VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Detekce kouře v obraze

Vision Based Smoke Detection

2018 Jan Osmančík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jan Osmančík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Detekce kouře v obrazech
Vision Based Smoke Detection

Jazyk vypracování: čeština

Zásady pro vypracování:

Detekce kouře je důležitým bezpečnostním prvkem v místech s vyšší koncentrací osob, jako jsou školy či nákupní centra. Tato místa jsou vybavena speciálními požárními hlásiči, stejně tak ale mnoha bezpečnostními kamerami, které však k účelu detekce kouře využity nejsou, ačkoliv by jako doplňkový detektor mohly sloužit. Cílem bakalářské práce je vytvořit software, který v obrazech detekuje oblasti, ve kterých se vyskytuje kouř. Aplikace bude implementována v jazyce C++ s využitím knihovny OpenCV.

Ve své práci proveďte:
1. Popište zadaný problém.
2. Analyzujte řešení a popište potřebnou teorii.
3. Vytvořte program pro detekci kouře v obrazech a proveďte jeho testování.
4. Zhodnoťte dosažené výsledky.

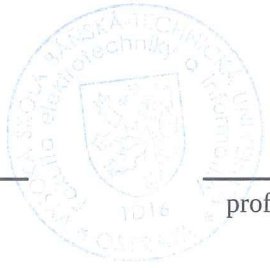
Seznam doporučené odborné literatury:

[1] Simone Calderara, Paolo Piccinini, and Rita Cucchiara: *Vision based smoke detection system using image energy and color information*. Mach. Vision Appl. 22, 4 (July 2011), pp. 705-719, 2011
[2] Hongda Tian, Wanqing Li, Lei Wang, and Philip Ogunbona: *Smoke Detection in Video: An Image Separation Approach*. Int. J. Comput. Vision 106, 2 (January 2014), pp. 192-209, 2014

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michael Holuša**

Datum zadání: 01.09.2017
Datum odevzdání: 30.04.2018



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2018 .....

Rád bych na tomto místě poděkoval vedoucímu této práce Ing. Michaelovi Holušovi za příkladné vedení a cenné rady v průběhu tvorby této práce, protože by tato práce bez jeho pomoci nevznikla.

Abstrakt

Cílem této bakalářské práce je analýza obrazu ze statické kamery a následné zpracování a nalezení oblastí, které obsahují kouř. Prvním krokem je extrakce pozadí pomocí modelu Gaussových směsí (Mixture of Gaussians). Toto pozadí je průběžně aktualizováno, abychom se vyhnuli případným změnám ve scéně např. změna světla nebo objektů. Rozdíl aktuálního snímku a pozadí s aplikováným prahováním jsou oblasti, které se ve videosekvenci mění. Detekce kouře z těchto oblastí provádíme porovnáváním jednotlivých barevných složek a kontrolou pohybu a změny velikostí kontur oblastí, ve kterých by se mohl potencionálně vyskytovat kouř.

Klíčová slova: kouř, detekce, model Gaussových směsí, MOG, barevný model, HSV, HSL, RGB, YUV

Abstract

The aim of this bachelor thesis is to analyze the image from a static camera and to subsequently process and to find areas containing smoke. The first step is background extraction using the Mixture of Gaussians. The background is updated many times during the application is running. The difference between the current frame and the applied threshold is the areas that are changing in the movie. Smoke detection from these areas is done by comparing each color component and controlling motion and changing the contour sizes of the areas where smoke might be present.

Key Words: smoke, detection, Mixture of Gaussians, MOG, color model, HSV, HSL, RGB, YUV

Obsah

Seznam použitých zkratek a symbolů **7**

Seznam obrázků **8**

Seznam tabulek **9**

Seznam výpisů zdrojového kódu **10**

1 Úvod **11**

2 Algoritmy a techniky použité k detekci **12**
2.1 Digitální obraz . . . 12
2.2 Barevné modely a prostory . . . 13
2.3 Gaussovy směsi (Mixture of Gaussians) . . . 17
2.4 Gaussovo rozostření (odstranění šumu) . . . 18
2.5 Metoda odstranění šumu pomocí Gaussovy funkce . . . 19
2.6 Nalezení kontur (obrysů) v obraze . . . 20

3 Analýza řešení problému **21**
3.1 Knihovna OpenCV . . . 21
3.2 Extrakce pozadí a vytvoření masky popředí . . . 21
3.3 Ověřování barvy . . . 25
3.4 Ověřování pohybu . . . 26

4 Testování **30**
4.1 Video 1 - motokára . . . 30
4.2 Video 2 - diskotékový kouř . . . 31
4.3 Video 3 - požár odpadkového koše v truhlářské dílně . . . 32
4.4 Video 4 - bílá dýmovnice venku . . . 33
4.5 Video 5 - simulovaný kouř v laboratoři . . . 34
4.6 Video 6 - nehoda v tunelu . . . 35
4.7 Shrnutí dosažených výsledků . . . 36

5 Závěr **37**
5.1 Zhodnocení přínosu a návrhy k rozšíření . . . 37

Literatura **39**

Seznam použitých zkratek a symbolů

HSV	–	Barevný model, složený ze složek tón, sytost, jas (Hue, Saturation, Value)
HSL	–	Barevný model, složený ze složek tón, sytost, jas (Hue, Saturation, Lightness)
RGB	–	Barevný model, složený ze složek červená, zelená a modrá (Red, Green, Blue)
YUV	–	Barevný model, složený ze složek jasu a dvou chomatických barevných kanálů

