрејм каде што се заокружени паричките и на нив пишува која паричка по кој редослед била пронајдена (овој дел подетално ќе биде објаснет во секциите 2. и 3.).

2. Опис на апликацијата



Сликата од погоре е само еден screenshot додека снимав место каде што има повеќе различни објекти и истата ќе ми послужи за објаснување на апликацијата.

Најпрво ќе почнам со горниот лев агол. Во горниот лев агол е прикажано со колку frame (слики) во секунда работи моментално камерата и оделено со мајмунче (@) е прикажано колкав е секој frame (поточно колкава е резолуцијата, на мојот телефон резолуцијата е 800x480).

Потоа во горниот десен агол пишува во моментот колку парички се детектирани на сликата.

И во главниот дел на view-то го дава излезот од камерата со сите детектирани монети зокружени со црвена линија и во нив напишан редниот број по кој биле пронајдени монетите.

3. Објаснување на алгоритмот

Сега доаѓаме до најважниот дел од апликацијата, каде што е употребено знаењето од овој предмет. После повеќе испробани (можеби над 10 алгоритми) се решив да го искористам овој алгоритам бидејќи ги даваше едни од најдобрите резултати (скоро 95% точност) но исто така беше изводлив за андроид телефон (во мојот случај телефон со 1 GHz процесор) имаше алгоритми кои даваа и многу подобри резултати но тие се извшуваа со скоро 0.2 frame (слики) во секунда. и Овој алгоритам ќе го поделам во пет чекори каде што секој чекор ќе го објаснам подетално и со соодветна слика добиена од алгоритмот.

- Чекор 1

Првиот чекор е превземањето на фрејмот од камерата.

Код:

```
rgbaFrame = inputFrame.rgba();
```

Слика:

- Чекор 2

Вториот чекор е претварање на оргиналната слика во сива бидејќи што помалце бои има тоа е полесно да се детектираат сите работи.

Код:

```
Imgproc.cvtColor(rgbaFrame, grayframe, Imgproc.COLOR_RGBA2GRAY);
```

(инаку сивата слика исто така може да се добие на следниов начин `grayframe = inputFrame.gray();` ,но не ја превземам така бидејќи на крај ќе ми треба и оргиналната слика, па затоа со помош на метод ја претварам)

Слика:

- Чекор 3

Бидејќи сите слики имаат шум, а посебно сликите добиени од камерата додека снима па затоа во овој трет чекор го остранувам шумот. Шумот може да се отстрани со повеќе методи, ја избрав да го остранам со Gaussian Blur. Исто така пробав и да го остранам со Median Blur кој даваше побрзо резултати но некогаш даваше и погрешни резултати, наоѓаше парички наде што нема парички, па затоа се одлучив за поспорото но посигурно прочистување на шумот. Исто така Gaussian Blur се базира на нормалната распределба која е застапена секаде околу нас и служи како најдобар апроксиматор за предвидување на шумот и соодветно справување со него.

Код:

```
Imgproc.GaussianBlur(grayframe, grayframe, kSize, 2, 2);
```

исто така еве го и кодот за другиот филтер што го спомнав Median Blur:

```
Imgproc.medianBlur(grayframe, grayframe, 7);
```

Слика:

- Чекор 4

Во овој чекор се прави најбитната работа за успешноста на овој алгоритам, се детектираат рабовите (edge detection), работејќи со edge detection се добиваа многу подобри резултати одколку со thresholding (иако јас кога започнав со правење на апликацијата мислев дека thresholding-от е вистинското решение, бидејќи има само 2 бои и јасно е одделена позадината од монетите но во секцијата 4 ќе објаснам зошто не е добро).

