Przetwarzanie i analiza obrazu – projekt 2D

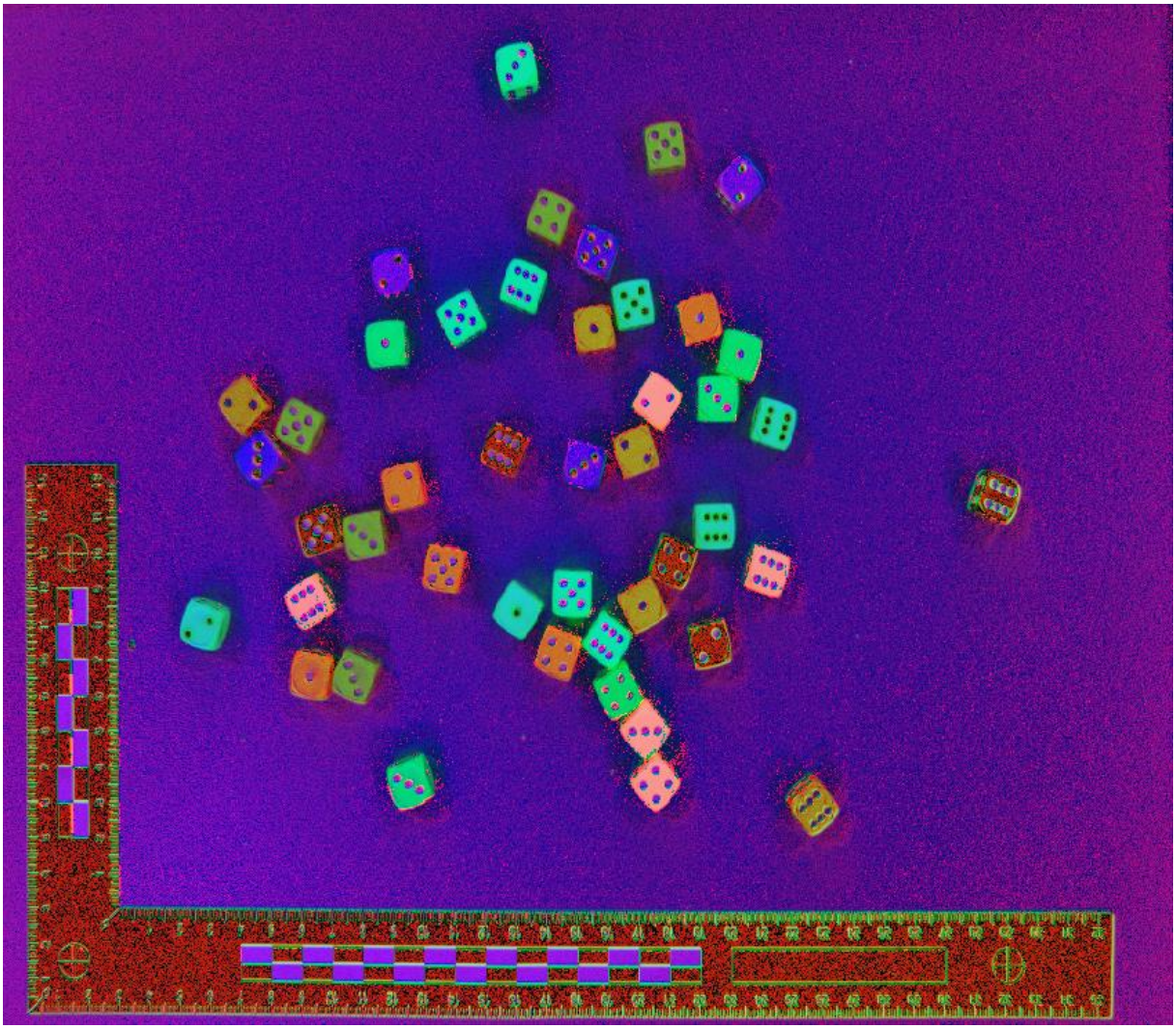
Krzysztof Bigaj
Gr. 63ip
16.04.2015
Temat: Y02o – wyznaczenie sumy oczek nieparzystych wyrzuconych na kostkach żółtych

I Algorytm postepowania dla widoków od góry:

Rys. topdown_02.png – zdjęcie oryginalne

1. Przejście z przestrzeni barw RGB na HSV

Pierwszym krokiem jest przejście z barw RGB na HSV. Dzięki temu łatwiej jest wyróżnić kolor żółty przy progowaniu w następnym kroku.



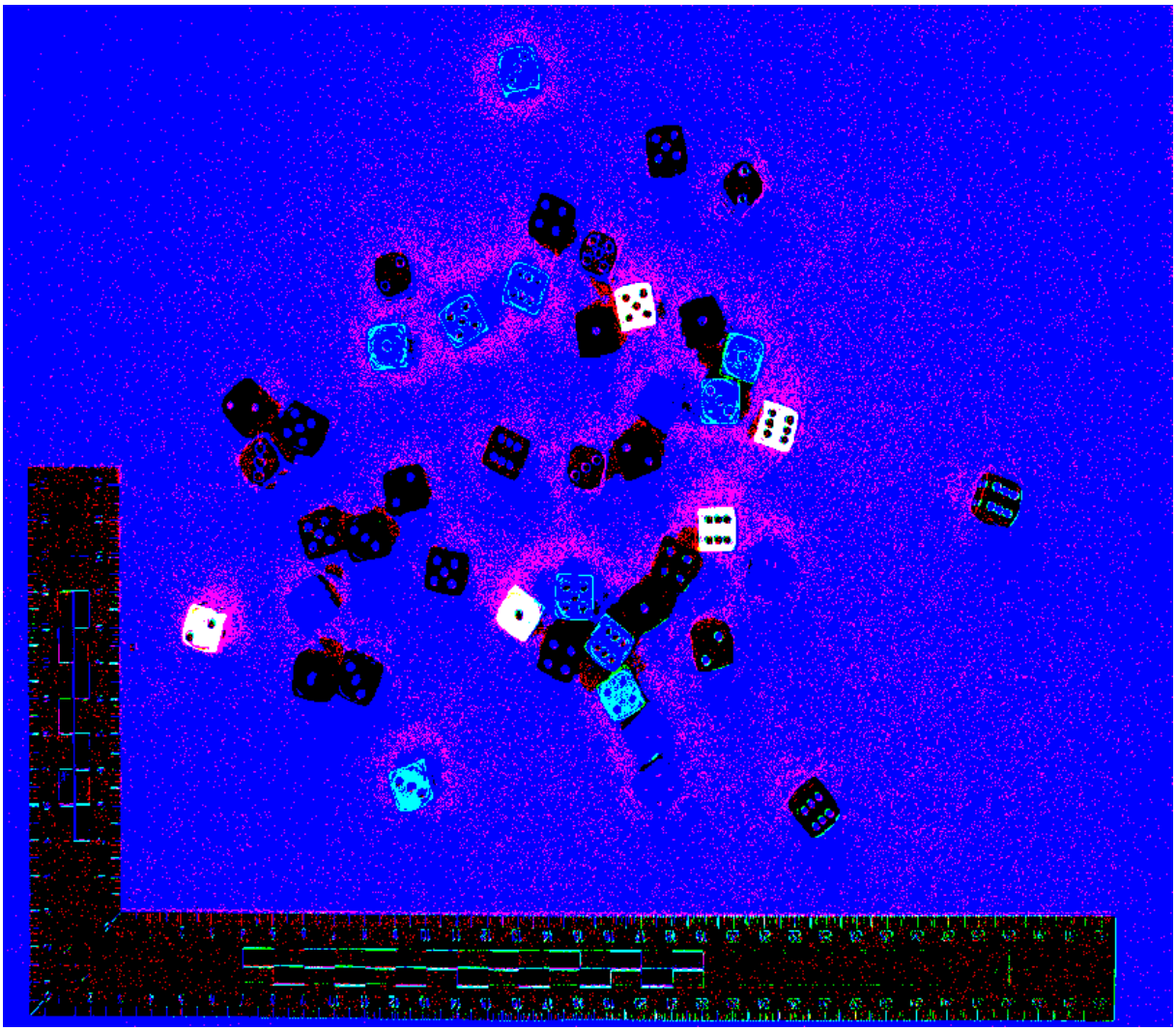
Rys. topdown_02.png – po zmianie przestrzeni barw na HSV

2. **Progowanie wartości poszczególnych kanałów (barwy, nasycenia koloru oraz mocy światła białego)**

Ustawiam oddzielny próg górny oraz dolny dla każdego z kanałów tak aby na zdjęciu wyróżnić kostki żółte. Dla zdjęć z góry są to wartości:

- barwa: od 0.1 do 0.17
- nasycenie koloru: od 0.7 do 0.85
- moc światła białego: od 0.3 do 0.82

Jeśli w danym kanale wartość danego pikselu mieści się w przedziale jest mu przypisywana wartość 1. W przeciwnym wypadku wartość 0.