Seznam obrázků

1 Rastrový obrázek o velikosti 16x16 pixelů . . . 13
2 RGB barevný model [17] . . . 14
3 Barevný válec HSV modelu [20] . . . 15
4 HSL mapované do koule s výřezem [19] . . . 15
5 Příklad U-V barevné plochy s hodnotou Y' 0,5 uvnitř RGB gamutu [18] . . . 17
6 Gaussův složený model 1D . . . 18
7 Ukázka konvoluční matice Gaussovy funkce pro okolí 10 pixelů a směrodatnou odchylku 1,8. [16] . . . 19
8 Ukázka redukce šumu pomocí Gaussova rozostření. Vlevo originální obrázky, vpravo po průchodu filtrem . . . 20
9 Ukázka práce funkcí `BackgroundSubtractorKNN` (vlevo) versus `BackgroundSubtractorMOG2` (uprostřed) . . . 23
10 Princip extrakce pozadí a popředí pomocí MoG [9] . . . 23
11 Vytvoření masky popředí pomocí funkce `absdiff` . . . 25
12 Využití Gaussova rozostření v aplikaci na obrázku extrahovaného popředí . . . 25
13 Příklady kouře při požárech [10] [11] [12] [13] [14] [15] . . . 26
14 Graf čtyř fází vývoje požáru . . . 27
15 Ukázka hledání kontur a jejich ohraničujících obdélníků . . . 29
16 Záběry z aplikace při testování prvního videa bez kouře . . . 30
17 Testování videa s diskotékovým kouřem . . . 31
18 Testování videa - požár koše v dílně . . . 32
19 Testování videa - bílá dýmovnice venku . . . 33
20 Testování videa - simulovaný kouř v laboratoři . . . 34
21 Testování videa - video z bezpečnostní kamery v dálničním tunelu . . . 35

Seznam tabulek

1 Tabulka dosažených výsledků z testování 36

Seznam výpisů zdrojového kódu

1 Metoda pro získání masky popředí odečtem pozadí s tolerancí th pomocí metody `cv:absdiff` 24
2 Vlastní struktura Rectangle 28

1 Úvod

Cílem této bakalářské práce je vytvořit software, který bude doplňkem pro detektory kouře, požární hlásiče, umístěné v budovách, jako jsou například školy, obchodní centra a velké továrny či výrobní haly, kde se vyskytuje velké riziko požáru. Tento software bude využívat k detekci kouře obraz ze statické bezpečnostní kamery, kterými jsou výše zmiňované objekty téměř vždy vybaveny, avšak k tomuto účelu bohužel využity nejsou, ačkoliv by jako doplňkový detektor mohly sloužit. Aplikace bude implementována v jazyce C++ s využitím knihovny OpenCV.

V dnešní době existuje spousta čidel detekujících kouř. Problém v praxi je doba, za kterou čidlo spustí poplach (která není zrovna zanedbatelná) a počet falešných poplachů. Čidla se zpravidla umísťují na strop, zatímco kouř ve většině případů stoupá od země, což může vézt k prodloužení doby mezi zahořením a detekcí pomocí tohoto čidla. Kouř totiž musí k čidlu tzv. "doputovat".

Detekce kouře zpracováním obrazu je stále otevřenou výzvou pro vývojáře, kteří se tímto odvětvím zabývají. Obchodní motivace je také velká, jelikož se zvyšují veškerá bezpečnostní opatření tak, aby k velkým požárům nedocházelo. A to hlavně v budovách s vyšší koncentrací osob nebo případných nebezpečí výbuchu vlivem vysoké teploty apod.

Úlohy analýzy obrazu pro detekci kouře nejsou triviální kvůli variabilitě tvaru, pohybu a struktury samotného kouře, který je závislý na světelných podmínkách, rozmanitosti pozadí a barvách scény. Vzhledem k tomu, že kouř modifikuje vizuální podněty pozadí, obvykle se používají techniky potlačení pozadí. Identifikace kouře se stává náročnější v přítomnosti jiných pohyblivých objektů, stínů a také kdykoliv je pozadí proměnlivé.

Práce je rozdělena do čtyř částí, kdy si v první části popíšeme danou problematiku, jak hlásičů požáru, tak detektorů kouře. Ve druhé části práce se budeme zabývat analýzou řešení a přiblížíme teorii, potřebnou k pochopení softwarového řešení a práce obecně. V následující a tedy třetí části provedeme testování na reálných datech, za použití nahraných videosekvencí s výskytem kouře i bez a na závěr zhodnotíme dosažené výsledky vytvořeného softwarového řešení.

2 Algoritmy a techniky použité k detekci

V této části práce si projdeme především teorii ohledně algoritmů a základních technik práce s digitálním obrazem, které jsou v dnešní době nejpoužívanější.

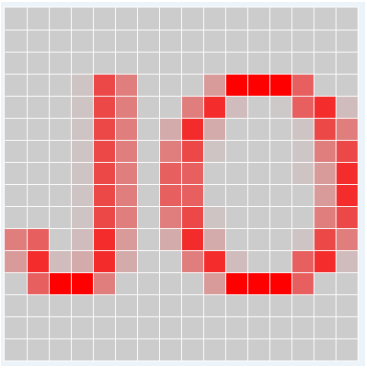
Pojmem zpracování obrazu se především rozumí zpracování dvoudimenzionálního obrazového signálu, který může být ve formě obrázku, videosekvence nebo jiných speciálních formách například ze zdravotnických zařízení, termokamery a dalších. V této oblasti vznikla velká škála různých algoritmů a technik, kdy vám popíši ty nejdůležitější, které lze aplikovat při detekci kouře v obrazech.

