|  |  |  |
| --- | --- | --- |
|  | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **Detekce barev kamerou** | | |
| Šimon Jantošík | | |
|  | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2024/2025 | |

#### Poděkování

* Dekuji panu učiteli Mgr. Markovi Lučnému a Ing. Petru Grussmanovi za rady při vytváření tohoto projektu.

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité   
informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové   
a umělecké škole v Opavě, Praskova 399/8.

V Opavě 31. 12. 2016

*podpis autora práce*

**ABSTRAKT**

Tento projekt se zabývá implementací systému pro rozpoznávání barev v reálném čase pomocí webové kamery. Jsou využita dvě technická řešení: první pomocí JavaScriptu, HTML5 Canvas API a CSS, druhé za využití knihovny OpenCV v Pythonu. Cílem práce je demonstrovat různé přístupy k detekci barev a jejich praktické využití, včetně identifikace barevných objektů, jejich lokalizace v obraze a prezentace výsledků uživatelsky přívětivým způsobem.

V rámci řešení s Canvas API je implementována analýza obrazu přímo v prohlížeči, kde uživatel může interaktivně vybírat barvy kliknutím na obraz. CSS stylování zajišťuje přehledné a atraktivní uživatelské rozhraní. Druhé řešení využívá Python a OpenCV pro pokročilé zpracování obrazu, včetně rozpoznání objektů na základě barevného rozsahu, výpočtu jejich rozměrů v reálných jednotkách a jejich označení v obraze.

Práce ilustruje přínosy a omezení obou technologií při zpracování obrazu v reálném čase. Výsledný systém má potenciální využití v robotice, průmyslové automatizaci nebo interaktivních aplikacích.

Klíčová slova: rozpoznávání barev, webová kamera, HSV model, OpenCV, Canvas API, CSS, detekce objektů, Python, JavaScript, zpracování obrazu, reálný čas.

OBSAH

**Anotace**. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3

**Úvod** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .5

**1 Teoretická a metodická východiska** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .6  
 1.1 Zpracování obrazu na počítači . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6  
 1.2 Barevný model HSV . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6  
 1.3 Metodologie . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6  
 1.4 Výběr použitých metod . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7

**2 Využité technologie** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8  
 2.1 HTML5 Canvas API, JavaScript a CSS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8  
 2.2 OpenCV a Python . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9  
 2.3 Použité zařízení . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .10

**3 Způsoby řešení a použité postupy** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .11  
 3.1 Celková architektura . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11  
 3.2 Webová aplikace s Canvas API, JavaScript a CSS . . . . . . . . . . . . . . . . . . . . . . 12  
 3.3 Desktopová aplikace s OpenCV a Python . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12  
 3.4 Testování a ověřování funkčnosti . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .13  
 3.5 Schémata a diagramy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

**4 Implementace** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .15  
 4.1 Backend implementace . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .15

**5 Výsledky řešení, výstupy, uživatelský manuál** . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17  
 5.1 Splněné cíle . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .17  
 5.2 Nesplněné cíle . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .17  
 5.3 Uživatelské rozhraní webové aplikace . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .17  
 5.4 Výstupy desktopové aplikace . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .18  
 5.5 Uživatelský manuál . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18  
 5.6 Zhodnocení práce . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .19

**Závěr** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .20  
**Seznam použitých informačních zdrojů** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

Úvod

Rozpoznávání barev hraje klíčovou roli v řadě aplikací, od robotiky přes průmyslovou automatizaci až po interaktivní systémy. Tento projekt se zaměřuje na návrh a implementaci dvou odlišných řešení detekce barev v reálném čase s využitím běžné webové kamery. První řešení využívá webové technologie jako JavaScript, HTML5 Canvas API a CSS pro zpracování a vizualizaci obrazu přímo v prohlížeči. Druhé řešení implementuje rozpoznávání barev pomocí Pythonu a knihovny OpenCV, která poskytuje pokročilé nástroje pro analýzu obrazu.