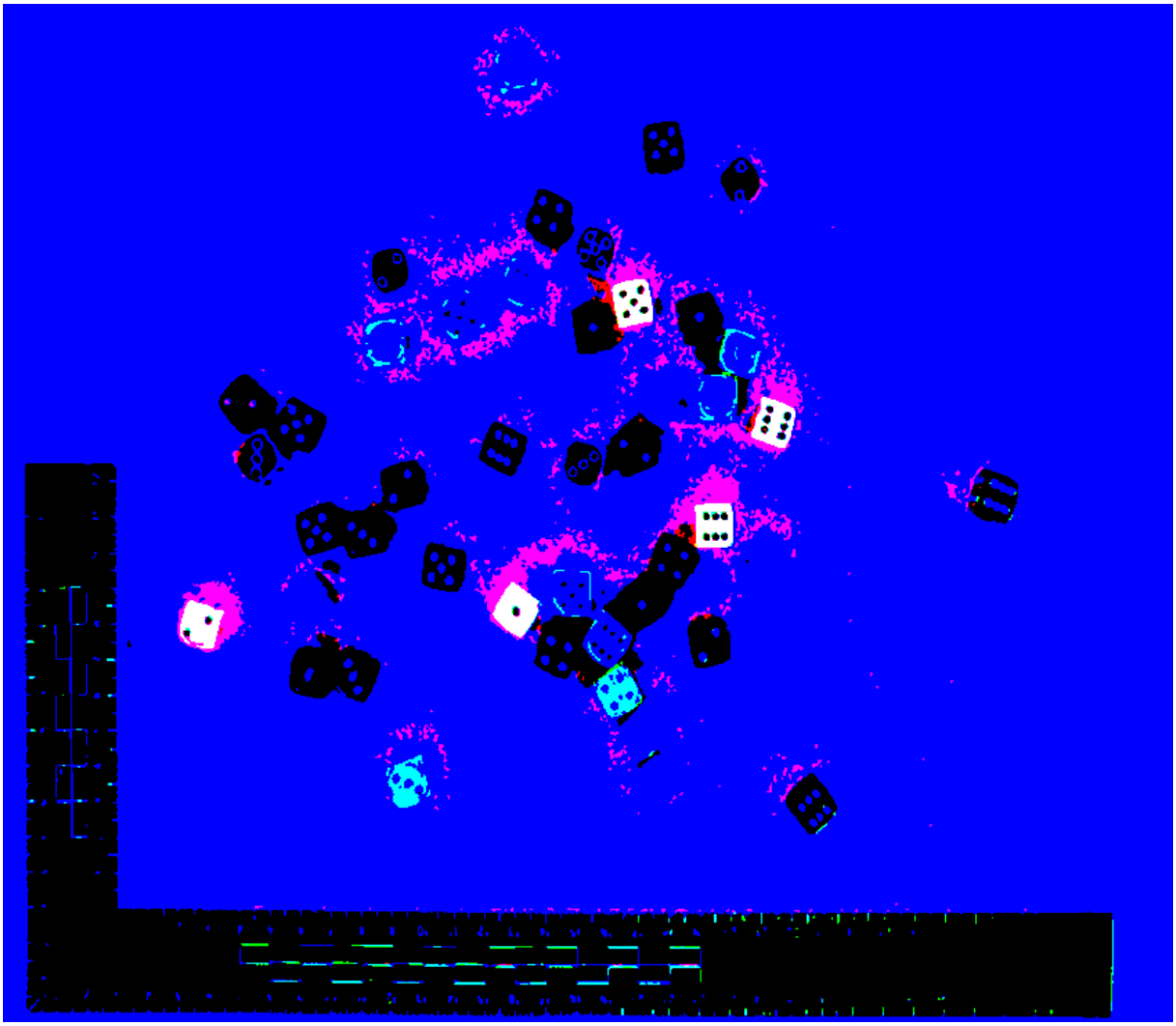
Rys. topdown_02.png – po progowaniu.

Kostki żółte widoczne są na zdjęciu na biało. Jest to spowodowane tym, że po progowaniu jedynie te kostki znalazły się we wszystkich trzech kanałach.

Jak widać elementy o innych kolorach niż żółty również znajdują się na zdjęciu jednak są obecne najwyżej w jednym lub dwóch kanałach.

3. Usuwanie szumów

Jak widać wyraźnie na zdjęciu po przeprowadzeniu progowania, występują liczne szumy typu „sól i pieprz” które nie usunięte byłyby widoczne po przeprowadzeniu binaryzacji w następnym kroku. W celu usunięcia szumów zastosowano filtr medianowy. Zdjęciu poddano filtracji ośmiokrotnie, aby usunąć również nieco większe szumy oraz „wygładzić kostki” posiadające drobne defekty.

