FAKULTAS ILMU KOMPUTER
UBP KARAWANG

# DETEKSI KOIN EMAS DAN PERAK MENGGUNAKAN
## Algoritma Support Vector Machine ( SVM )


Authors:

Bayu Prasetyo (21416255201070)

Arief Maulana (21416255201049)

Anto Kuswanto (21416255201171)

Muhamad Ikbal Ramdani (21416255201035)

FAKULTAS ILMU KOMPUTER
UBP KARAWANG

UBP KARAWANG

- Pendahuluan
- Metode
- Hasil
- Kesimpulan

# Pendahuluan

Mendeteksi koin merupakan topik yang menarik dalam pengolahan citra digital. Dengan fitur ini, memungkinkan sistem untuk secara otomatis mengidentifikasi jenis koin dan memberi keunggulan dalam hal efisiensi dan akurasi, termasuk menggunakan Segmentasi berbasis warna, Augmentasi, Ekstraksi Fitur dan Algoritma *Support Vector Machine* (SVM).

Beberapa penelitian terkait ;


1. A. Karim, “implementasi algorotma promethee dalam melakukan analisa performa matauang virtual,” jurnal ilmu komputer dan informatika, vol. 06, p. 10, 2022.
2. T. D. R. L. Ahmad Karim, “Implementasi Algoritma Promethee Dalam Melakukan Analisa Performa,” ALGORITMA: Jurnal Ilmu Komputer dan Informatika, vol. 06, p. 10, 2022.
3. Hendra, A. & Masykur, A. (2020). “Deteksi dan Pengenalan Koin Menggunakan Metode Local Binary Pattern (LBP) dan Support Vector Machine (SVM).” Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), 7(3), 251- 258
4. Raharjo, W. & Prayitno, E. (2022). “Deteksi dan klasifikasi koin dengan metode ekstraksi ciri dan klasifikasi SVM.” Jurnal Ilmiah Teknologi Informasi Terapan, 5(1), 26-32.
5. Firdaus, I.I. dan Widyawan, W. (2021). “Deteksi Koin Menggunakan Metode Canny Edge Detection dan Region Growing.” Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), 5(1), 1155-1160.


Tujuan kami adalah mengembangkan sistem yang dapat mengenali jenis koin secara otomatis dengan akurasi tinggi dan memungkinkan jenis dan klasifikasi yang tepat untuk diidentifikasi.

FAKULTAS ILMU KOMPUTER
UBP KARAWANG

# Metode Pre-Process

UBP KARAWANG

**Pre-process** merupakan langkah awal dalam pengolahan citra digital untuk mempersiapkan citra untuk diproses lebih lanjut. Ini membutuhkan penyesuaian dan peningkatan gambar untuk meningkatkan kualitas dan menfasilitasi analisis lebih lanjut.

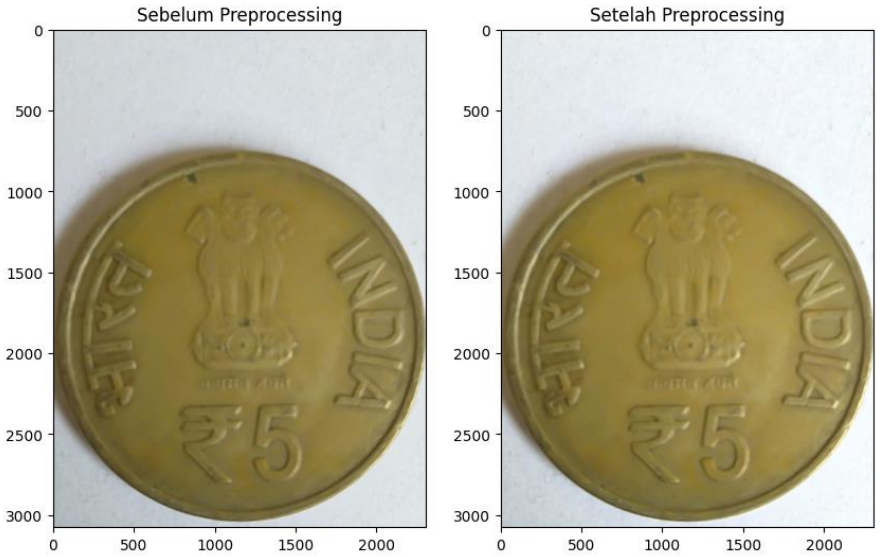
Penjelasan metode yang kami gunakan :