Hlavní motivací pro zpracování tohoto projektu je rostoucí význam barevného vidění ve světě technologií. Schopnost automaticky identifikovat barvy a objekty na základě barevných kritérií má široké uplatnění v řadě oborů, včetně zlepšení lidsko-strojového rozhraní, optimalizace výrobních procesů nebo vývoje chytrých zařízení. Přidáním CSS stylů se zvyšuje uživatelská přívětivost webového řešení, což přispívá k celkové atraktivitě a snadnému použití. Propojení dvou technicky rozdílných přístupů umožňuje srovnání jejich výhod, nevýhod a praktické použitelnosti.

Cílem práce je:

* Implementovat dvě funkční aplikace pro rozpoznávání barev s využitím rozdílných technologií.
* Demonstrovat rozdíly v přístupu ke zpracování obrazu v prohlížeči a v desktopovém prostředí.
* Porovnat efektivitu a možnosti těchto technologií při řešení úloh detekce barev.
* Poskytnout čtenáři komplexní přehled o problematice a návod na použití obou řešení v praxi.

Práce je rozdělena do několika kapitol. Nejprve se zaměříme na teoretický základ zpracování obrazu a popis použitých technologií, včetně principů barevného modelu HSV, CSS stylů a algoritmů pro detekci objektů. Následuje podrobný popis implementace obou řešení a jejich vzájemné srovnání. V závěru práce budou shrnuty dosažené výsledky, výhody a omezení obou přístupů a možné oblasti budoucího rozvoje.

# teoretická část

**1.1 Zpracování obrazu na počítači**

Zpracování obrazu na počítači je disciplína zaměřená na analýzu a manipulaci obrazových dat. Využívá algoritmy digitálního zpracování signálů, které pracují s obrazovými daty ve formě matic obsahujících barevné informace. Tento projekt se zaměřuje na detekci barev a rozpoznávání objektů pomocí barevného modelu HSV, který umožňuje přesnější analýzu barevných odstínů než tradiční RGB model. Pro zlepšení uživatelské přívětivosti webového řešení byly přidány CSS styly, které zdůrazňují klíčové vizuální prvky a zajišťují atraktivní vzhled aplikace.

### 1.2 Barevný model HSV

HSV model (Hue, Saturation, Value) rozděluje barevný prostor do tří složek:

* **Hue (odstín):** Určuje základní barvu v rozmezí 0–360°.
* **Saturation (sytost):** Vyjadřuje intenzitu barvy.
* **Value (jas):** Udává světlost barvy.

Výhodou HSV oproti RGB je schopnost lépe reprezentovat lidské vnímání barev, což usnadňuje detekci specifických barev a jejich rozsahů.

## 1.3 Metodologie

### Formulace hypotéz

1. Použití barevného modelu HSV zlepší přesnost detekce barev ve srovnání s RGB.
2. Rozpoznávání barev na základě uživatelem zadaných rozsahů umožní spolehlivou lokalizaci objektů na základě jejich barev.
3. Webová i desktopová implementace bude schopna fungovat v reálném čase při zachování dostatečné přesnosti a výkonu.

### Použité metody

1. **Analýza obrazu:**
   * Převod obrazu z RGB do HSV modelu.
   * Vytvoření binární masky pro daný rozsah barev pomocí prahování.
2. **Detekce objektů:**
   * Identifikace oblastí odpovídajících dané barvě pomocí algoritmů pro hledání kontur.
   * Ověření minimální velikosti objektů pro eliminaci šumu.
3. **Styling aplikace:**
   * CSS zajišťuje vizuální konzistenci a lepší čitelnost výsledků, např. zvýraznění detekovaných oblastí a interaktivní vizuální prvky.
4. **Testování:**
   * Webová aplikace byla testována na různých prohlížečích (Chrome, Firefox) a za různých světelných podmínek.
   * Desktopová aplikace byla testována na různých scénách s cílem porovnat přesnost a výkon.

## Výběr použitých metod

### Proč HSV model?

HSV model byl zvolen pro svou schopnost oddělit barevné odstíny (Hue) od sytosti a jasu. Tato vlastnost umožňuje snadno nastavit barevné rozsahy nezávisle na změnách světelných podmínek, což je klíčové pro spolehlivou detekci barev.