2.1 Digitální obraz

V počítačové technice se pod pojmem obraz rozumí binární reprezentace digitálního obrazu. Digitální obraz rozdělujeme na rastrový a vektorový, v závislosti na možnosti změny rozlišení, kdy u rastrového obrázku je rozlišení fixní, na rozdíl od vektorového, kde se rozlišení obrázku může měnit. Pokud není definováno jinak, pak v běžné praxi se pojem digitální obraz s rastrovým často zaměňují.

2.1.1 Rastrový obrázek

Rastrový obrázek je popsán konečným počtem obrazových bodů, kterým říkáme *pixely* (dále budeme používat název pixely). Tyto pixely jsou uspořádány v dvourozměrné mřížce - matici. Každý pixel má přesně definovanou pozici v mřížce, což znamená, že jsou rozměry předem dané a nelze je dále změnit bez ztráty informace. Pixel je nejmenší složka obrázku a jeho hodnota definuje barvu tohoto konkrétního bodu v obrázku. Povolené rozsahy hodnot, kterými může pixel nabývat, jsou definovány dále v kapitole 2.2 Barevné modely. Čím více hodnot může pixel nabývat, tím více barev je možno v daném barevném modelu vyjádřit. Tento počet možných barev se nazývá *barevná hloubka - color depth* obrázku. Rastrové obrázky můžeme získat pomocí digitálního aparátu, digitální kamery, scanneru, apod. Schéma rastrového obrázku o velikosti 16x16 pixelů je znázorněno na obrázku 1, kde jeden čtvereček znázorňuje jeden pixel, a tedy jeden prvek v matici.



Obrázek 1: Rastrový obrázek o velikosti 16x16 pixelů

2.1.2 Digitální video

Digitální video je reprezentováno sérií digitálních obrázků, které jsou zobrazovány v rychlé posloupnosti za sebou konstantní rychlostí. Tyto digitální obrázky, které jsou chápány ve smyslu videosekvence se nazývají *snímky - frames*. Parametrem digitálního videa je počet zobrazovaných snímku za jednotku času. Tento počet se většinou udává ve *snímcích za sekundu*, tedy *frames per second - FPS*.

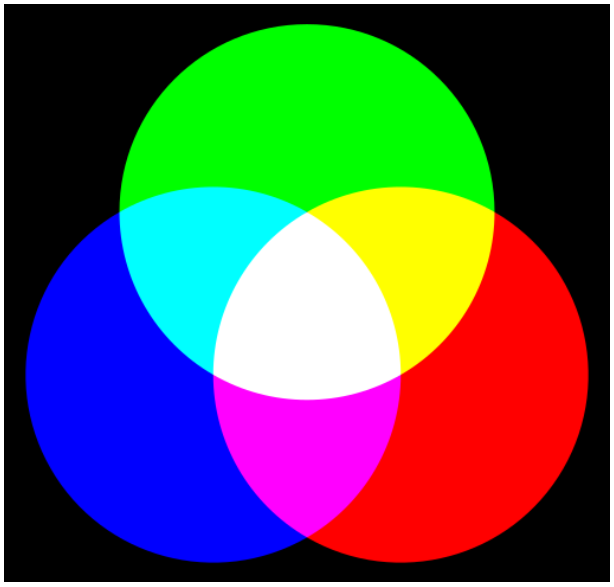
2.2 Barevné modely a prostory

Barevný model, je matematický model, popisující způsob, jakým se dá určitá barva vyjádřit posloupností čísel. Tato posloupnost se většinou skládá z tří až čtyř prvků a to v závislosti na složitosti modelu. Seznam všech barev, které můžeme pomocí daného modelu vyjádřit nazýváme *barevný prostor*.

2.2.1 RGB nebo RGBA model

Barevný model RGB (red, green, blue) je jedním z nejrozšířenějších barevných modelů a to díky jeho využití v počítačových monitorech a projektorech. Jedná se o aditivní model, což znamená, že výsledné barvy dosáhneme mícháním barev a postupným přidáváním jednotlivých složek světla v určitém poměru, jak už ze zkratky vyplývá, jedná se o světlo červené, zelené a modré. Tento poměr je možné vyjádřit v procentech nebo číselně, v závislosti od počtu vyhrazených bitů pro reprezentaci jedné barevné komponenty, přičemž vyšší hodnota znamená větší intenzitu barvy. Typicky může být barevná intenzita jedné ze tří základních komponent vyjádřena osmi bity. Rozsah dané komponenty je tedy v rozmezí od 0 do 255. Jednoduchým výpočtem získáme množství všech barev daného barevného prostoru: $256 \times 256 \times 256 = 16777216$ barev. RGB model je znázorněn na obrázku 2.

U modelu RGBA, což je rozšířený RGB model, přidáváme čtvrtý kanál, tzv. *alfa kanál*, tedy průhlednost barvy jednotlivých pixelů.



Obrázek 2: RGB barevný model [17]

2.2.2 HSV

HSV je barevným modelem, nejvíce odpovídajícím lidskému vnímání barev, na rozdíl od modelu RGB. Skládá se ze tří základních prvků:

1. *Hue* - reprezentuje barevný tón, nebo také barevný odstín. Jeho hodnota je vyjádřená ve stupních v rozsahu $0° - 360°$. Obvykle tento stupeň reprezentuje název barvy, jako je např. modrá.

