**Przemysłowy system wizyjny do detekcji drobnych przedmiotów złotniczych i jubilerskich**

**Grupa laboratoryjna 3**

Stanisław Horna **241202**

Przemysław Kowalski **241214**

Spis treści

[**1.** **Temat projektu** 3](#_Toc166012102)

[**2.** **Wstęp** 3](#_Toc166012103)

[**3.** **Materiały i metody** 3](#_Toc166012104)

[3.1. Opis stanowiska symulującego taśmociąg 3](#_Toc166012105)

[**3.1.1. Taśmociąg** 3](#_Toc166012106)

[**3.1.2. Kamera** 3](#_Toc166012107)

[**3.1.3. Oświetlenie** 4](#_Toc166012108)

[**3.1.4. Przedmioty** 4](#_Toc166012109)

[3.2. Metody 4](#_Toc166012110)

[**3.2.1 Wprowadzanie obrazu** 4](#_Toc166012111)

[**3.2.2 Transformacja obrazu** 5](#_Toc166012112)

[**3.2.3 Detekcja obiektów** 5](#_Toc166012113)

[**3.2.4 Zliczanie obiektów** 5](#_Toc166012114)

[**4.** **Wyniki** 5](#_Toc166012115)

[**5.** **Wnioski** 5](#_Toc166012116)

[**6. Literatura** 5](#_Toc166012117)

# **Temat projektu**

Przemysłowy system wizyjny, który w czasie rzeczywistym, wykona detekcję, a następnie klasyfikację wyrobów jubilerskich umieszczonych na przemieszczającym taśmociągu.

# **Wstęp**

Do realizacji projektu zostało przygotowane m.in. stanowisko (taśmociąg) oraz wykorzystano wiele różnych metod przetwarzania obrazu jak:

* progowanie [[1]](https://pl.wikipedia.org/wiki/Progowanie_obrazu),
* rozmycie metodą Gaussa [[2]](https://pl.wikipedia.org/wiki/Rozmycie_gaussowskie),
* detekcja krawędzi metodą Canny [[3]](https://pl.wikipedia.org/wiki/Canny),
* morfologiczne domknięcie krawędzi [[4]](https://pl.wikipedia.org/wiki/Cyfrowe_przetwarzanie_obraz%C3%B3w_binarnych#Operacje_morfologiczne),
* detekcja obiektów [[5]](https://pl.wikipedia.org/wiki/Detekcja_obiekt%C3%B3w).

Szczegółowe opisy znajdują się w dalszej części raportu.

# **Materiały i metody**

## 3.1. Opis stanowiska symulującego taśmociąg

### **3.1.1. Taśmociąg**

Drobne przedmioty złotnicze i jubilerskie będą wrzucane na taśmociąg, który będzie poruszał się ze stałą prędkością. Pas transmisyjny taśmociągu powinien być gładki wykonany z gumy lub papieru w kolorze białym (*#ffffff* w systemie heksadecymalnym).

### **3.1.2. Kamera**

Zainstalowana kamera będzie rejestrować przedmioty transportowane przez taśmociąg pod kątem 90° z odległości 30 cm, która nagrywa minimum w rozdzielczości 1920x1080 (FullHD) oraz 30 klatkach na sekundę (30fps). Obraz powinien zostać odpowiednio pomniejszony przed rozpoczęciem analizy poszczególnych klatek.

### **3.1.3. Oświetlenie**

Taśmociąg będzie oświetlony dwiema lampami generującymi **8550-9000lm**. Lampy będą ustawione w odległości **70cm** od taśmy przesuwającej przedmioty pod kątem **45°**

### **3.1.4. Przedmioty**

Przedmioty wrzucane na taśmę to drobne przedmioty złotnicze i jubilerskie w kategoriach:

* naszyjniki,
* pierścionki,
* kolczyki.

Przedmioty wpadają na taśmociąg w sposób, który umożliwia ich przyleganie do siebie, ale nie nakładanie się jednego przedmiotu na drugi, czy umieszczanie jednego w drugim.

## 3.2. Metody

Obraz zawierający linia, biały

Opis wygenerowany automatycznie

### **3.2.1 Wprowadzanie obrazu**

Program umożliwia przechwytywanie obrazu z kamery, lub w celach testowych oraz rozwojowych, odtworzenie pliku wideo ze wskazanej ścieżki. Do przechwytywania obrazu wykorzystujemy klasę *VideoCapture* z biblioteki *OpenCV*.   
W programie wykorzystujemy konstruktor klasy *VideoCapture,* który pozwala na przekazanie jako argument indeks urządzenia, z którego chcemy przechwytywać obraz, lub ścieżkę do pliku z nagranym materiałem wideo.  
Po pomyślnym utworzeniu instancji klasy *VideoCapture* zmniejszamy rozdzielczość przetwarzanego obrazu do rozmiaru 480x320 pixeli, który pozwala uzyskać większą wydajność programu, oraz w zupełności wystarcza do poprawnej identyfikacji obiektów.