### Proč dvě řešení?

Dvě rozdílná technologická řešení poskytují možnost srovnání přístupu ke zpracování obrazu:

* **Webová aplikace:** Lehká a dostupná bez instalace speciálního software. Je ideální pro rychlé nasazení a jednodušší úlohy.
* **Desktopová aplikace:** Nabízí vyšší přesnost a flexibilitu díky využití knihovny OpenCV. Je vhodná pro komplexnější úlohy a pokročilé analýzy.

# Využité technologie

**2.1 HTML5 Canvas API, JavaScript a CSS**

Jedno z řešení projektu bylo implementováno pomocí webových technologií HTML5 Canvas API, JavaScriptu a CSS.

**Klíčové vlastnosti HTML5 Canvas API:**

* Poskytuje nástroje pro kreslení a manipulaci s 2D grafikou přímo v prohlížeči.
* Umožňuje práci s jednotlivými pixely obrazu, což je klíčové pro analýzu barev.
* Snadná integrace s dalšími webovými technologiemi, včetně JavaScriptu a CSS.
* Vysoká rychlost vykreslování díky hardwarové akceleraci.

**Klíčové vlastnosti JavaScriptu:**

* Dynamický a flexibilní jazyk vhodný pro vývoj klientských aplikací.
* Podpora asynchronního programování pomocí Promises a async/await, což umožňuje plynulé zpracování obrazu v reálném čase.
* Široká podpora v moderních prohlížečích.
* Snadná integrace se systémy HTML a CSS pro tvorbu interaktivního uživatelského rozhraní.
* Tato kombinace umožňuje zpracování obrazu přímo na straně klienta, což minimalizuje závislost na externích knihovnách.

**Klíčové vlastnosti CSS:**

* Umožňuje stylizovat webovou aplikaci a zlepšuje uživatelskou zkušenost.
* Přidává vizuální efekty, jako je zaoblení plátna, stíny nebo animace při najetí kurzorem.
* Zajišťuje responzivní design, který přizpůsobuje rozvržení obsahu různým zařízením.

Ukázka CSS stylu pro plátno:

canvas {

border: 3px solid #1b5e20;

border-radius: 8px;

background-color: #ffffff;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

transition: transform 0.2s ease-in-out;

}

canvas:hover {

transform: scale(1.05);

box-shadow: 0 6px 12px rgba(0, 0, 0, 0.3);

}

## OpenCV a Python

Druhé řešení bylo realizováno pomocí knihovny OpenCV a jazyka Python, který je oblíbený pro své široké možnosti v oblasti zpracování obrazu.

**Klíčové vlastnosti OpenCV:**

* Podpora práce s různými barevnými modely, včetně HSV, RGB a BGR.
* Pokročilé algoritmy pro detekci a analýzu objektů v obraze.
* Možnost snadné integrace s knihovnami pro strojové učení, jako je TensorFlow nebo PyTorch.
* Optimalizace pro výkon, včetně podpory GPU akcelerace.

**Klíčové vlastnosti Pythonu:**

* Jednoduchá syntaxe, která umožňuje rychlý vývoj aplikací.
* Bohatá sada knihoven a nástrojů pro vědu o datech a počítačové vidění.
* Silná komunita a rozsáhlá dokumentace.
* Podpora interaktivního vývoje a testování.

Toto řešení poskytuje robustnější možnosti zpracování obrazu a je vhodné pro aplikace vyžadující vyšší míru přizpůsobení a výkonu.

## Použité zařízení

Pro záznam obrazu byla použita klasická webkamera Connect IT, která nabízí dostatečné rozlišení pro zpracování obrazu v reálném čase.

**Klíčové vlastnosti webkamery:**

* Rozlišení 640 × 480 pixelů, vhodné pro analýzu obrazu bez výrazného zatížení systému.
* Podpora standardního rozhraní USB, zajišťující snadné připojení k různým systémům.
* Stabilní snímková frekvence pro práci v reálném čase.

Toto zařízení bylo zvoleno pro svou dostupnost, jednoduchou integraci a dostatečné parametry pro potřeby projektu.

# Způsoby řešení a použité postupy

## 3.1 Celková architektura

Projekt je rozdělen do dvou částí: webová aplikace využívající HTML5 Canvas API, JavaScript a CSS, a desktopové řešení s využitím OpenCV a Pythonu. Obě implementace mají za cíl demonstrovat různé přístupy k rozpoznávání barev a jejich efektivitu při řešení úloh zpracování obrazu v reálném čase.