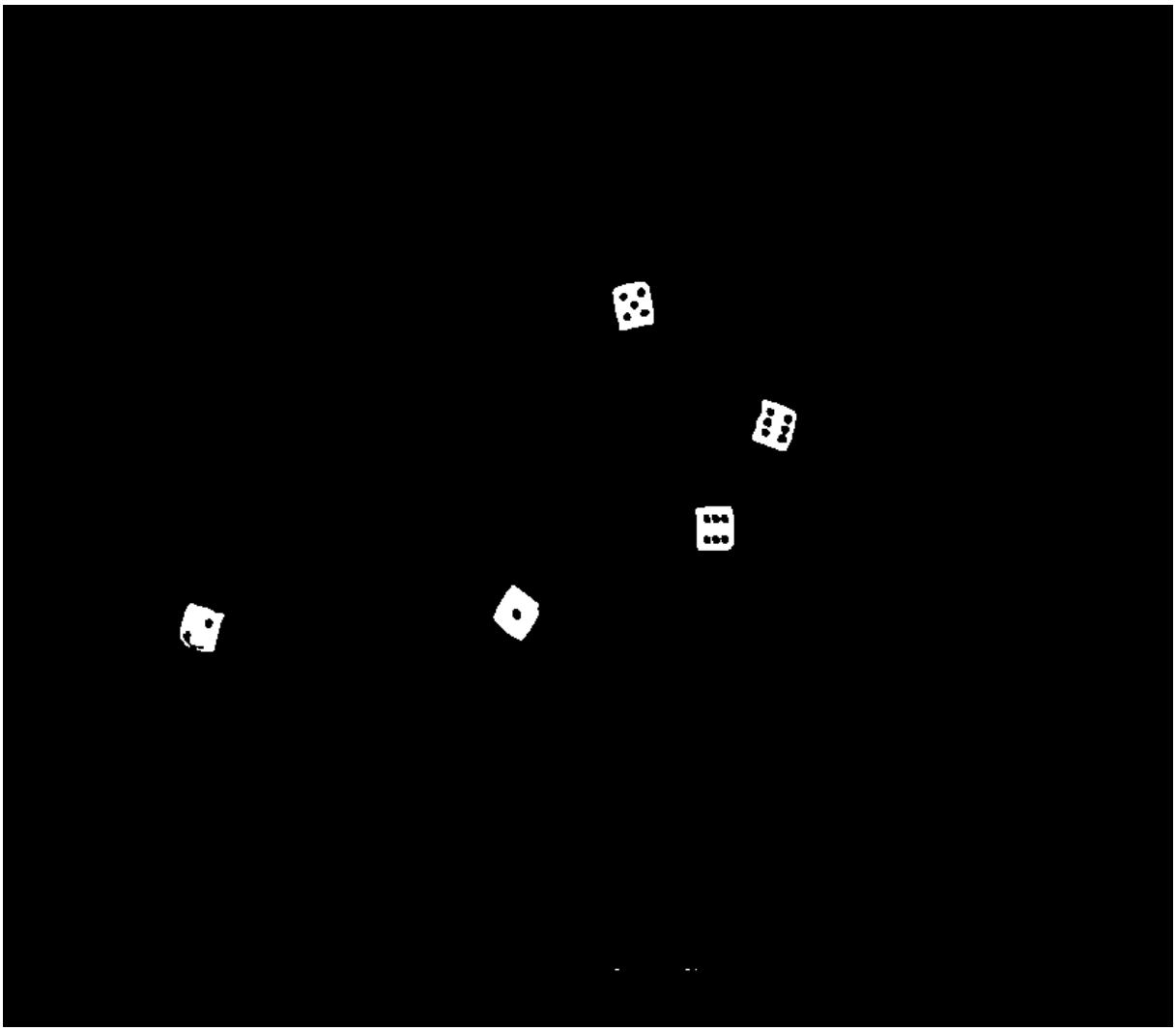


Rys. topdown_02.png – po filtracji filtrem medianowym.

Jak widać filtr medianowy skutecznie usunął szumy tła oraz ujednolicił kostki, które posiadały drobne defekty spowodowane niedoskonałym dobraniem wartości podczas progowania. Filtracja nie wpłynęła natomiast na kształt kostek oraz ilość ich oczek co jest pożądanym rezultatem.

4. Binaryzacja

Aby na zdjęciu pozostały jedynie kostki żółte należy przeprowadzić binaryzację. Jeśli piksel ma wartość 1 we wszystkich trzech kanałach wówczas zostaje on na zdjęciu. W przeciwnym razie jest on usuwany.

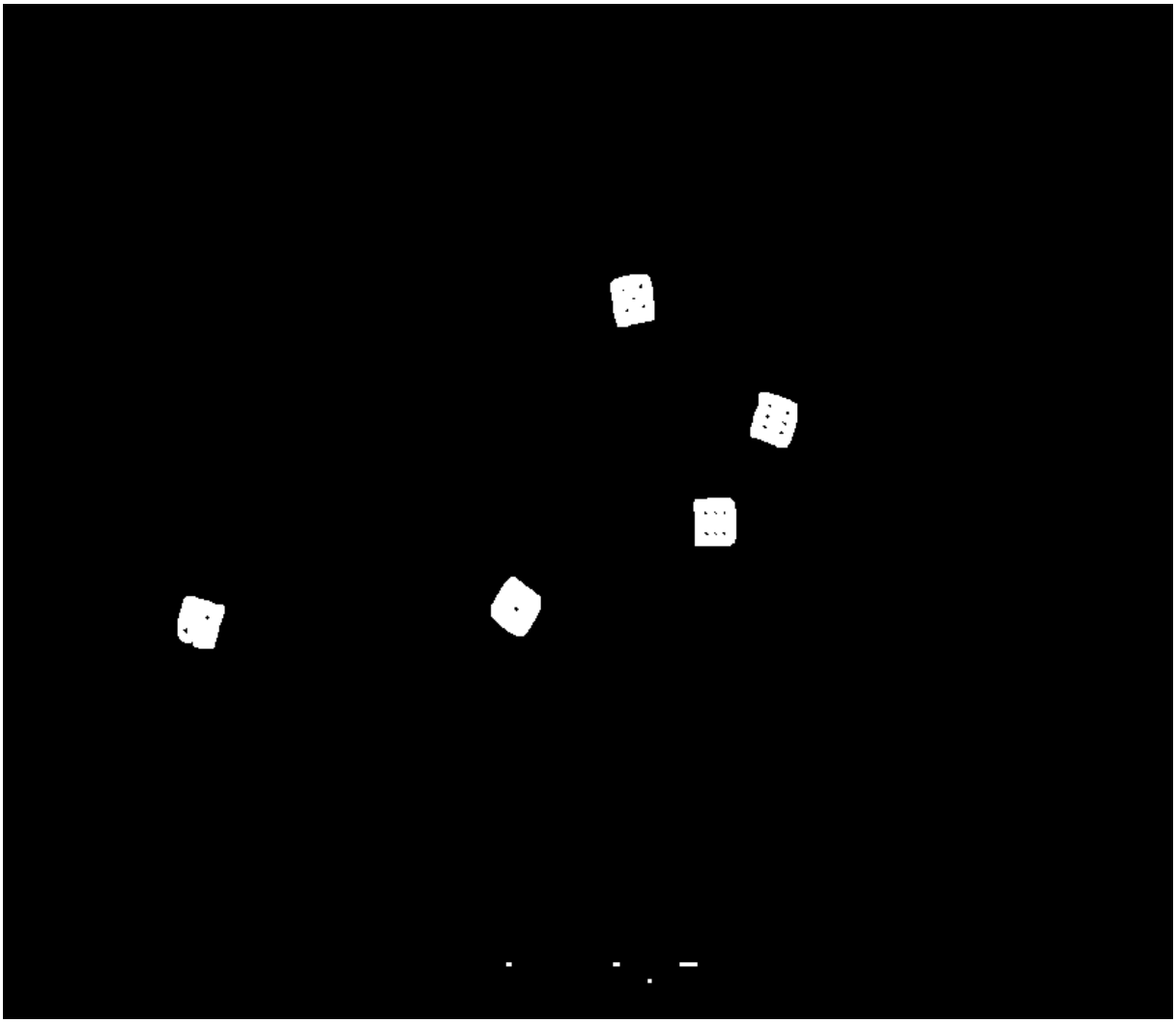


Rys. topdown_02.png – po binaryzacji.

Na zdjęciu pozostały jedynie kostki pożądanego przez nas koloru. W tle widać jedynie kilka pikseli szumu.