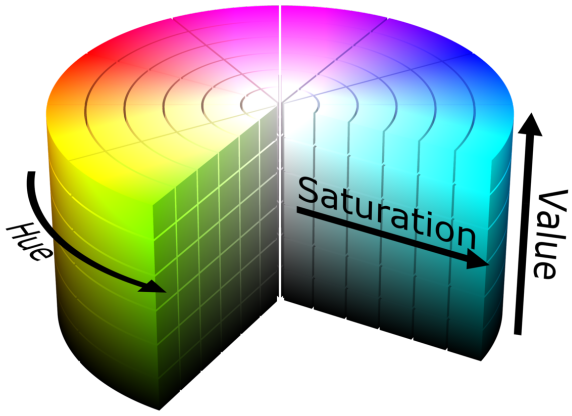
2. *Saturation* - vyjadřuje sytost barvy. Jinak řečeno se může jednat o sílu barvy nebo její čistotu. Vyjadřuje se v procentech, nejčastěji $0\% - 100\%$ nebo desetinným číslem $0,00 - 1,00$.

3. *Value* - reprezentuje jas nebo množství bílého světla, obsaženého v barvě. Jinými slovy pro snížení jasu se přidá do barvy více černé. Udává se podobně jako saturation v procentech v rozsahu $0\% - 100\%$ nebo desetinným číslem $0,00 - 1,00$.

HSV model můžeme vyjádřit i graficky pomocí barevného válce, kde jednotlivé složky jsou znázorněny na obrázku 3 níže.

2.2.3 HSV versus HSL

Barevné modely HSL a HSV jsou si podobné. Výhodou HSL je pojímání sytosti a světlosti jako dvě nezávislé veličiny, což může být intuitivnější. Na druhou stranu v HSL můžou mít téměř bílé barvy 100% sytost, což naopak příliš intuitivní není. Zda je pro uživatelská rozhraní vhodnější HSL či HSV je proto sporné. Výhody HSL jsou v symetrii světlosti a tmy, to znamená:

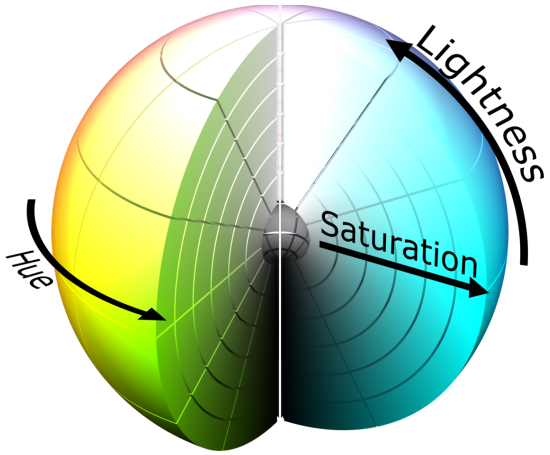


Obrázek 3: Barevný válec HSV modelu [20]

1. Sytost jde vždy od plně syté barvy k ekvivalentu šedé (v HSV, to jde od plně syté barvy k bílé).

2. Světlost má vždy rozsah od černé přes zvolený odstín až k bílé (v HSV, jde poloviční cestou, od černé k volenému odstínu).

HSL model můžeme vyjádřit pomocí koule, kdy jednotlivé složky jsou znázorněny na obrázku 4 níže:



Obrázek 4: HSL mapované do koule s výřezem [19]

2.2.4 Převod RGB do HSV

Složky r, g, b jsou červená, zelená a modrá složka barvy, jejichž hodnoty jsou reálná čísla mezi 0 a 1. Maximální hodnota se rovná největší z nich a minimální hodnota té nejmenší. Následně se spočítají hodnoty (h, s, v) v HSV prostoru, kde $h \in [0, 360]$ je úhel odstínu ve stupních, a $s, v \in [0, 1]$ jsou sytost a hodnota jasu. Vypočítáme:

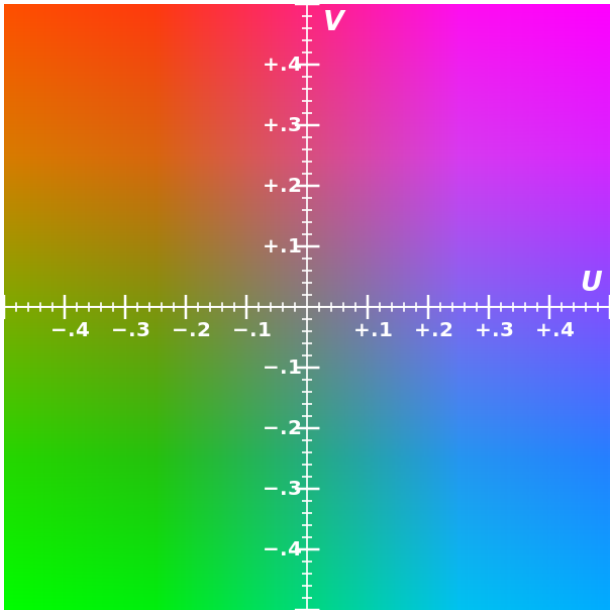
$$h = \begin{cases} \text{nedefinován}, & \text{jestliže } max = min \\ 60° \times \frac{g-b}{max-min} + 0°, & \text{jestliže } max = r \text{ a } g \geq b \\ 60° \times \frac{g-b}{max-min} + 360°, & \text{jestliže } max = r \text{ a } g < b \\ 60° \times \frac{b-r}{max-min} + 120°, & \text{jestliže } max = g \\ 60° \times \frac{r-g}{max-min} + 240°, & \text{jestliže } max = b \end{cases} \tag{1}$$