### Hlavní principy architektury:

* **Modularita:** Oddělení jednotlivých částí aplikace pro snadnější údržbu a rozšíření.
* **Efektivita:** Minimalizace náročnosti na hardware při zachování plynulého zpracování obrazu.
* **Interaktivita:** Možnost uživatelsky vybírat barvy a zobrazovat výsledky analýzy v reálném čase.

## 3.2 Webová aplikace s Canvas API, JavaScript a CSS

Webová aplikace využívá technologie dostupné přímo v moderních prohlížečích. HTML5 Canvas API je klíčový nástroj pro manipulaci s pixely v reálném čase, zatímco CSS zajišťuje vizuální přehlednost a atraktivitu.

### Implementační kroky:

1. **Získání videostreamu:**  
   Pomocí rozhraní navigator.mediaDevices.getUserMedia() je přistupováno k webkameře, jejíž obraz je následně zobrazen v prvku <video>.

async function startVideo() {

try {

const stream = await navigator.mediaDevices.getUserMedia({

video: { width: 640, height: 480 }

});

video.srcObject = stream;

video.play();

} catch (err) {

console.error("Error accessing the webcam: ", err);

}

}

1. **Zpracování obrazu:**  
   Obraz z kamery je snímán a vykreslován na plátno (<canvas>), kde jsou data pixelů analyzována. Barvy jsou převáděny do HSV modelu, což umožňuje přesnější detekci barev než RGB model.

function rgbToHsv(r, g, b) {

r /= 255;

g /= 255;

b /= 255;

let max = Math.max(r, g, b);

let min = Math.min(r, g, b);

let h, s, v = max;

let d = max - min;

s = max === 0 ? 0 : d / max;

if (max === min) {

h = 0;

} else {

switch (max) {

case r: h = (g - b) / d + (g < b ? 6 : 0); break;

case g: h = (b - r) / d + 2; break;

case b: h = (r - g) / d + 4; break;

}

h /= 6;

}

return { h: h \* 180, s: s \* 255, v: v \* 255 };

}

1. **Interakce s uživatelem:**  
   Uživatel kliknutím na obraz vybírá barvu, kterou chce detekovat. Po výběru je rozsah barev uložen a použity algoritmy pro zvýraznění odpovídajících oblastí.
2. **Vizualizace výsledků:**  
   Detekované oblasti jsou vykresleny na výstupním plátně ve formě černobílého maskovaného obrazu a ohraničení.

### Parametry a vlastnosti:

* **Rozlišení:** 640 × 480 pixelů.
* **Rychlost zpracování:** Přizpůsobeno snímkovací frekvenci kamery (~30 FPS).
* **Tolerance barev:** Nastavitelná hodnota pro zvýšení flexibility.

## 3.3 Desktopová aplikace s OpenCV a Python

Desktopová aplikace je implementována v Pythonu s využitím knihovny OpenCV, která nabízí pokročilé nástroje pro práci s obrazem.

### Implementační kroky:

1. **Získání obrazu:**  
   OpenCV využívá knihovnu cv2.VideoCapture pro přístup ke kameře. Obraz je okamžitě převeden do HSV modelu pomocí funkce cv2.cvtColor.
2. **Výběr barvy:**  
   Kliknutím na obraz uživatel vybere barvu, přičemž se určí rozsah HSV hodnot pro maskování. Speciální pozornost je věnována odstínům červené, které vyžadují rozdělení na dvě části kvůli cyklické povaze HSV modelu.
3. **Maskování a detekce objektů:**  
   Oblasti odpovídající zadané barvě jsou extrahovány pomocí funkce cv2.inRange. Kontury jsou následně detekovány pomocí cv2.findContours.
4. **Měření objektů:**  
   V případě známého rozměru referenčního objektu lze vypočítat reálné rozměry detekovaných objektů. K tomu slouží kalibrační poměr, který uživatel zadává při označení referenčního objektu.
5. **Zobrazení výsledků:**  
   Výsledky jsou vykresleny přímo na obraz s popisky rozměrů a názvem rozpoznané barvy.