5. Dylatacja

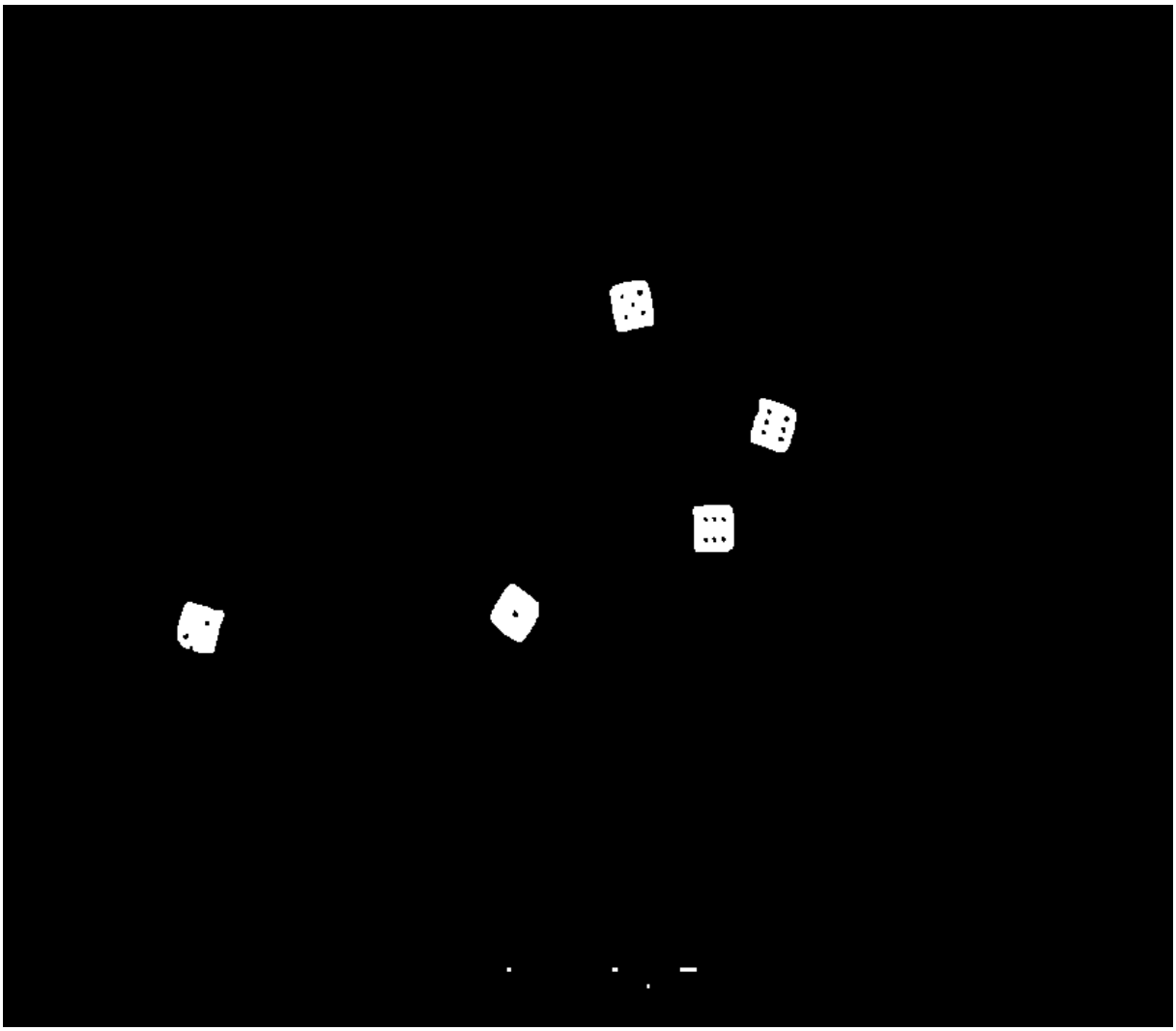
Jak widać na poprzednim zdjęciu część z kostek posiada wady w postaci „sklejenia” ze sobą oczek lub „poszarpania” kształtu kostki co może prowadzić do błędnego rozpoznania ilości oczek. Dylatacja pozwoli „zasklepić” te wady. Aby wyeliminować nawet duże wady dylatacja została przeprowadzona trzykrotnie (większa ilość dylatacji doprowadziłaby do całkowitego zaniku oczek).



Rys. topdown_02.png – po dylatacji.

6. Erozja

Po tak silnej dylatacji przeprowadzonej w poprzednim kroku oczka na kostkach stały się bardzo małe. Aby je powiększyć przeprowadzana jest erozja. Zmniejszą się również rozmiary szumów które zostały wyeksponowane w poprzednim kroku.



Rys. topdown_02.png – po erozji.

7. Oznaczanie kostek

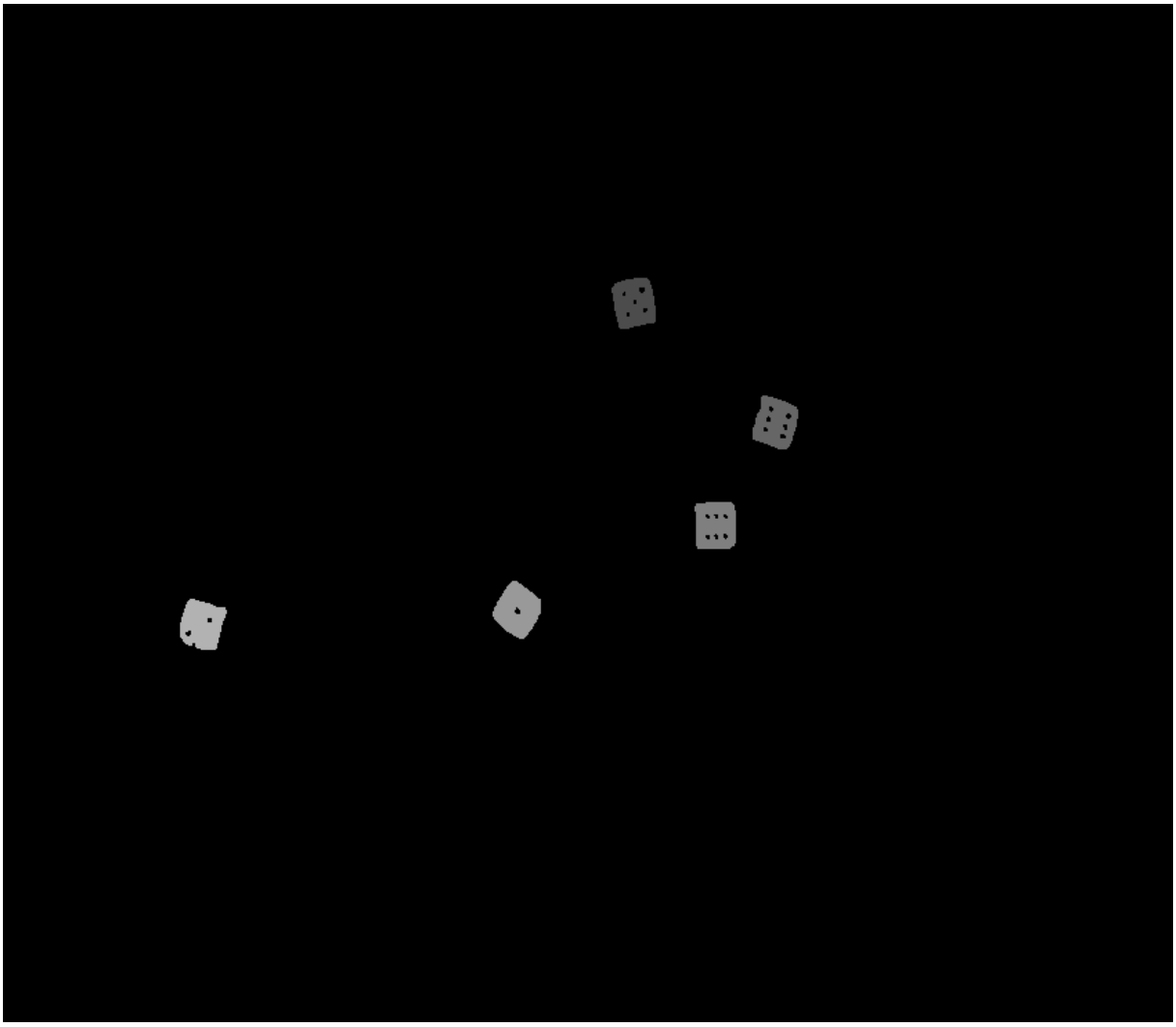
Proces ten polega na odróżnianiu kostek od tła. Każdej z kostek przypisana zostaje inna etykieta (inny kolor).

Przed zastosowaniem algorytmu wyczyszczono kanał drugi obrazu, aby móc w nim później zapisywać etykiety kostek.

Algorytm wykorzystuje do swojego działania stos (std::stack).

Działanie algorytmu:

- krok 1. Zacznij od pierwszego piksela obrazu. Ustaw obecną etykietę na początkową wartość. Przejdź do kroku 2.
- krok 2. Jeśli dany piksel jest kostką (ma wartość 1) I nie został jeszcze oznaczony, nadaj mu obecną etykietę i dodaj jego pozycję jako pierwszy element do stosu, następnie przejdź do kroku 3. Jeśli natomiast piksel jest tłem (ma wartość 0) powtórz krok 2 dla kolejnego piksela.
- krok 3. Pobierz pozycję piksela ze stosu i przeszukaj jego sąsiadów. Jeśli sąsiad jest pikselem kostki (ma wartość 1) i nie został jeszcze oznaczony nadaj mu obecną etykietę i dodaj jego pozycję do stosu. Powtarzaj krok 3 aż nie będzie więcej elementów na stosie.
- krok 4. Zwiększ wartość obecnej etykiety i przejdź do kroku 2.