Код:

```
Imgproc.Sobel(grayframe, gradientX, ddepth, 1, 0, 3, scale, delta, Imgproc.BORDER_DEFAULT );
Imgproc.Sobel(grayframe, gradientY, ddepth, 0, 1, 3, scale, delta, Imgproc.BORDER_DEFAULT);

Core.convertScaleAbs(gradientX, absGradientX);
Core.convertScaleAbs(gradientY, absGradientY);

Core.addWeighted(absGradientX, 0.5, absGradientY, 0.5, 0, edgeDetect);
```

Слика:

- Чекор 5

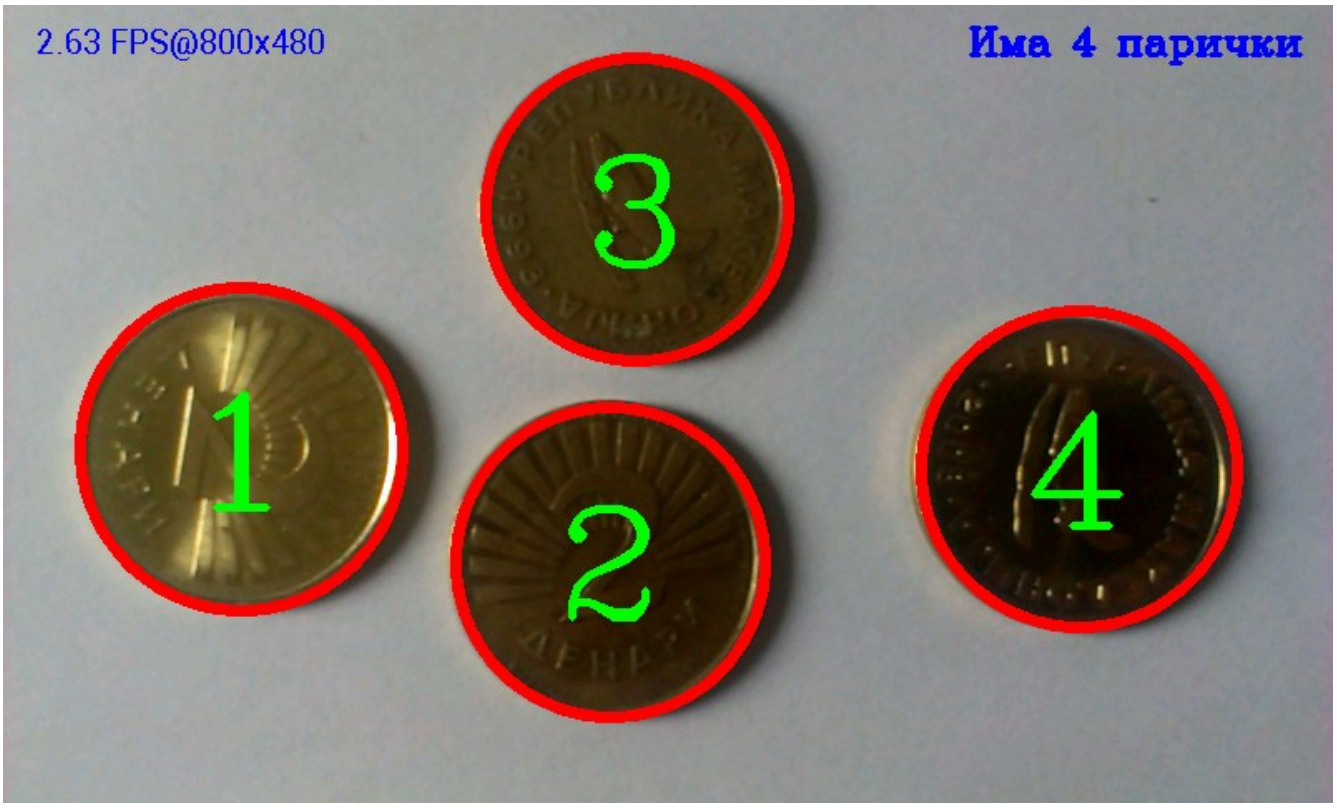
И ова е крајниот чекор каде што ги наоѓам паричките и само ги заокружувам.