1. Membaca citra menggunakan OpenCV **cv2.imread**
2. Processing dengan penajaman warna **color_enhancement_faktor,**
3. Mendefinisikan fungsi untuk penajaman warna pada citra menggunakan **enhance_color**
4. Kemudian menambahkan citra hasil preprocessing ke dalam list **preprocessed_images.append**
5. Setelah melakukan Preprocessing, hasil preprocessing ditampilkan menggunakan **matplotlib.pyplot.**

# Hasil

Source Code :

```python
import numpy as np  #Library ini digunakan untuk melakukan operasi numerik dan manipulasi array multidimensi.
import pandas as pd  # Library untuk manipulasi dan analisis data
import matplotlib.pyplot as plt # Library untuk visualisasi data
from sklearn.model_selection import train_test_split   # Library untuk membagi dataset
from sklearn.metrics import accuracy_score   # Library untuk mengukur akurasi
import os # Library untuk berinteraksi dengan sistem operasi
import cv2  #Library untuk pengolahan citra
from skimage import exposure   # Library untuk ekualisasi histogram dan penyesuaian kontras

# Mendefinisikan kelas dan label
classes = {
    'emas': 0,
    'perak': 1
}

# Mendefinisikan path dataset koin
dataset_path = '/content/drive/MyDrive/Final Project Pengolahan Citra Digital/Dataset/Training'

# Mendefinisikan faktor penajaman warna
color_enhancement_factor = 1.1

# Mendefinisikan fungsi untuk penajaman warna pada citra
def enhance_color(image, color_enhancement_factor):
    enhanced_image = cv2.convertScaleAbs(image, alpha=color_enhancement_factor, beta=0)
    return enhanced_image

# Mendefinisikan list untuk menyimpan citra hasil preprocessing
preprocessed_images = []

# Loop melalui setiap file citra dalam dataset
for class_name, class_label in classes.items():
    class_path = os.path.join(dataset_path, class_name)
    for image_file in os.listdir(class_path):
        if image_file.endswith('.jpg') or image_file.endswith('.png'):
            image_path = os.path.join(class_path, image_file)
            # Membaca citra menggunakan OpenCV
            image = cv2.imread(image_path)
            # Preprocessing dengan penajaman warna
            enhanced_image = enhance_color(image, color_enhancement_factor)
            # Menambahkan citra hasil preprocessing ke dalam list
            preprocessed_images.append((image, enhanced_image, class_label))

# Menampilkan contoh citra sebelum dan sesudah preprocessing
fig, axes = plt.subplots(1, 2, figsize=(10, 10))
axes[0].imshow(cv2.cvtColor(preprocessed_images[0][0], cv2.COLOR_BGR2RGB))
axes[0].set_title('Sebelum Preprocessing')
axes[1].imshow(cv2.cvtColor(preprocessed_images[0][1], cv2.COLOR_BGR2RGB))
axes[1].set_title('Setelah Preprocessing')
plt.show()
```



# Metode Segmentasi

**Segmentasi** dalam pengolahan citra digital adalah proses membagi piksel suatu gambar menjadi beberapa kelompok atau segmen berdasarkan karakteristik atau atribut tertentu. Tujuannya adalah untuk mengidentifikasi dan membedakan objek atau area tertentu dalam gambar yang memiliki karakteristik serupa seperti warna, tekstur, intensitas, atau bentuk.

Penjelasan metode yang kami gunakan :

1. melakukan segmentasi dengan metode Canny Edge Detection dan menimpa garis tepi di atas gambar asli menggunakan **segment_canny**
2. Mengubah citra menjadi skala abu-abu
3. Menggunakan metode Canny Edge Detection dengan **threshold 50 dan 150**
4. Membuat salinan gambar asli
5. Mengganti warna garis tepi menjadi silver
6. Melakukan segmentasi pada dataset koin training
7. List untuk menyimpan gambar-gambar hasil segmentasi
8. Loop untuk setiap gambar dalam dataset
9. Melakukan segmentasi pada gambar hasil preprocessing
10. Menambahkan citra hasil preprocessing, citra hasil segmentasi, dan label kelas ke list **segmented_images**
11. Lakukan segmentasi pada dataset koin training menggunakan **segmented_images**
12. Setelah dilakukan segmentasi, hasil citra ditampilkan menggunakan **matplotlib.pyplot**