### Parametry a vlastnosti:

* **Tolerance barev:** Nastavitelná pro každý kanál HSV (odstín, sytost, jas).
* **Rozměry objektů:** Přesnost závisí na kvalitě kamery a kalibrace.

## 3.4 Testování a ověřování funkčnosti

Pro testování byla použita klasická webkamera při různých světelných podmínkách a s různými typy objektů.

### Postup testování:

1. **Jednoduché objekty:** Testováno na základních barvách (červená, zelená, modrá).
2. **Komplexní scény:** Obsahující více objektů a barev.
3. **Různé osvětlení:** Ověření funkčnosti při změnách intenzity a barvy světla.

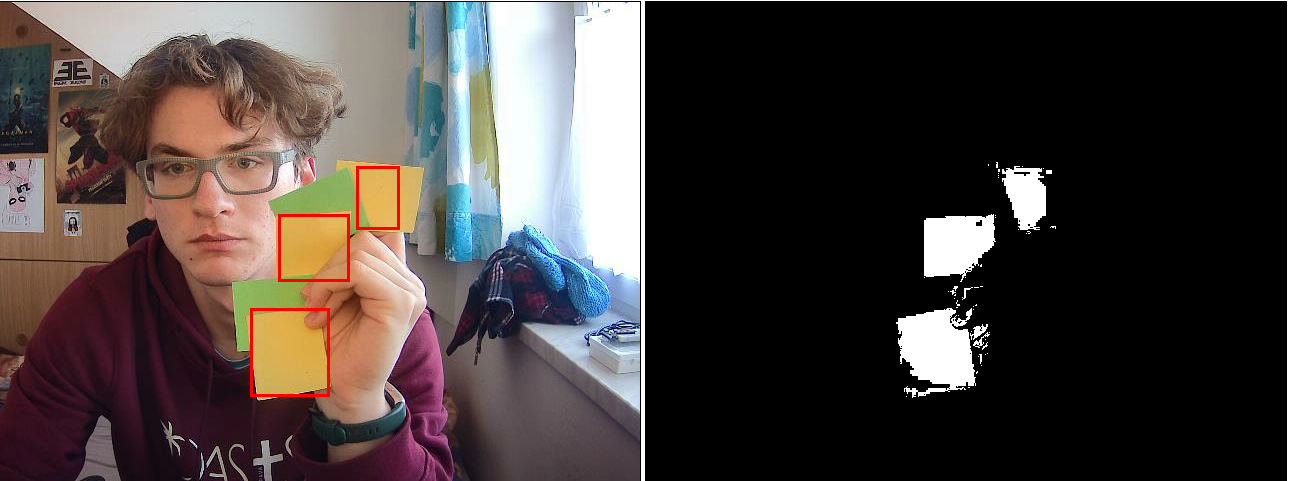
### Výsledky:

* Webová aplikace vykazuje dobrý výkon při menší zátěži a jednodušších scénách.
* Desktopové řešení poskytuje přesnější výsledky a lepší robustnost vůči změnám prostředí.