Код:

```
Imgproc.HoughCircles(edgeDetect, detectedCoins,
  Imgproc.CV_HOUGH_GRADIENT, 2,
  grayframe.rows() / 5, cannyUpperThreshold,
  accumulator, coinMinRadius, coinMaxRadius);
```

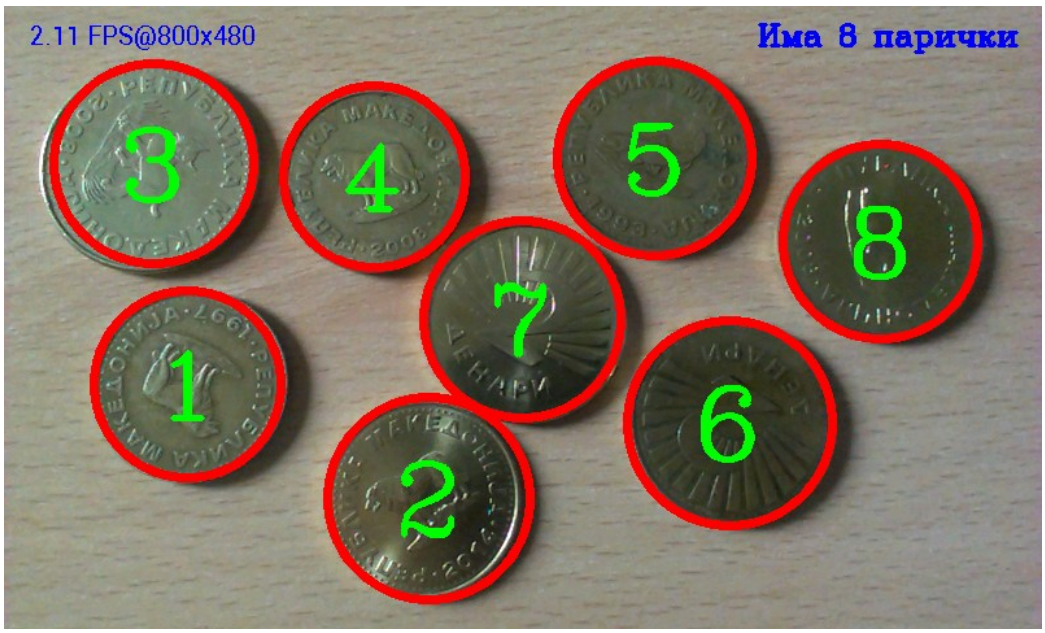
(остатокот од кодот е во проектот Coin Detection)

Слика:

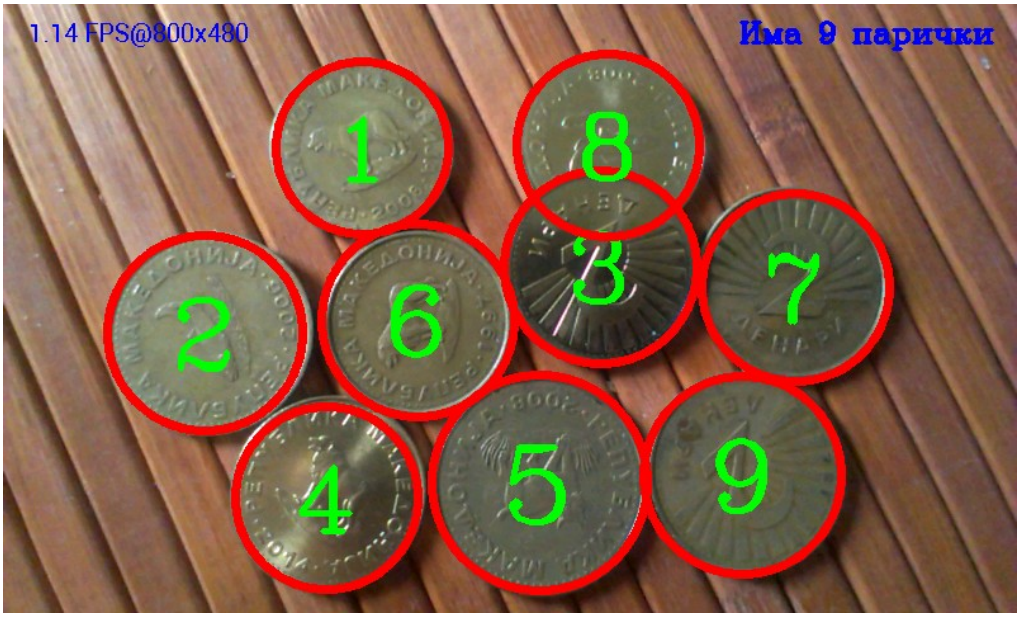


Овој алгоритам не работи само на бела позадина како што беше прикажано на горните слики, туку работи на било каква позадина еве неколку примери:

Дрвена подлога (слична боја како парите)