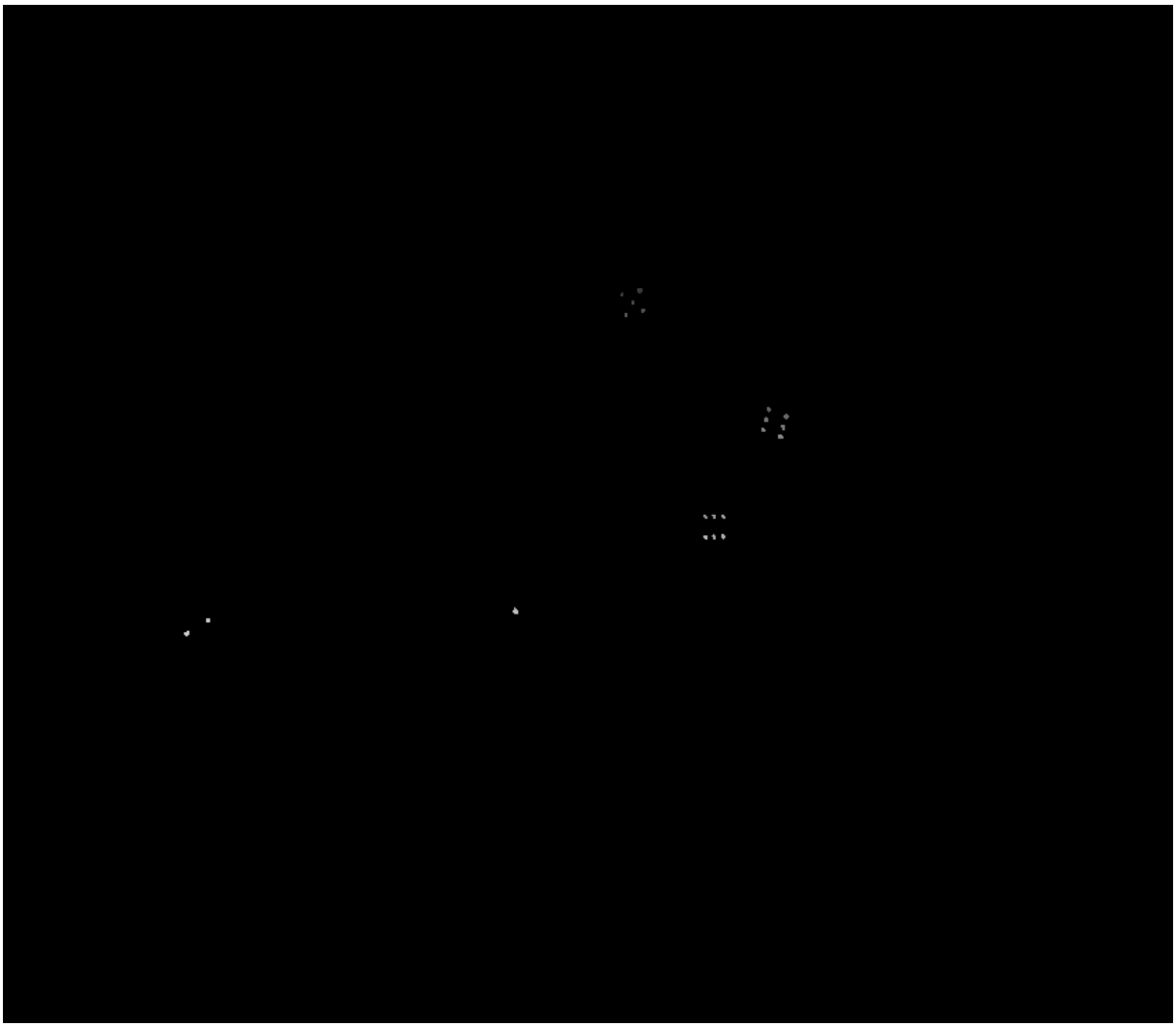


Rys. topdown_02.png (kanał drugi) – po oznaczeniu kostek.

Etykiety zostały zapisane jako różne wartości w kanale drugim obrazu. Jak widać, po takiej operacji każda z kostek ma inny kolor. W algorytmie zaimplementowano próg minimalnego rozmiaru kostki na 400 pikseli, tak aby szumy tła nie zostały oznaczone jako kostki.

8. Oznaczanie oczek

Aby móc policzyć oczka należy przypisać każdemu z oczek inną etykietę, analogicznie jak w poprzednim kroku było to robione z kostkami. W algorytmie zaimplementowano próg maksymalnego rozmiaru oczka na 160000 pikseli, tak aby tło nie zostało oznaczone jako oczko. Etykiety oczek zostały zapisane w kanale trzecim obrazu.



Rys. topdown_02.png (kanał trzeci) – po oznaczeniu oczek.

9. Dopasowanie oczek do kostki

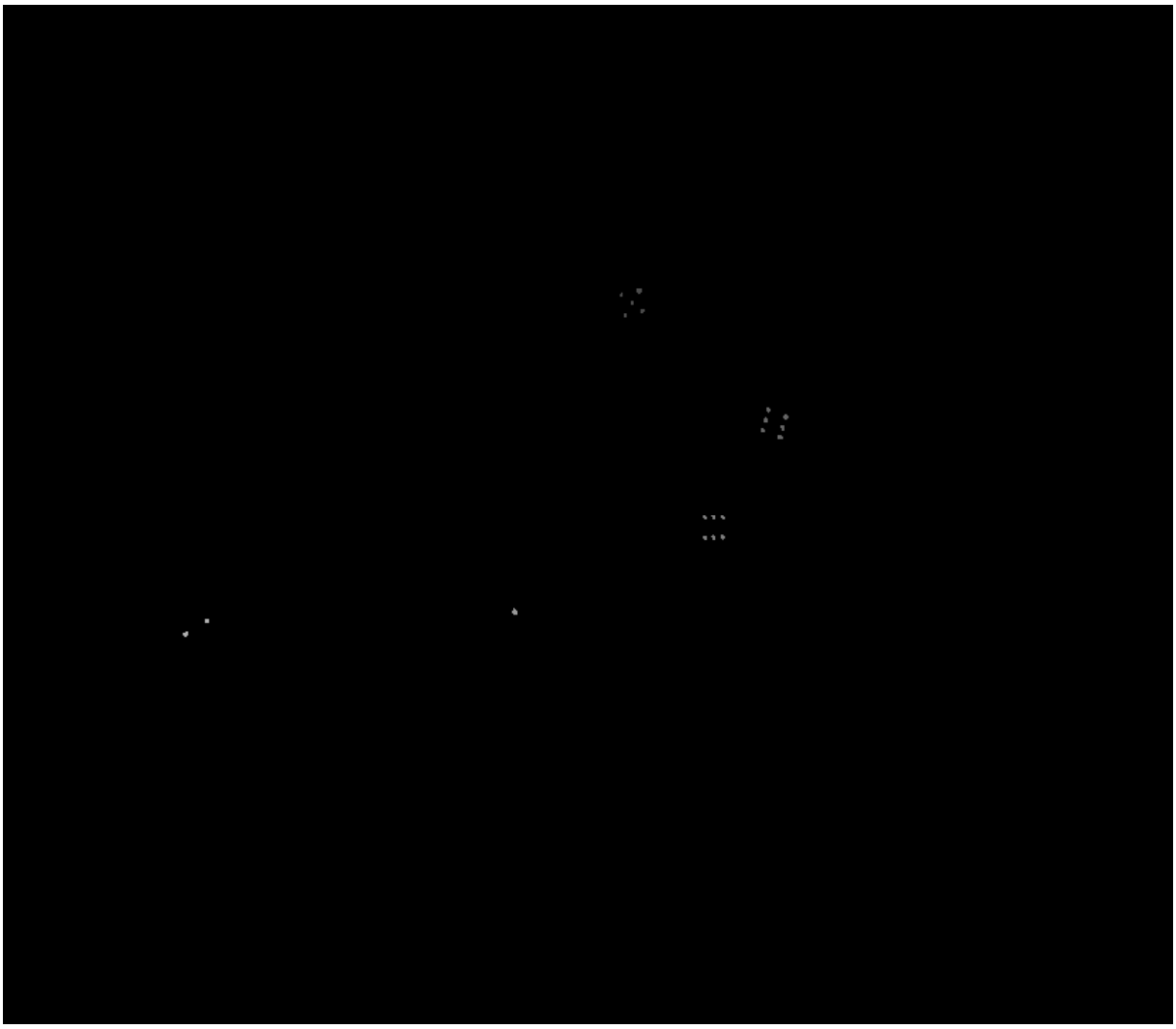
Kolejnym etapem jest przyporządkowanie oczek do kostek. Dzięki temu możliwe jest stwierdzenie ile dana kostka ma wyrzuconych oczek, aby można było zsumować jedynie oczka nieparzyste.