$$s = \begin{cases} 0, & \text{jestliže } max = 0 \\ \frac{max-min}{max} = 1 - \frac{min}{max}, & \text{jinak} \end{cases} \tag{2}$$

$$v = max \tag{3}$$

2.2.5 YUV

YUV je barevný model používaný v televizním vysílání v normě PAL (jeden ze standardů kódování barevného signálu pro analogové televizní vysílání zavedený v roce 1963 ve Velké Británii a 1967 v Německu) a HDTV (High-definition television - HDTV, neboli televize s vysokým rozlišením označuje formát vysílání televizního signálu s výrazně vyšším rozlišením, než jaké umožňují tradiční formáty jako např. PAL). Model k popisu barvy používá tříprvkový vektor $[Y, U, V]$, kde Y je jasová složka a U a V jsou barevné složky. U je také někdy označováno jako $B - Y$ a V odpovídá $R - Y$. Barevné složky se používají v rozsahu od $-0,5$ do $+0,5$, jasová složka má rozsah $0 - 1$. Výhodou YUV je oddělení jasové složky, kterou člověk přesněji vnímá. Pak je možné vyhradit pro chromatickou složku menší šířku přenosového pásma. Na obrázku 5 níže je znázorněna plocha $U - V$ s hodnotou $Y = 0,5$ uvnitř RGB gamutu (jedná se o dosažitelnou oblast barev v určitém barevném prostoru).



Obrázek 5: Příklad U-V barevné plochy s hodnotou Y' 0,5 uvnitř RGB gamutu [18]

Pro převod modelu RGB do YUV se používá vzorec:

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \times \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{4}$$

2.3 Gaussovy směsi (Mixture of Gaussians)

Gaussovy směsi (Mixtura Gaussiánů nebo také MOG - zkratka z anglického Mixture of Gaussians) je jednou z nejpoužívanějších rekurzivních technik, se kterou se lze v literatuře setkat. Varianta algoritmu, publikovaná Staufferem a Grimsonem, kdy se k modelování jasů pixelů pozadí používají distribuční funkce normálního rozdělení, které bývá také označováno Gaussovo rozdělení. Jednotlivé funkce hustoty pravděpodobnosti reprezentují možné typy povrchů vyskytujících se v místě daného pixelu. Takovýmto způsobem tedy pomocí funkce f modelujeme pravděpodobnost výskytu k-tého povrchu na určité pozici pixelu X v n-rozměrném (kanálovém) obraze. [2] Graf tohoto modelu v 1D můžeme vidět na obrázku 6.

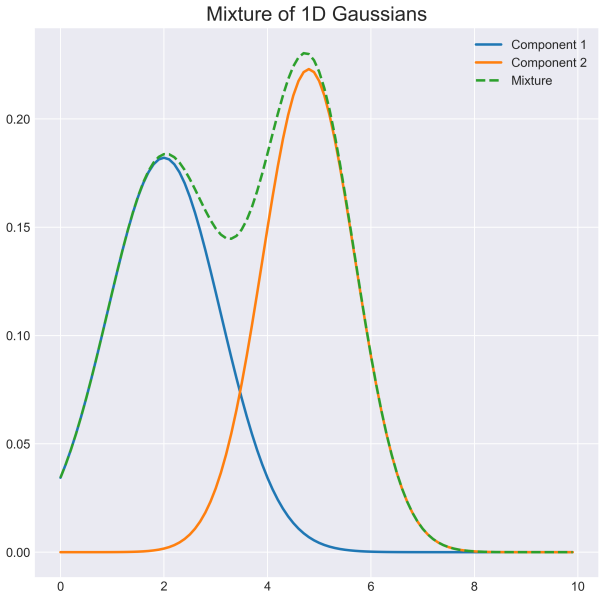
$$f_{X|k}(X|l, \Theta_k) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(X-\mu_k)^T \Sigma_k^{-1}(X-\mu_k)}, \tag{5}$$

Kde $\Theta_k = \{\mu_k, \sigma_k\}$ a diagonální kovarianční matice $\Sigma_k = \sigma_k^2 I$. Index k označuje povrch, objevující se na v daném čase na daném místě zpracovávaného pixelu, jenž utváří výslednou barvu. Předpokládáme, že hodnoty v jednotlivých kanálech jsou vzájemně nezávislé. Jednodušší

implementace bychom docílili, pokud bychom pracovali s jednokanálovým obrazem ve stupních šedi. Funkce by poté byla ve tvaru

$$f_{X|k}(X|l,\Theta_k) = \frac{1}{\sigma_k\sqrt{2\pi}} e^{\left(-\frac{(X-\mu_k)^2}{2\sigma_k^2}\right)}. \tag{6}$$

Jednotlivé funkce vyhodnocujeme pro každý pixel s dynamickou aktualizací pomocí změny parametrů μ_k a σ_k.



Obrázek 6: Gaussův složený model 1D

2.4 Gaussovo rozostření (odstranění šumu)

Odstranění šumu, potlačení šumu nebo redukce šumu je proces, který se pokouší odstranit ze signálu šum. Znalost charakteristik šumu a typu signálu může pomoci k výběru konkrétní metody. Bez znalosti charakteristiky šumu nebo signálu není možné šum odstranit. Nejčastěji činěnými předpoklady o šumu jsou: vysoká frekvence, nízká energie a malá korelace[1] s původním signálem. V digitálním obraze rozeznáváme nejčastěji dva typy šumu:

[1]Korelace znamená vzájemný vztah mezi dvěma procesy nebo veličinami. Pokud se jedna z nich mění, mění se korelativně i druhá a naopak.