# Hasil

Source Code :

```
# Fungsi untuk melakukan segmentasi dengan metode Canny Edge Detection dan menimpa garis tepi di atas gambar asli
def segment_canny(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  # Mengubah citra menjadi skala abu-abu
    edges = cv2.Canny(gray, 50, 100)  # Menggunakan metode Canny Edge Detection dengan threshold 50 dan 150
    segmented_image = np.copy(image)  # Membuat salinan gambar asli
    segmented_image[edges != 0] = (192, 192, 192)  # Mengganti warna garis tepi menjadi hijau
    return segmented_image

# Fungsi untuk melakukan segmentasi pada dataset koin training
def segment_dataset(images):
    segmented_images = []  # List untuk menyimpan gambar-gambar hasil segmentasi
    for image in images:  # Loop untuk setiap gambar dalam dataset
        segmented_image = segment_canny(image[1])  # Melakukan segmentasi pada gambar hasil preprocessing
        segmented_images.append((image[1], segmented_image, image[2]))  # Menambahkan citra hasil preprocessing, citra hasil segmentasi, dan label kelas ke list segmented_images
    return segmented_images

# Lakukan segmentasi pada dataset koin training
segmented_images = segment_dataset(preprocessed_images)

# Tampilkan contoh gambar hasil segmentasi
fig, axes = plt.subplots(1, 2, figsize=(10, 10))
axes[0].imshow(cv2.cvtColor(segmented_images[0][0], cv2.COLOR_BGR2RGB))
axes[0].set_title('Setelah Preprocessing')
axes[0].axis('off')
axes[1].imshow(cv2.cvtColor(segmented_images[0][1], cv2.COLOR_BGR2RGB))
axes[1].set_title('Hasil Segmentasi')
axes[1].axis('off')
plt.show()
```

Setelah Preprocessing


Hasil Segmentasi

# Metode Augmentasi

**Augmentasi** dalam pengolahan citra digital adalah proses penambahan variasi atau perubahan pada data citra yang sudah ada. Tujuannya adalah untuk memperluas keragaman data, meningkatkan keragaman objek, dan memperkaya citra yang digunakan dalam pelatihan atau pengujian pengenalan citra atau mode klasifikasi.

Penjelasan metode yang kami gunakan :

1. Mendefinisikan faktor kecerahan menggunakan **brightness_factor**
2. Menghilangkan kontur pada citra hasil augmentasi menggunakan metode Canny **augmented_image_without_contour**
3. Menggunakan metode Canny untuk mendapatkan tepi menggunakan **edges = cv2.Canny**
4. Menemukan kontur pada citra hasil augmentasi menggunakan **cv2.findContours**
5. Menggambar kontur pada citra hasil augmentasi dengan latar belakang hitam **cv2.drawContours**
6. Hasil Augmentasi ditampilkan menggunakan **matplotlib.pyplot**

# Hasil

Source Code :

```python
# Mendefinisikan faktor kecerahan
brightness_factor = 1.2

# Mendefinisikan list untuk menyimpan citra hasil augmentasi
augmented_images = []

# Loop melalui setiap citra hasil segmentasi
for original_image, preprocessed_image, class_label in segmented_images:
    # Mengatur kecerahan citra hasil segmentasi
    brightened_segmented_image = cv2.convertScaleAbs(preprocessed_image, alpha=brightness_factor, beta=0)

    # Menambahkan citra hasil segmentasi dan citra augmentasi ke dalam list
    augmented_images.append((preprocessed_image, brightened_segmented_image, class_label))

# Menampilkan contoh citra hasil segmentasi dan augmentasi
fig, axes = plt.subplots(1, 2, figsize=(10, 10))
axes[0].imshow(cv2.cvtColor(augmented_images[0][0], cv2.COLOR_BGR2RGB))
axes[0].set_title('Citra Hasil Segmentasi')
axes[0].set_xticks([])
axes[0].set_yticks([])

# Menghilangkan kontur pada citra hasil augmentasi menggunakan metode Canny
augmented_image_without_contour = augmented_images[0][1].copy()
gray_augmented_image = cv2.cvtColor(augmented_image_without_contour, cv2.COLOR_BGR2GRAY)

# Menggunakan metode Canny untuk mendapatkan tepi
edges = cv2.Canny(gray_augmented_image, 50, 150)

# Menemukan kontur pada citra hasil augmentasi
contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Menggambar kontur pada citra hasil augmentasi dengan latar belakang hitam
cv2.drawContours(augmented_image_without_contour, contours, 0, (0, 0, 0), thickness=cv2.FILLED)

axes[1].imshow(cv2.cvtColor(augmented_image_without_contour, cv2.COLOR_BGR2RGB))
axes[1].set_title('Citra Hasil Augmentasi')
axes[1].set_xticks([])
axes[1].set_yticks([])
plt.show()
```