### **3.2.2 Transformacja obrazu**

Obraz zawierający diagram, tekst, linia, zrzut ekranu

Opis wygenerowany automatycznie

W pierwszym kroku zamieniamy kolorową ramkę na odcienie szarości, przy użyciu funkcji *cvtColor* z biblioteki *OpenCV*.  
Następnie stosujemy filtr Gaussa korzystając z metody *GausianBlur* z biblioteki *OpenCV*, dzięki czemu redukujemy szumy oraz niepotrzebne szczegóły na klatce obrazu. Za pomocą parametru *ksize* definiujemy wysokość i szerokość jądra na 5, a odchylenie standardowe w obu osiach (*sigmaX* i *sigmaY*) na 1.  
Ostatnim wspólnym krokiem w transformacji obrazu dla wszystkich typów obiektów jest detekcja krawędzi algorytmem Canny’ego, na podstawie gradientów jasności pikseli. Korzystamy z funkcji *Canny* z biblioteki *OpenCV*, ustawiając minimalną wartość gradientu (*threshold1*) na 60, a wartość maksymalną (*threshold2*) na 255.

W następnych krokach zdecydowaliśmy się na przygotowywanie dwóch różnych ramek, na podstawie których będziemy rozpoznawać przedmioty z poszczególnych kategorii, ze względu na odbicia światła, które zaburzały kolejne kroki bazujące na dokładnym domknięciu krawędzi obserwowanych obiektów.

Do detekcji kolczyków i naszyjników wykorzystujemy morfologiczne domknięcie krawędzi, przy użyciu funkcji *morphology.closing* z biblioteki *skimage*, aby uzupełnić niedoskonałości w wyniku działania funkcji *Canny*. Jako maskę (*footprint*) dla operacji domknięcia stosujemy promień o wartości 2 (*morphology.disk()*).  
Następnie przy użyciu metody *findContours* z biblioteki *OpenCV* ponownie odnajdujemy domknięte już krawędzie w trybie *cv2.RETR\_LIST*, który nie zwraca uwagi na hierarchię konturów, korzystając z metody *cv2.CHAIN\_APPROX\_SIMPLE*, pozwalającej na kompresję poziomów konturów do ich punktów końcowych. Odnalezione kontury wypełniamy kolorem czarnym za pomocą funkcji *drawContours* z biblioteki *OpenCV*, ustawiając parametr *contourIdx* na wartość -1 oraz podając każdy kontur jako listę, dzięki czemu wszystkie znalezione kontury zostaną wypełnione.

W przypadku odnajdywania pierścionków kluczowe jest wykrycie wewnętrznego okręgu (miejsca na palec osoby, która go nosi). Funkcja *morphology.closing* z biblioteki *skimage* z maską (*footprint*) ustawioną na promień o wartości 4 (*morphology.disk()*) domknie większe odległości pomiędzy niedomkniętymi konturami wewnętrznych okręgów. Zastosowanie odrębnej operacji dla pierścionków jest konieczne, ze względu na refleksy powstające na wąskiej krawędzi metalu biżuterii, co uniemożliwiało domknięcie okręgów korzystając z mniejszego promienia dysku.

### **3.2.3 Detekcja obiektów**

Wszystkie obiekty są wykrywane poprzez *SimpleBlobDetector*[[6]](https://learnopencv.com/blob-detection-using-opencv-python-c/) wbudowaną w bibliotekę *OpenCV*. *SimpleBlobDetector* pozwala na detekcję *Blob* czyli grupy połączonych pikseli na obrazie, które mają podobne właściwości np. kolor. *SimpleBlobDetector* wykonuje detekcję na podstawie m.in.:

* rozmiaru zajmowanego obszaru (area),
* wypukłości (convexity),
* inercji (inertia),
* kolistości (circularity),
* koloru (color)

a następnie oznacza znalezione regiony.

Dla wszystkich obiektów wartość parametru *blobColor* powinna być ustawiona na wartość 0 (czerń), gdyż po wstępnej transformacji obraz jest w odcieniach czerni.