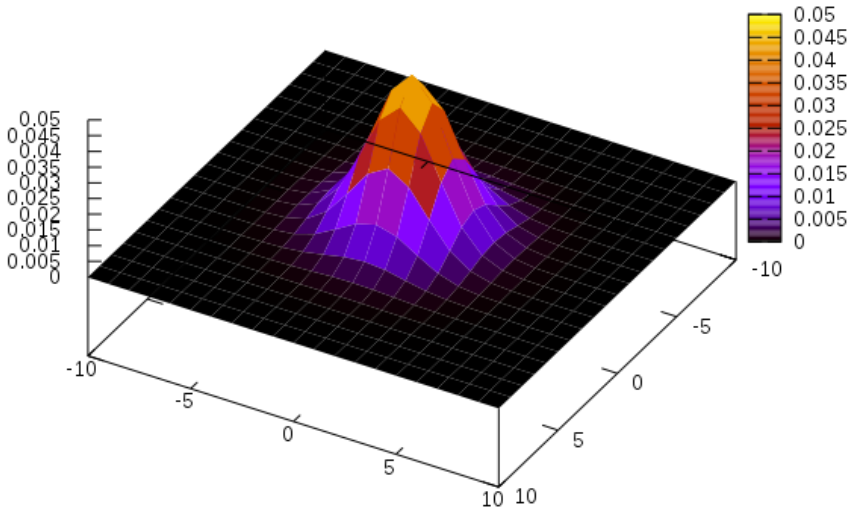
1. náhodný šum, také nezávislý šum, je způsoben například vadnými CCD elementy, což jsou elektronické součástky, které se používají ke snímání obrazové informace (příkladem tohoto typu šumu je šum typu "sůl a pepř")

2. Gaussův šum, také závislý šum, kde je každý pixel obrazu mírně pozměněn

2.5 Metoda odstranění šumu pomocí Gaussovy funkce

Jedna z metod odstranění šumu je konvoluce[2] s maskou, která se skládá z elementů určených Gaussovou funkcí. Tato metoda vede k rozmazání obrázku, což může být pro další zpracování obrazu problém (například pro detekci hran, což je postup ve zpracování obrazu, který slouží k nalezení oblastí, ve kterých se razantně mění jas). Je to efektivní technika k potlačení Gaussova šumu. Vzorec pro 2D Gaussovu funkci:

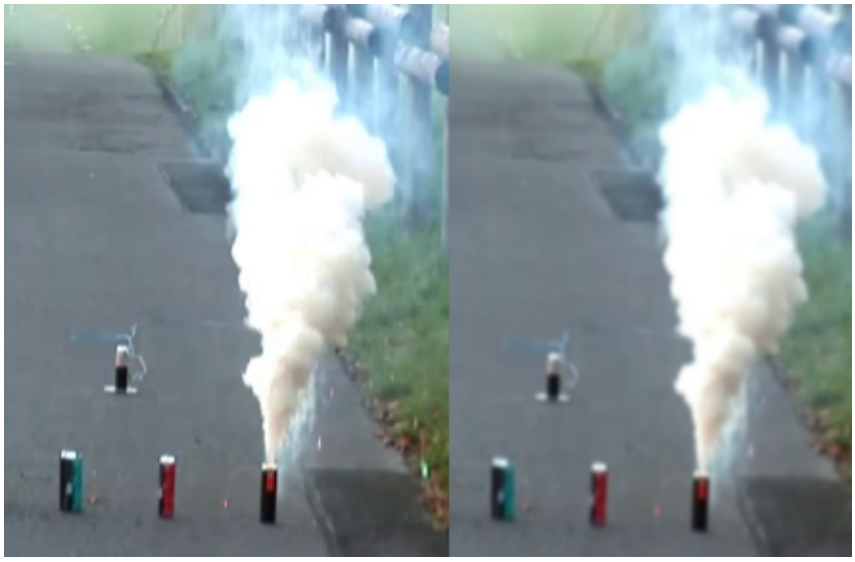
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{7}$$



Obrázek 7: Ukázka konvoluční matice Gaussovy funkce pro okolí 10 pixelů a směrodatnou odchylku 1,8. [16]

Pro potřeby aplikace rozmazáváme detekované popředí pro redukci šumu celkového obrázku a snadnější detekci kouře.

[2]konvoluce je matematický operátor, zpracovávající dvě funkce.



Obrázek 8: Ukázka redukce šumu pomocí Gaussova rozostření. Vlevo originální obrázky, vpravo po průchodu filtrem

2.6 Nalezení kontur (obrysů) v obraze

Kontury lze vysvětlit jednoduše jako křivky, spojující všechny spojité body (podél hranice), které mají stejnou barvu nebo intenzitu. Detekce kontur je užitečným nástrojem pro analýzu tvaru a detekci a rozpoznávání objektů.

Pro lepší přesnost používáme binární obrázky (doporučováno také v dokumentaci OpenCV [1]). Takže před nalezením kontur použijeme prahovou hodnotu nebo detekci okrajů. V OpenCV je nalezení kontur podobné jako hledání bílého objektu z černého pozadí. Takže objekt, který se má najít, by měl být bílý a pozadí by mělo být černé.