Citra Hasil Segmentasi


Citra Hasil Augmentasi

# Metode Ekstraksi Fitur

**Ekstraksi Fitur** adalah proses pemulihan dan penyajian informasi penting atau fitur penting dari gambar. Dalam ekstraksi fitur citra, tujuan utamanya adalah untuk mengidentifikasi dan mengekstraksi fitur yang dapat membedakan atau mewakili objek atau pola tertentu. Fitur tersebut berupa atribut seperti bentuk gambar, tekstur, warna atau pola spasial.

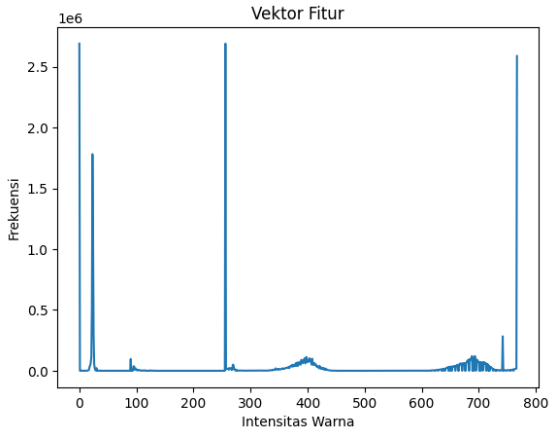
Penjelasan metode Ekstraksi Fitur Warna yang kami gunakan :

1. Ekstraksi Fitur Warna menggunakan perintah **extract_color_features**
2. Konversi gambar ke format HSV menggunakan **cv2.cvtColor**
3. Hitung histogram **hist_hue**, **hist_saturation** dan **hist_value** menggunakan **cv2.calcHist**
4. Gabungkan semua histogram menjadi satu vektor menggunakan **np.concatenate**
5. Tambahkan vektor ke dalam list menggunakan perintah **feature_vectors.append**
6. Menampilkan contoh vektor fitur menggunakan **num_examples**
7. Hasil Ekstraksi Fitur menggunakan fungsi warna HSV ditampilkan menggunakan **matplotlib.pyplot**

# Hasil

Source Code :

```python
# Fungsi untuk ekstraksi fitur menggunakan fitur warna (HSV)
def extract_color_features(image):
    # Konversi citra ke HSV
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    # Menghitung histogram hue
    hist_hue = cv2.calcHist([hsv], [0], None, [256], [0, 256])
    hist_hue = hist_hue.flatten()

    # Menghitung histogram saturation
    hist_saturation = cv2.calcHist([hsv], [1], None, [256], [0, 256])
    hist_saturation = hist_saturation.flatten()

    # Menghitung histogram value
    hist_value = cv2.calcHist([hsv], [2], None, [256], [0, 256])
    hist_value = hist_value.flatten()

    # Menggabungkan histogram menjadi satu vektor fitur
    feature_vector = np.concatenate((hist_hue, hist_saturation, hist_value))

    return feature_vector

# Mendefinisikan list untuk menyimpan vektor fitur hasil ekstraksi
feature_vectors = []

# Loop melalui setiap citra hasil augmentasi
for original_image, augmented_image, class_label in augmented_images:
    # Ekstraksi fitur menggunakan fitur warna
    color_features = extract_color_features(augmented_image)

    # Menambahkan vektor fitur ke dalam list
    feature_vectors.append(color_features)

# Menampilkan contoh vektor fitur
num_examples = 1  # Jumlah contoh yang ingin ditampilkan

for i in range(num_examples):
    # Menampilkan vektor fitur
    plt.plot(feature_vectors[i])
    plt.title('Vektor Fitur')
    plt.xlabel('Intensitas Warna')
    plt.ylabel('Frekuensi')
    plt.show()
```



# Algoritma SVM

**Algoritma *Support Vector Machine* (SVM)** adalah salah satu metode yang digunakan dalam klasifikasi dan regresi. Terutama untuk mendeteksi koin dan menjumlahkannya. Dalam konteks pendeteksian koin, algoritma SVM dapat digunakan untuk memisahkan koin dari latar belakang atau objek lain pada gambar dan untuk mengklasifikasi jenis koin yang berbeda.