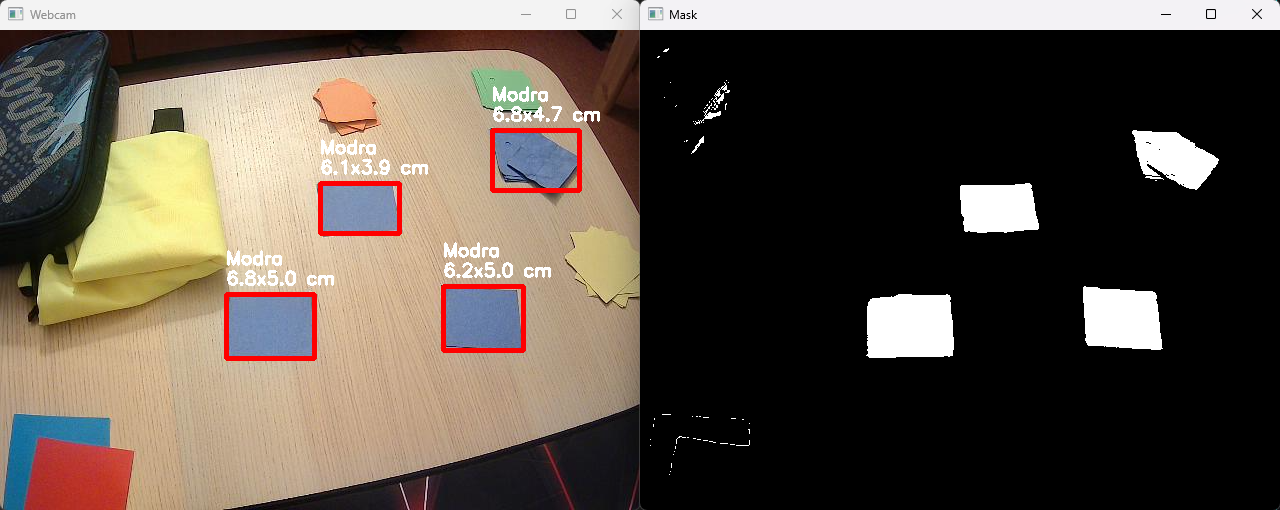
## 3.5 Schémata a diagramy

Následující obrázky ilustrují architekturu a průběh zpracování obrazu v obou řešeních:

Webová aplikace s využitím Canvas API:



Desktopová aplikace s využitím OpenCV:



# 4 IMPLEMENTACE

### 4.1 Backend implementace

### Zpracování obrazu v desktopové aplikaci

Desktopová aplikace byla navržena pro pokročilejší zpracování obrazu s využitím knihovny OpenCV. Implementace zahrnuje následující kroky:

1. **Přístup k videostreamu:**  
   OpenCV využívá cv2.VideoCapture pro získání obrazu z webkamery.

video = cv2.VideoCapture(0)

1. **Převod do HSV modelu:**  
   Obraz je okamžitě převeden do barevného modelu HSV pomocí funkce cv2.cvtColor.

hsv\_image = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

1. **Maskování barev:**  
   Specifický rozsah barev je detekován pomocí cv2.inRange.

try:

if isinstance(lower, tuple):

mask1 = cv2.inRange(hsv\_image, lower[0], upper[0])

mask2 = cv2.inRange(hsv\_image, lower[1], upper[1])

mask = cv2.bitwise\_or(mask1, mask2)

else:

mask = cv2.inRange(hsv\_image, lower, upper)

1. **Detekce a označení objektů:**  
   Kontury detekovaných objektů jsou nalezeny funkcí cv2.findContours a vykresleny na obraz.

**contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)**

**for contour in contours:**

**if cv2.contourArea(contour) > 500:**

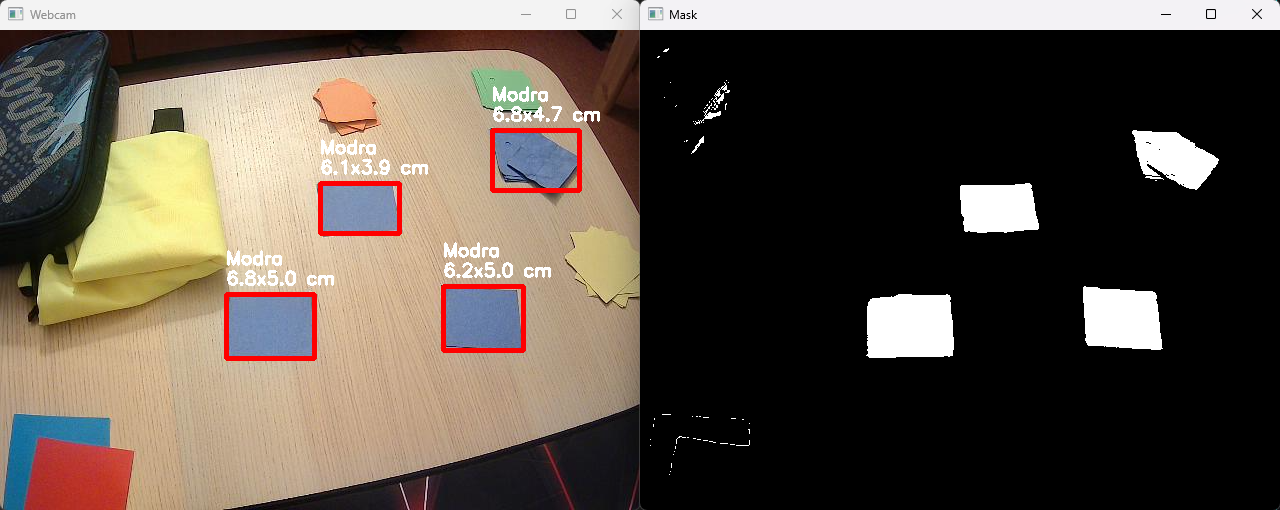
**rect = cv2.minAreaRect(contour)**

**box = cv2.boxPoints(rect)**

**box = box.astype(np.int32)**

**cv2.drawContours(img, [box], 0, (0, 0, 255), 3)**

1. **Měření rozměrů objektů:**  
   Pokud je znám referenční objekt, lze vypočítat reálné rozměry detekovaných objektů.



Ukázka výsledného obrazu s označenými objekty:  
(Obrázek 4.2: Výstup desktopové aplikace s detekovanými objekty)

### Testování funkcionality

Pro testování byly použity různé scénáře s důrazem na spolehlivost a přesnost obou implementací:

1. **Jednoduché scény:** Detekce základních barev (červená, zelená, modrá) na homogenním pozadí.
2. **Komplexní scény:** Detekce barev v prostředí s více objekty a různými odstíny.
3. **Změny osvětlení:** Testování při různých intenzitách a typech osvětlení.

Výsledky:

* Webová aplikace poskytuje rychlou zpětnou vazbu s omezenou přesností.
* Desktopové řešení nabízí vyšší přesnost a stabilitu i za náročných podmínek.

(Obrázek 4.3: Porovnání výsledků webové a desktopové aplikace)

# VÝSLEDKY ŘEŠENÍ A ZHODNOCENÍ

**5.1 Splněné cíle**

Projekt splnil následující cíle:

1. **Implementace dvou funkčních řešení:**
   * **Webová aplikace:** Umožňuje uživateli detekovat barvy a vizualizovat výsledky přímo v prohlížeči pomocí HTML5 Canvas API a JavaScriptu.
   * **Desktopová aplikace:** Nabízí pokročilé zpracování obrazu pomocí Pythonu a knihovny OpenCV.
2. **Porovnání přístupů:**
   * Webová aplikace poskytuje rychlé a snadno použitelné řešení dostupné v jakémkoliv moderním prohlížeči.
   * Desktopová aplikace poskytuje vyšší přesnost a flexibilitu při práci s komplexními scénami.
3. **Testování a ověření funkčnosti:**
   * Obě řešení byla testována v různých podmínkách a prokázala schopnost detekovat barvy v reálném čase.
4. **Výsledné parametry:**
   * Webová aplikace: Rychlost zpracování přibližně 25 FPS, rozlišení 640 × 480 pixelů.
   * Desktopová aplikace: Přesnost měření objektů v závislosti na kvalitě kalibrace, rozlišení stejné jako u webové aplikace.

**5.2 Nesplněné cíle**

Projekt čelil několika omezením:

* **Závislost na kvalitě kamery:** Přesnost výsledků je ovlivněna rozlišením a světelnými podmínkami.
* **Omezená kalibrace ve webové aplikaci:** Možnost měření rozměrů objektů nebyla v Canvas API implementována.

**5.3 Uživatelské rozhraní webové aplikace**

Webová aplikace byla navržena tak, aby byla uživatelsky přívětivá a snadno ovladatelná.

**Klíčové prvky:**

* **Interaktivní výběr barvy:** Uživatel může kliknutím na obraz vybrat barvu, kterou chce detekovat.
* **Zobrazení výsledků:** Detekovaná barva je vizualizována na černobílém plátně, kde odpovídající oblasti jsou zvýrazněny.

**Obrázek 4.1:** Ukázka webového rozhraní s detekovanou červenou barvou.

**5.4 Výstupy desktopové aplikace**

Desktopová aplikace poskytuje detailnější analýzu obrazu.

**Klíčové funkce:**

* **Maskování a detekce objektů:** Vytváření binární masky pro daný barevný rozsah.
* **Měření rozměrů:** Na základě kalibrace lze určit rozměry detekovaných objektů v reálných jednotkách.
* **Popisky:** Každý detekovaný objekt je označen popisem barvy a rozměrů.