3 Analýza řešení problému

V této části bude znázorněna implementace a konkrétní využití technik a algoritmů, které byly použity v této práci. Na základě tří odlišných prací, věnovaných se problematice detekce kouře, jsme vybrali části řešení, které již byly testovány a vypracovány v pracech, uvedených v referencích. Z práce [4] jsme vybrali extrakci pozadí pomocí MOG. Z druhé zmiňované práce [5] jsme vybrali porovnání barev RGB modelu a ze třetí práce [6] analýzu změny tvaru každé oblasti, která by mohla být adeptem na oblast s výskytem kouře.

3.1 Knihovna OpenCV

OpenCV je otevřená multiplatformní knihovna pro manipulaci s obrazem. Zaměření této knihovny je především na počítačové vidění a zpracování obrazu v reálném čase. Původně tuto knihovnu vyvíjela společnost Intel. Dokáže se tedy zrychlit ve spolupráci s knihovnou Intel IPP - Integrated Performance Primitives[3]. Knihovnu je možné využít v prostředí jazyků C, C++ a s generátorem rozhraní SWIG (vývojářský nástroj, který propojuje programy psané v C/C++ s vysokoúrovňovými programovacími jazyky) také Python (vysokoúrovňový skriptovací programovací jazyk, který v roce 1991 navrhl Guido van Rossum) a Octave (svobodný software pro provádění číselných výpočtů šířený pod licencí GPL, do určité míry kompatibilní s programem MATLA). Knihovnu můžeme spouštět na obou platformách, jak stolní počítače (operační systémy Windows, Linux, Android, MacOS, FreeBSD a OpenBSD), tak také na mobilních zařízeních se systémy Android, Maemo a iOS [1].

3.2 Extrakce pozadí a vytvoření masky popředí

Odčítání pozadí je důležitým předzpracováním v mnoha aplikacích, založených na zpracování obrazu. Zvažme například případy jako počítadlo návštěvníků, kde statická kamera zaujme počet návštěvníků, kteří vstupují do místnosti nebo z ní odcházejí, nebo dopravní kamera, která získává informace o vozidlech apod. Ve všech těchto případech je nejprve třeba extrahovat osoby nebo vozidla samotné. Z technického hlediska je nutné vyjmout pohyblivé popředí ze statického pozadí.

Pokud máme pouze obrázek pozadí, jako je například obraz místnosti bez návštěvníků, je to snadná práce. Stačí odečíst nový obrázek z pozadí. Tím získáme pouze objekty v popředí. Ve většině případů však nemusíme mít takový obrázek nebo se můžou měnit světelné podmínky (např. zajde slunce a následně vyjde, rozsvítíme a zhasneme, apod.), takže musíme extrahovat pozadí z jakýchkoli obrázků, které máme. Stává se to komplikovanější, když se na obrázku vyskytuje stín vozidel, osob apod. Vzhledem k tomu, že se stín také pohybuje, jednoduché

[3]Při použití funkcí knihovny IPP dochází k optimalizaci výpočetních algoritmů a tím k urychlení času potřebného k vykonání celé aplikace.

odečtení bude stíny považovat jako popředí. Pro tento účel bylo v OpenCV vytvořeno několik funkcí, využívajících různé algoritmy. Nás budou zajímat dvě hlavní funkce a to:

1. `BackgroundSubtractorMOG2`
2. `BackgroundSubtractorKNN`

3.2.1 BackgroundSubtractorMOG2

Funkce segmentace pozadí a popředí, využívající algoritmus, založený na Gaussově směsi na pozadí nebo popředí v segmentaci. Je založen na dvou studiích [7] a [8]. Jednou z důležitých rysů tohoto algoritmu je, že vybírá odpovídající počet gaussovských distribucí pro každý pixel. Poskytuje lepší přizpůsobivost různým scénám díky změnám osvětlení apod.

Nejprve je třeba vytvořit objekt `BackgroundSubtractor`. Zde si můžeme zvolit, zda chceme detekovat stín nebo ne. Pokud `detectShadows = True` (což je standardně), detekuje a označuje stíny, ale snižuje rychlost. Stíny budou označeny šedou barvou. V práci [8] je popsána vylepšená adaptace algoritmu MOG, který se v této funkci používá. Dále je k dispozici podrobná dokumentace funkce [1].