Penjelasan Metode yang kami gunakan :

1. Membuat list **feature_vectors** yang digunakan untuk menyimpan vektor fitur hasil ekstraksi
2. Membuat list **X** dan **Y** yang digunakan untuk menyimpan label kelas dan fitur dari vektor fitur. Kita melakukan loop melalui setiap vektor fitur dalam **feature_vectors** dan kemudian menambahkannya ke dalam list **X** dan **Y**.
2. Mengubah list **X** dan **Y** menjadi **array numpy** menggunakan **np.array().**
3. Menggunakan **train_test_split()** dari modul **sklearn.model_selection**, data dibagi menjadi data latihan dan data uji.
4. Model Support Vector Machine (SVM) dilatih dan dibuat dengan menggunakan kelas SVC yang ditemukan dalam modul **sklearn.svm.**
5. Untuk menghitung skor akurasi, metode **score()** digunakan pada model, dengan memasukkan data latih dan data uji sebagai argumen.
6. gambar sebanyak 16 buah, akan ditampilkan menggunakan **matplotlib.pyplot.**

# Hasil

Source Code :

```
# Mendefinisikan list untuk menyimpan vektor fitur hasil ekstraksi
feature_vectors = []

# Loop melalui setiap citra hasil augmentasi
for original_image, augmented_image, class_label in augmented_images:
    # Ekstraksi fitur menggunakan fitur warna
    color_features = extract_color_features(original_image)

    # Menghilangkan kontur pada citra
    no_contour_image = augmented_image.copy()
    gray = cv2.cvtColor(no_contour_image, cv2.COLOR_BGR2GRAY)
    _, threshold = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV)
    contours, _ = cv2.findContours(threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cv2.drawContours(no_contour_image, contours, -1, (0, 0, 0), 3)
    no_contour_image = cv2.cvtColor(no_contour_image, cv2.COLOR_BGR2RGB)

    # Menambahkan vektor fitur ke dalam list
    feature_vectors.append((no_contour_image, color_features, class_label))

# Mendefinisikan list untuk menyimpan label dan fitur
X = []
Y = []

# Loop melalui setiap vektor fitur
for _, feature_vector, class_label in feature_vectors:
    X.append(feature_vector)
    Y.append(class_label)

# Konversi menjadi array numpy
X = np.array(X)
Y = np.array(Y)

# Membagi data menjadi data latih dan data uji
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=10)

# Normalisasi data
x_train = x_train / 255.0
x_test = x_test / 255.0

# Melatih model SVM
svm_model = SVC()
svm_model.fit(x_train, y_train)

# Mengevaluasi model
train_score = svm_model.score(x_train, y_train)
test_score = svm_model.score(x_test, y_test)

print("Training Score:", train_score)
print("Testing Score:", test_score)

# Menampilkan hasil gambar klasifikasi dan menghitung jumlah koin emas
fig, axes = plt.subplots(2, 4, figsize=(10, 10))

for i, (no_contour_image, _, class_label) in enumerate(feature_vectors[:8]):
    ax = axes[i // 4, i % 4]

    ax.imshow(no_contour_image)

    if class_label == 0:
        ax.set_title('Koin Emas')
    elif class_label == 1:
        ax.set_title('Koin Perak')

    ax.axis('off')

plt.tight_layout()
plt.show()
```

# Kesimpulan

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan bahwa algoritma SVM adalah pilihan yang baik untuk klasifikasi koin berdasarkan fiturnya. SVM dapat memberikan hasil klasifikasi yang akurat dan dapat diandalkan melalui pemrosesan data yang tepat dan penyesuaian yang optimal. Selain itu, penulis membuat program yang pada dasarnya sama dengan program lain yang menggunakan algoritma Support Vector Machine, tetapi mereka menambahkan fitur sum pada nilai mata uang koin yang diklasifikasikan. Diharapkan bahwa pengguna program akan mudah mengklasifikasi jenis dan nilai mata uang koin dengan program ini.

FAKULTAS ILMU KOMPUTER
UBP KARAWANG

UBP KARAWANG

# TERIMA KASIH