Oczka zapisane są w trzecim kanale, natomiast kostki w drugim. Etykiety oczka-kostka zapisywane będą na kanale pierwszym, należy go więc na początku wyczyścić. Działanie algorytmu:

-przeszukuj kanał trzeci aż natrafisz na piksel oczka.

-od pozycji znalezionego oczka poruszaj się po kanale drugim w jednym kierunku aż natrafisz na piksel kostki (jest to zawsze słuszne ponieważ oczka znajdują się wewnątrz kostek).

-przypisz pikselowi na pozycji oczka etykietę kostki na kanale pierwszym.



Rys. topdown_02.png (kanał pierwszy) – po dopasowaniu oczek do kostek.

Jak widać na zdjęciu oczka tej samej kostki mają jednakowe etykiety.

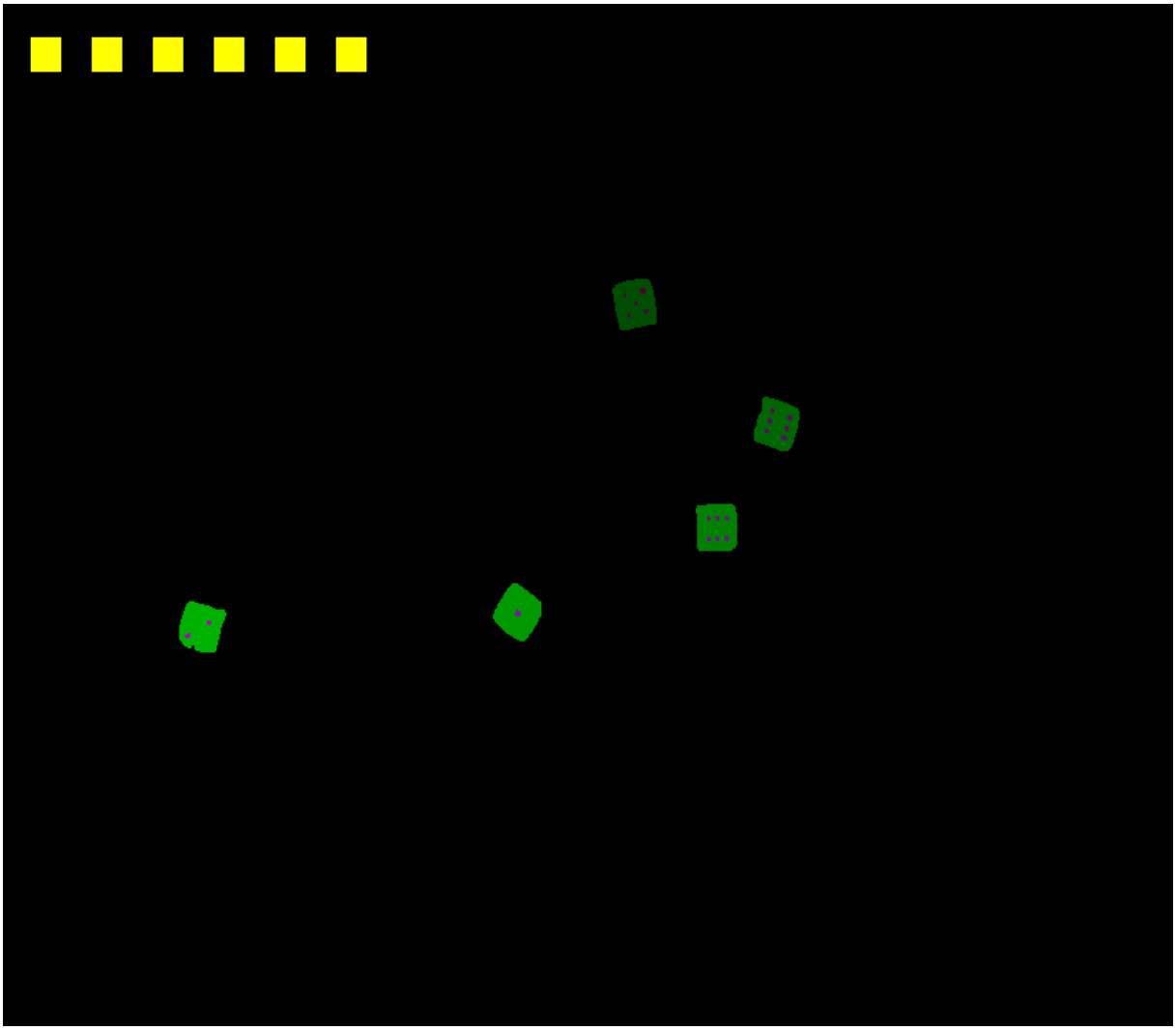
10. Obliczanie sumy wyrzuconych oczek nieparzystych

W celu obliczenia sumy oczek nieparzystych zaimplementowano dwie mapy (std::map):
-pierwsza mapa przechowuje informację czy dane oczko zostało już policzone
-druga mapa przechowuje informację na temat sumy oczek poszczególnych kostek

Algorytm przeszukuje kanał trzeci. W przypadku gdy natrafi na piksel oczka sprawdza w pierwszej mapie czy dane oczko nie zostało jeszcze policzone. Jeśli nie było policzone, sprawdza jaką etykietę kostki ma dane oczko na kanale pierwszym. Następnie zwiększa o jeden ilość oczek danej kostki w mapie drugiej. Na koniec w mapie pierwszej ustawia flagę, że dane oczko zostało policzone.

11. Rysowanie wyniku

Obliczona suma oczek nieparzystych przedstawiana jest na obrazie w postaci żółtych kwadracików rysowanych na obrazie w lewym górnym rogu.