Naszyjniki zajmują najwięcej obszaru oraz mają kształt kulisty. Najlepsze wyniki uzyskaliśmy po odpowiednio ustawionych parametrach:

* *minArea* na 100 000, który wykonuje detekcję elementów o **minimalnej** powierzchni równej 100 000 jednostek kwadratowych,
* *maxArea*na10 000 000, który wykonuje detekcję elementów o **maksymalnej** powierzchni równej 10 000 000 jednostek kwadratowych,
* *minCircularity* na 0.25, który wykonuje detekcję elementów zbliżonych do kształtu czworokąta,
* *maxCircularity* na 0.9, który wykonuje detekcję elementów zbliżonych do kształtu okręgu.

Pozostałe parametry typu *convexity (wypukłość)* i *inertia (inercja)* są niepotrzebne, gdyż w naszym przypadku naszyjniki zajmują najwięcej powierzchni.

Pierścionki mają również kształt kulisty oraz zajmują mniej obszaru niż naszyjniki. *SimpleBlobDetector* [[6]](https://learnopencv.com/blob-detection-using-opencv-python-c/) wykonuje detekcję pierścionków na podstawie okręgu wewnętrznego okręgu (miejsca na palec osoby, która go nosi). Dla pierścionków wykorzystujemy wszystkie dostępne parametry detekcji obiektów w *SimpleBlobDetector* [[6]](https://learnopencv.com/blob-detection-using-opencv-python-c/). Najlepsze wyniki uzyskujemy dla ustawień, które identyfikują obiekty najbardziej zbliżone do kształtu kulistego:

* *minArea*=7000,
* *maxArea*=12000,
* *minCircularity*=0.6,
* *maxCircularity*=1,
* *minConvexity*=0.6, który pozwala na detekcję obiektów o **minimalnej** wypukłości,
* *maxConvexity*=1, ,który pozwala na detekcję obiektów o **maksymalnej** wypukłości
* *minInertiaRatio*=0.5, który pozwala na detekcję obiektów o **minimalnej** inercji,
* *maxInertiaRatio*=1, który pozwala na detekcję obiektów o **maksymalnej** inercji,

Kolczyki mają różne kształty oraz zajmują jeszcze mniej obszaru niż pierścionki. *SimpleBlobDetector* [[6]](https://learnopencv.com/blob-detection-using-opencv-python-c/) wykonuje detekcję kolczyków na podstawie kształtu oraz inercji. Dla kolczyków pomijamy detekcję poprzez parametr *convexity (wypukłość)* w *SimpleBlobDetector* [[6]](https://learnopencv.com/blob-detection-using-opencv-python-c/). Najlepsze wyniki uzyskujemy dla ustawień:

* *minArea*=1200,
* *maxArea*=10900,
* *minCircularity*=0.35,
* *maxCircularity*=0.80,
* *minInertiaRatio*=0.12,
* *maxInertiaRatio*=0.90,

### **3.2.4 Zliczanie obiektów**

# **Wyniki**

Dla obiektów typu pierścionki wyniki były najlepsze. Skuteczność detekcji wynosi ok.90%. Dla kolczyków jest to ok. 80% natomiast dla naszyjników ok. 60%.

# **Wnioski**

Drobne przedmioty złotnicze i jubilerskie odbijają światło co początkowo utrudniało prawidłową detekcję obiektów. Dzięki metodom takimi jak progowanie [[1]](https://pl.wikipedia.org/wiki/Progowanie_obrazu), rozmycie obrazu metodą Gaussa [[2]](https://pl.wikipedia.org/wiki/Rozmycie_gaussowskie), detekcje krawędzi metodą Canny [[3]](https://pl.wikipedia.org/wiki/Canny) oraz morfologiczne domknięcie krawędzi [[4]](https://pl.wikipedia.org/wiki/Cyfrowe_przetwarzanie_obraz%C3%B3w_binarnych#Operacje_morfologiczne) uzyskaliśmy obraz, który miał zniwelowane efekty odbicia światła. Na bazie zmodyfikowanego obrazu mogliśmy wykonać właściwą detekcję obiektów. Rezultaty naszej metody detekcji obiektów są bardziej skuteczne przy lepszym oświetleniu stanowiska.

# **6.** **Literatura**

[[1] Progowanie](https://pl.wikipedia.org/wiki/Progowanie_obrazu)

[[2] Rozmycie metodą Gauss’a](https://pl.wikipedia.org/wiki/Rozmycie_gaussowskie)

[[3] Detekcja krawędzi metodą Canny](https://pl.wikipedia.org/wiki/Canny)

[[4] Morfologiczne domknięcie krawędzi](https://pl.wikipedia.org/wiki/Cyfrowe_przetwarzanie_obraz%C3%B3w_binarnych#Operacje_morfologiczne)

[[5] Detekcja obiektów](https://pl.wikipedia.org/wiki/Detekcja_obiekt%C3%B3w)

[[6] SimpleBlobDetector](https://learnopencv.com/blob-detection-using-opencv-python-c/)