**Obrázek 4.2:** Výstup desktopové aplikace s detekovanými objekty a jejich rozměry.

**5.5 Uživatelský manuál**

**Webová aplikace:**

1. **Přístup:** Otevřete main.html v libovolném moderním prohlížeči.
2. **Použití:**
   * Spusťte kameru a klikněte na oblast obrazu, kterou chcete analyzovat.
   * Výsledky se zobrazí na černobílém výstupním plátně.

**Desktopová aplikace:**

1. **Instalace:**
   * Nainstalujte Python a knihovnu OpenCV pomocí příkazu pip install opencv-python.
2. **Použití:**
   * Spusťte color\_recognition.py.
   * Klikněte na požadovanou oblast pro výběr barvy.
   * Výsledky se zobrazí v reálném čase v okně aplikace.

**5.6 Zhodnocení práce**

Projekt úspěšně demonstroval dva různé přístupy k detekci barev, které jsou vhodné pro různé typy aplikací. Webová aplikace se ukázala jako rychlé a jednoduché řešení pro méně náročné úlohy. Desktopové řešení nabízí větší flexibilitu a přesnost, což je výhodné pro aplikace s vyššími požadavky na zpracování obrazu.

Obě řešení poskytují stabilní výkon a mohou být dále rozšířena o pokročilé funkce, jako je rozpoznávání tvarů, pokročilá analýza scén nebo integrace s externími systémy.

# Závěr

Tento projekt si kladl za cíl implementovat dvě odlišná řešení pro rozpoznávání barev v reálném čase s využitím webové kamery. Webová aplikace založená na HTML5 Canvas API a JavaScriptu představuje rychlé a snadno dostupné řešení pro uživatele, kteří potřebují analyzovat obraz přímo v prohlížeči. Desktopová aplikace postavená na Pythonu a knihovně OpenCV nabízí pokročilé možnosti zpracování obrazu a přesnější detekci barev a objektů.

V průběhu práce byly dosaženy všechny hlavní cíle. Obě aplikace úspěšně prokázaly svou funkčnost, a to jak při testování základních scén, tak při složitějších úlohách s více objekty a proměnlivým osvětlením. Díky těmto řešením lze ilustrovat výhody a nevýhody obou přístupů, což umožňuje informovaný výběr technologie podle specifických potřeb.

Praktické využití projektu lze nalézt v různých oblastech, jako je robotika, průmyslová automatizace, kontrola kvality či interaktivní aplikace. Webové řešení je ideální pro rychlé nasazení v prostředí s omezenými nároky na výkon. Desktopová aplikace nachází uplatnění tam, kde je vyžadována vyšší přesnost a flexibilita.

Do budoucna by projekt mohl být rozšířen o následující funkce:

* **Integrace strojového učení:** Pro rozpoznávání složitějších vzorů a tvarů.
* **Podpora pokročilé kalibrace:** Umožnění přesnějšího měření objektů i v nepříznivých podmínkách.
* **Mobilní aplikace:** Rozšíření přístupnosti na mobilní zařízení.

Projekt tak představuje nejen funkční řešení, ale i základ pro další vývoj v oblasti zpracování obrazu.

Seznam použitýCH INFORMAČNÍCH ZDROJů

[1] OpenCV Team. *OpenCV Documentation* [Online]. 2024 [cit. 2024-12-29]. Dostupné   
z: https://docs.opencv.org

[2] Mozilla Developer Network. *Canvas API Documentation* [Online]. 2024 [cit. 2024-12-29]. Dostupné   
z: <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API>

[3] Python Software Foundation. *Python 3 Documentation* [Online]. 2024 [cit. 2024-12-29]. Dostupné   
z: <https://docs.python.org>

[4] GitHub Contributors. *Sample Color Detection Using OpenCV* [Online]. 2024 [cit. 2024-12-29]. Dostupné z: <https://github.com>

[5] Stack Overflow Community. *HSV Color Model Questions and Answers* [Online]. 2024 [cit. 2024-12-29]. Dostupné z: <https://stackoverflow.com>