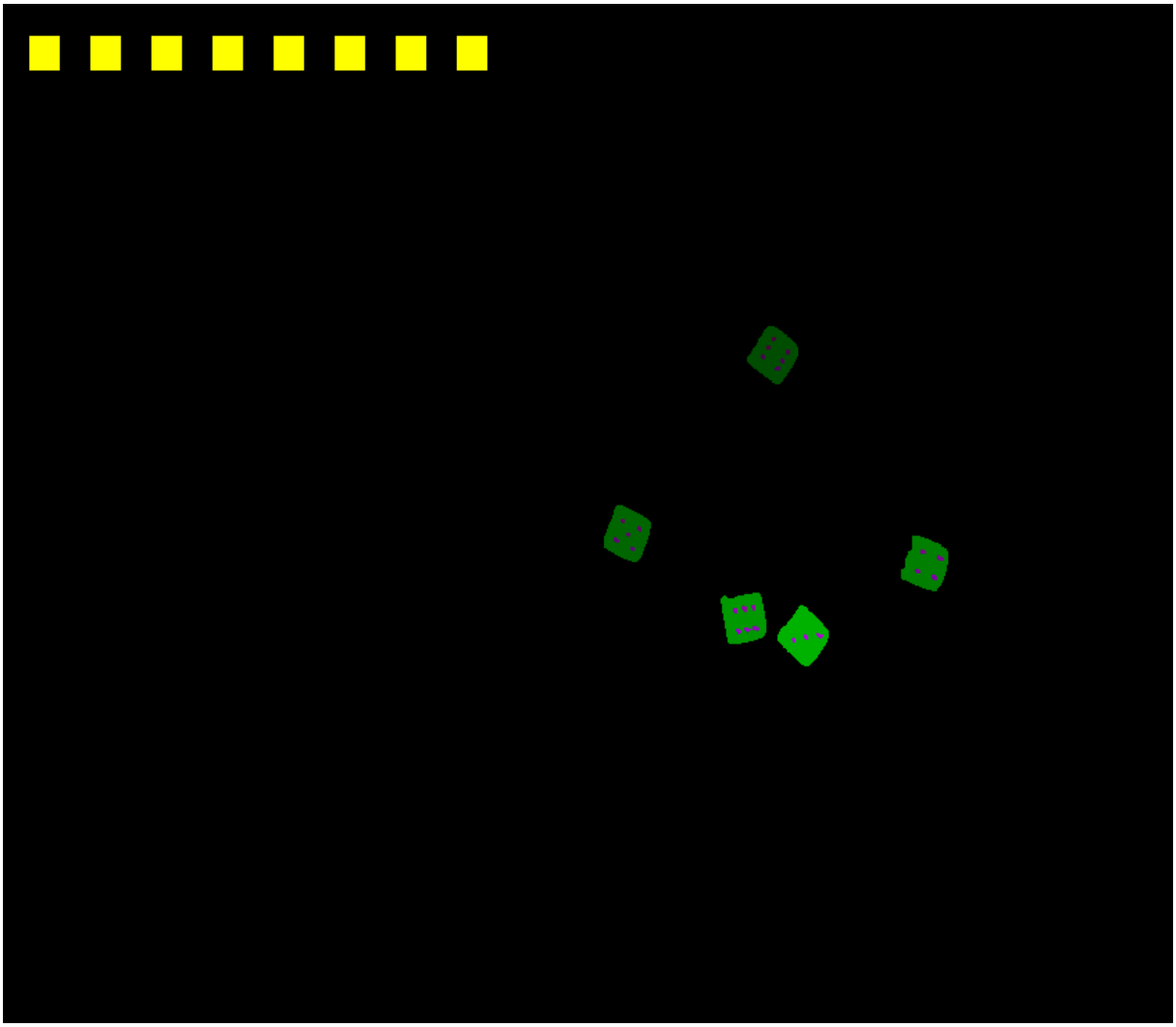


Rys. topdown_02.png – przedstawienie wyniku

12. Podsumowanie

Algorytm bez żadnych modyfikacji działa poprawnie również dla drugiego zdjęcia (widok od góry).

Rys. topdown_17.png – zdjęcie oryginalne.



Rys. topdown_17.png – obraz końcowy po zastosowaniu wszystkich kroków

II Zdjęcia z widoku z perspektywy

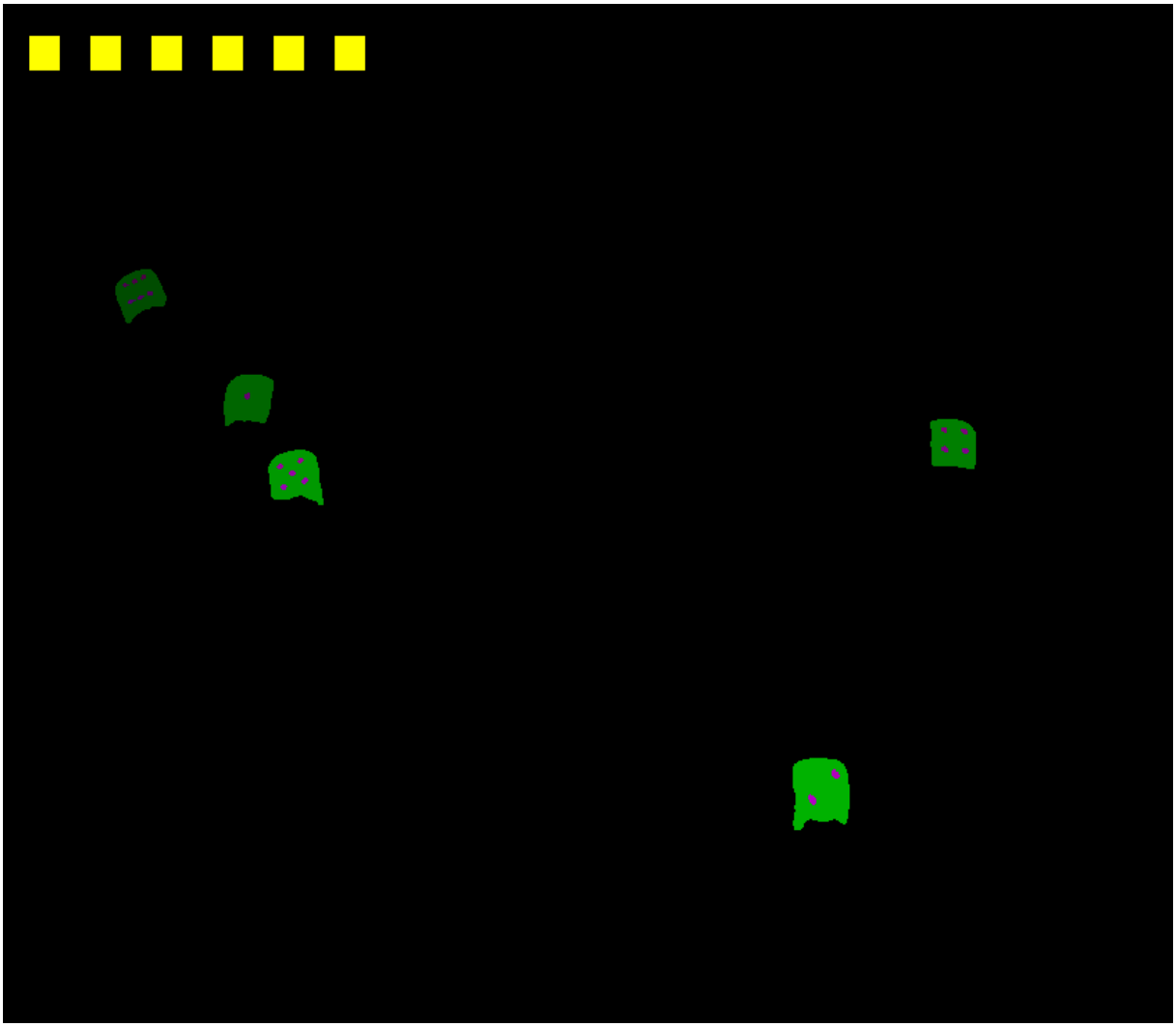
Algorytm postępowania dla zdjęć z perspektywy wykorzystuje te same narzędzia co algorytm dla zdjęć z widoku od góry. Różni się on m.in. doborem wartości progów podczas progowania kanałów (należało dobrać takie wartości aby widoczne boczne ścianki kostek zostały odrzucone) oraz ilością przeprowadzonych operacji filtracji i dylatacji.