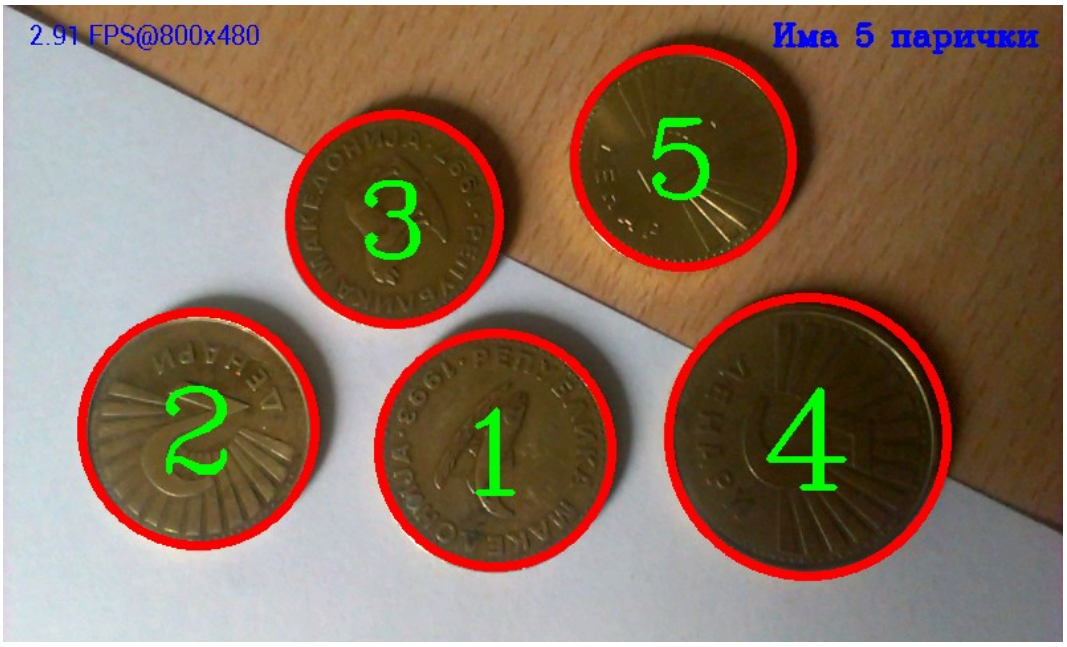


Исто така дрвена позадина со линии

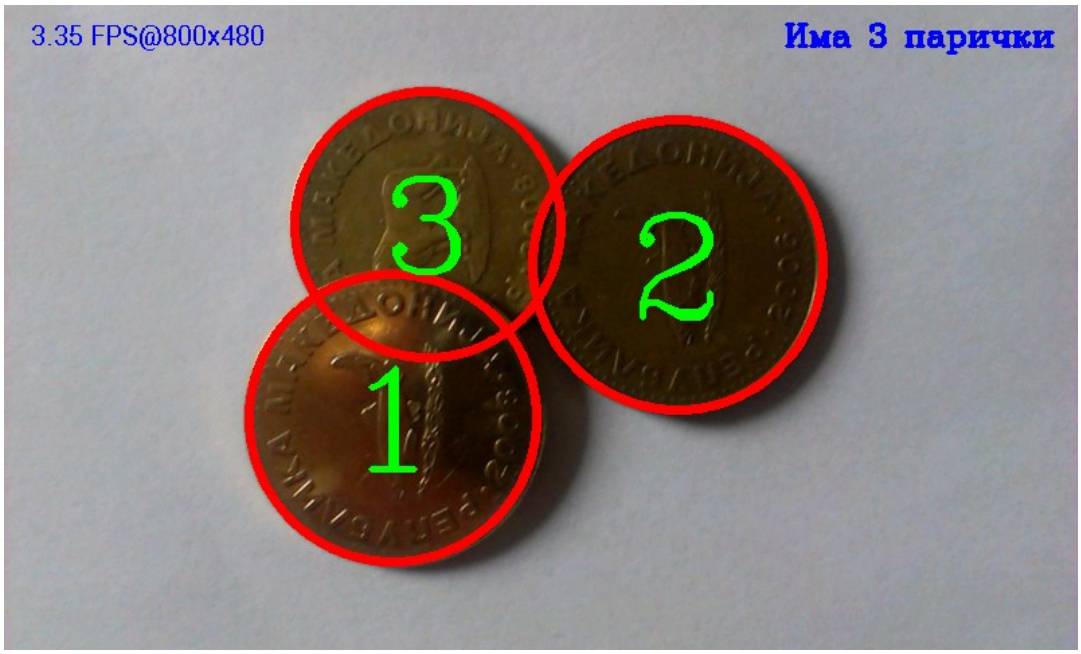


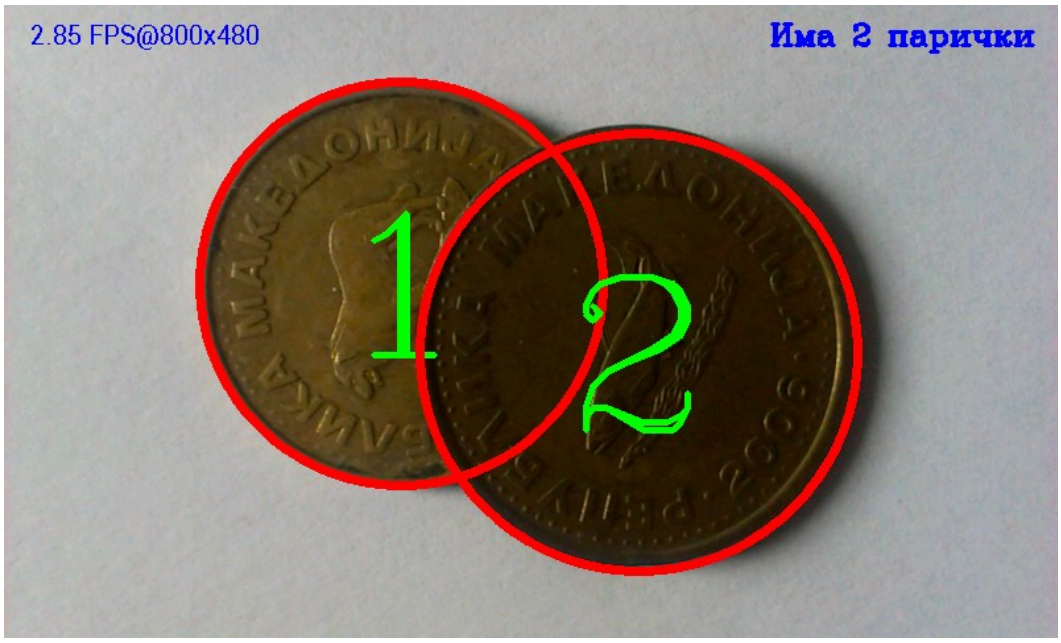
Сина позадина со сенка и нестабилна контура

Полу бела и полу дрвена подлога



Потоа овој алгоритам супер работи и кога паричките се преклопуваат, значи не е потребно да се гледа целата паричка. Во продолжение неколку слики со примери.