Algorytm postępowania:

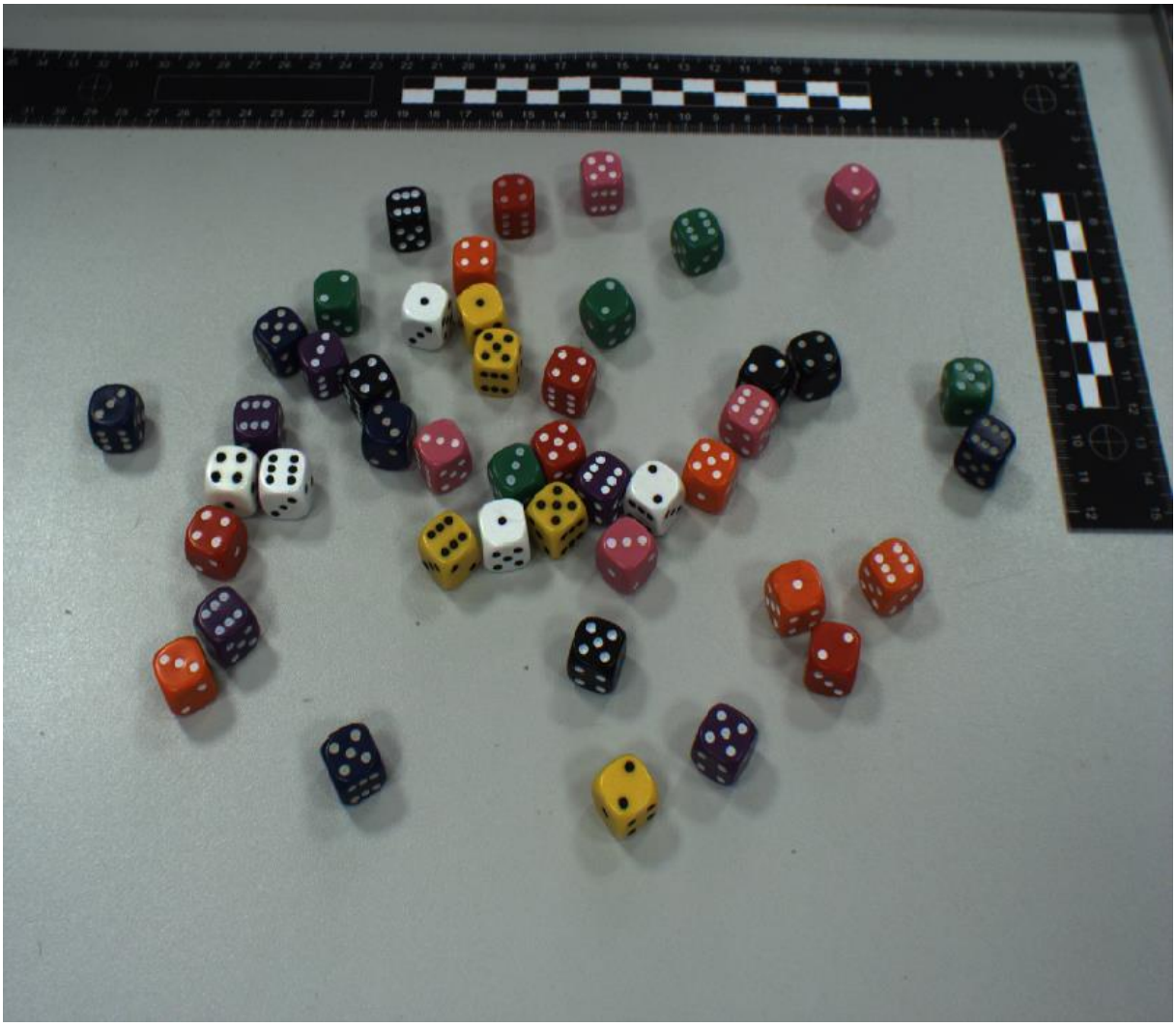
-przejście z przestrzeni barw RGB na HSV
-progowanie kanałów:
    kanał 1 (barwa): od 0.1 do 0.17
    kanał 2 (nasycenie kolorów): od 0.5 do 0.85
    kanał 3 (moc światła białego): od 0.41 do 0.82
-binaryzacja
-filtr medianowy (6 razy)
-dylatacja(2 razy)
-filtr medianowy (1 raz)

Dodatkowa filtracja filtrem medianowym po dylatacji pomaga w rozdzielaniu „sklejonych” oczek (przeprowadzenie trzykrotnej dylatacji, tak jak w przypadku zdjęć w widoku od góry, powodowało w przypadku zdjęć z perspektywy całkowite zasklepienie oczek).

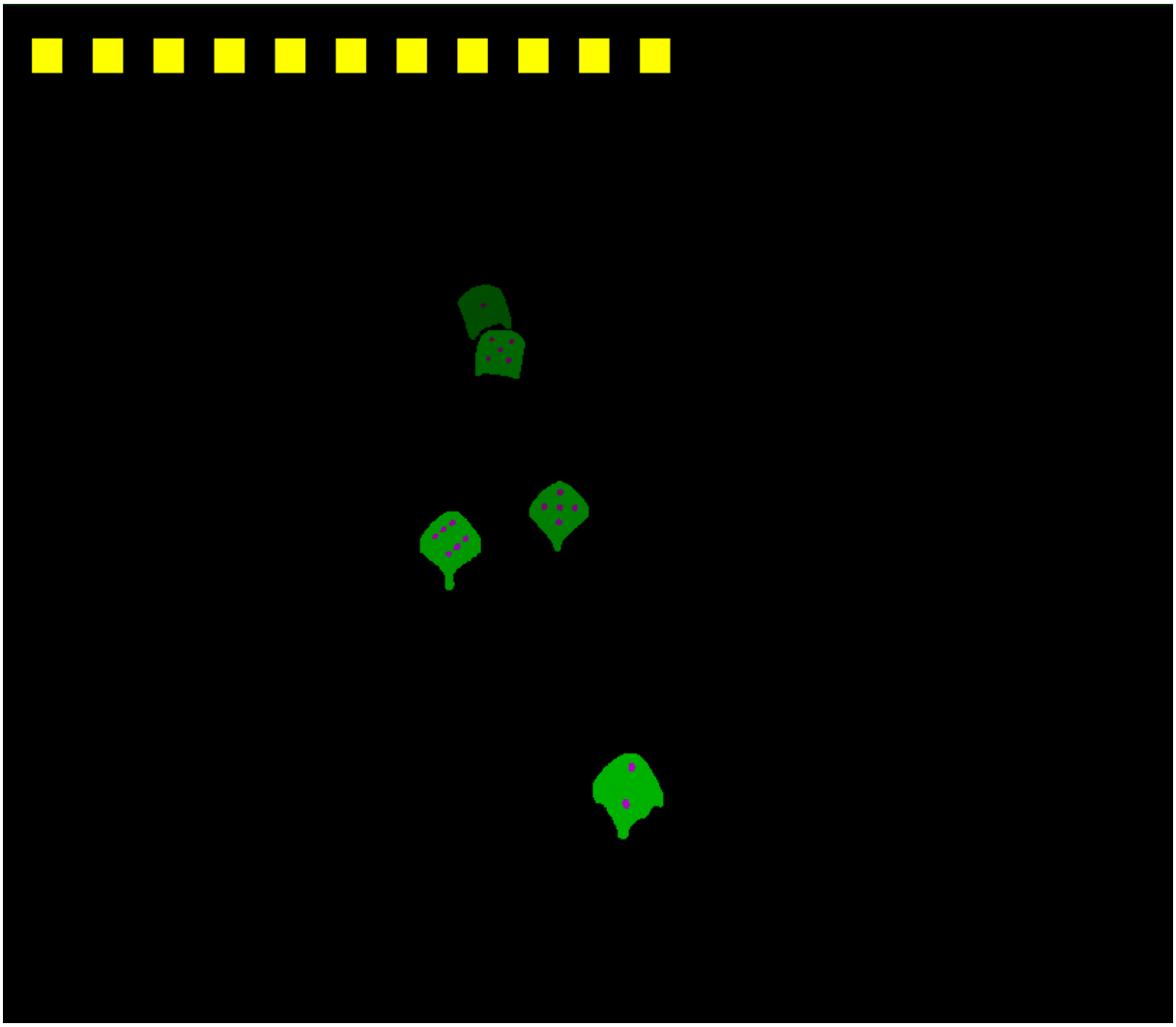
Rys. perspective_02.png – zdjęcie oryginalne



Rys. perspective_02.png – obraz końcowy po zastosowaniu wszystkich kroków



Rys. perspective_17.png – zdjęcie oryginalne



Rys. perspective_17.png – obraz końcowy po zastosowaniu wszystkich kroków