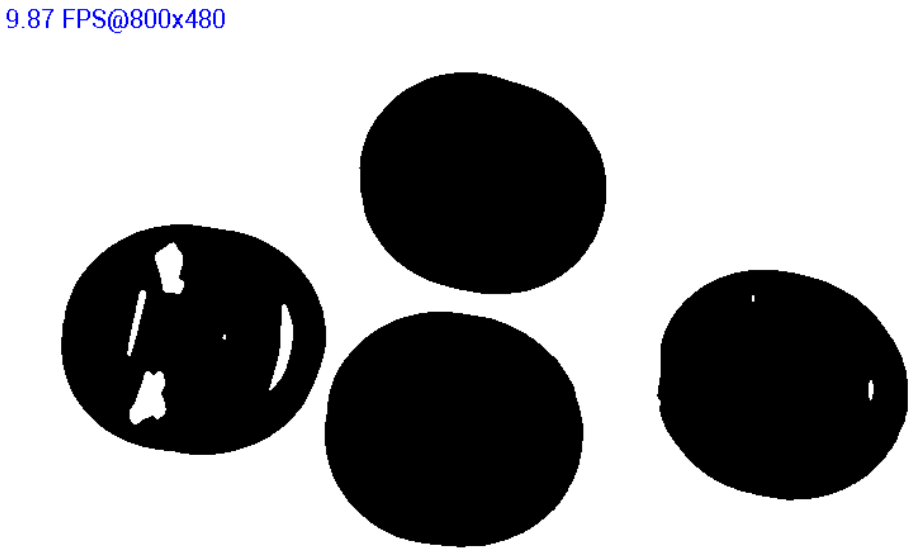



2.85 FPS@800x480
Има 2 парички
1
2

4. Објаснување на други испробани алгоритми и експерименти во текот на изработка на апликацијата

Како што претходно напоменав за целта на овој проект експериметирав со повеќе алгоритми, од кој ќе се задржам на алгоритмот со threshold бидејќи мислев дека треба да е најуспешен но тој не даде никакви резултати, еве илустрирано со слики како изгледаше.

Ова е thresholding-от кој многу убаво ги одвојува паричките од позадината и јасно е што е што бидејќи само 2 бои се застапени, но сепак резултатот не беше очекуван.



Ова е резултатот и како што може да се примети не е детектирана ниту една паричка и од тука заклучив дека алгоритмот Hough Circles не функционира со thresholding па затоа се префрлив на edge detection.

Бидејќи многу време посветив на овој проект и ми стана интересен затоа продолжив да експериментирам и алгоритмите да ги испробувам на компјутер каде што многу побрзо и многу покомплицирани алгоритми се извршуваа за многу мало време и скоро да немаше кочење. Па затоа решив освен coin detection да испробам и coin recognition. Значи поточно сега имам апликација која брои колку парички има на сликата, но развивам и апликација која ќе изброи колку денари има на сликата (пр: ако има 2 петки, 1 двојка и 3 единици да изброи дека има 15 денари).

Бидејќи со coin detection ги пронајдов регионите на паричките и само ми останува да ги препознаам цифрите во паричките и да избројам колку вкупно денари имам. Препознавањето на паричките го правам со помош на Tesseract OCR (OCR – Optical Character Recognition). Но наидов на еден нов проблем, кога паричката е ротирана, не е исправена, тогаш многу тешко ја наоѓа бројката. А друг проблем е што многу споро се извршува на Android телефон (поточно на мојот со 1 GHz процесор, може на некој појак телефон би се извршувало побрзо, а на лаптоп со 2.6 GHz процесор ми се извршува супер).