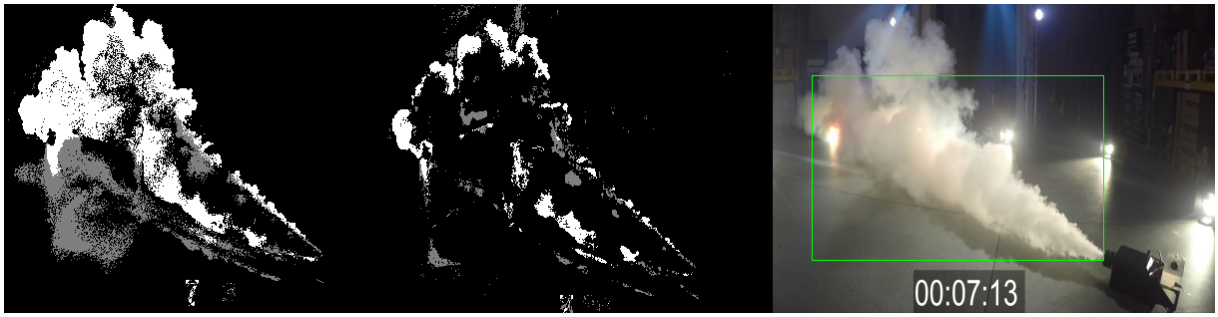
3.2.2 BackgroundSubtractorKNN

Funkce segmentace pozadí a popředí, využívající algoritmus K-nejbližších prvků. Tento model se však více využívá, pokud pixelů v popředí je co nejméně [7]. Tento algoritmus je metoda pro učení s učitelem, kdy se klasifikují prvky, reprezentované vícedimenzionálními vektory, do dvou nebo více tříd. Ve fázi učení se předzpracuje trénovací množina tak, aby všechny příznaky měly střední hodnotu 0 a rozptyl 1 - toto umístí každý prvek trénovací množiny do některého místa v N-rozměrném prostoru. Ve fázi klasifikace umístím dotazovaný prvek do téhož prostoru a najdu k nejbližších sousedů. Objekt je pak klasifikován do té třídy, kam patří většina z těchto nejbližších sousedů.

Pokud je $k = 1$, jde o speciální zjednodušený případ, metodu nejbližšího souseda.

Pro hledání nejbližšího souseda v množině se nejčastěji používá euklidovská metrika nebo Hammingova metrika.

V této práci pro extrakci pozadí používáme MOG2 model, jejíž algoritmus je popsán v kapitole 2.3 výše. Srovnání a ukázku výsledků masky detekovaného popředí můžeme vidět na obrázku 9. Při vývoji této aplikace jsme však zjistili, že funkce `BackgroundSubtractorKNN` sice lépe detekovala popředí, ale na úkor rychlosti. Za další tato funkce nenabízí možnost extrahovat obrázek pozadí, což výše zmiňovaná `BackgroundSubtractorMOG2` nám tuto možnost nabízí.



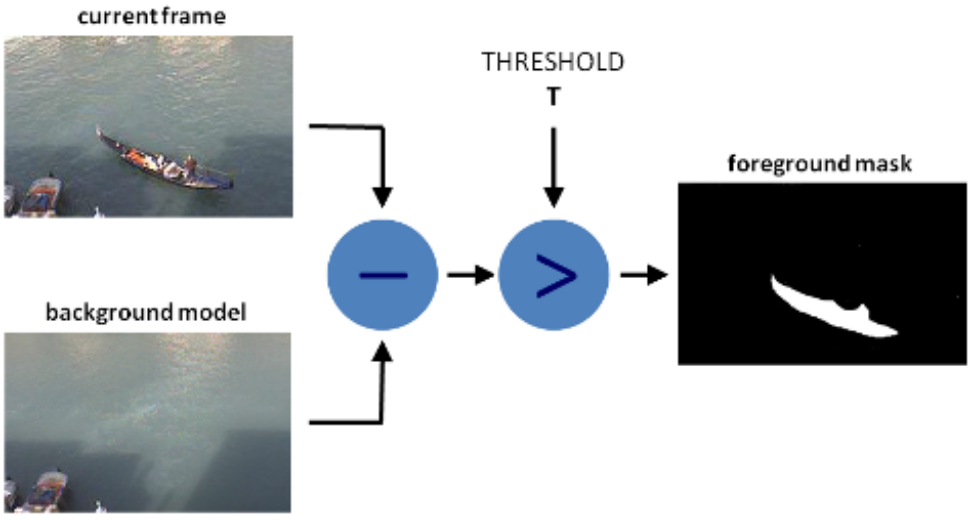
Obrázek 9: Ukázka práce funkcí `BackgroundSubtractorKNN` (vlevo) versus `BackgroundSubtractorMOG2` (uprostřed)

3.2.3 Inicializace a aktualizace pozadí

Modelování pozadí se skládá ze dvou základních kroků:

1. Inicializace pozadí
2. Aktualizace pozadí

V prvním kroku se vypočítá počáteční model pozadí z předem zvoleného počtu snímků (10), zatímco ve druhém kroku je tento model aktualizován, aby se přizpůsobil možným změnám ve scéně (v případě této aplikace se jedná o 200 snímků), zároveň však ošetřujeme, zda-li nemáme adepty na kouř již ve stádiu rozpracování, jednalo by se o případ, kdy máme jednu nebo více potvrzených oblastí s kouřem, uložených v poli a tato oblast nebo oblasti s kouřem byly již potvrzeny (a uloženy v poli) na více, než polovině kapacity tohoto pole, tedy počtu snímků, které jsou nutné k potvrzení výskytu kouře v detekované oblasti. Grafické znázornění principu extrakce pozadí, v tomto případě z prvních 10 snímků, můžeme vidět na obrázku 10



Obrázek 10: Princip extrakce pozadí a popředí pomocí MoG [9]

3.2.4 Extrakce popředí

Extrakci popředí provádíme porovnáváním snímku pozadí, který jsme si již inicializovali, popřípadě aktualizovali, a aktuálního snímku pomocí metody `cv::absdiff(InputArray src1, InputArray src2, OutputArray dst)`. Tento rozdíl však vrátí i pixely, které mohly být modifikovány například ztrátovou komprimací kodeku videa, či vyvážením kontrastu v hardwaru kamery. Stručně řečeno tato metoda vrátí rozdílné pixely s nulovou tolerancí.

Pro tento případ aplikujeme zvolený práh $th = 90$ (tolerance), kdy součet hodnot rozdílu všech kanálů aktuálního snímku je větší, než daný práh th. Pokud je podmínka splněna, pak do masky popředí zapíšeme tento bod. Na konci tato metoda vrátí novou masku popředí s redukovanými pixely.

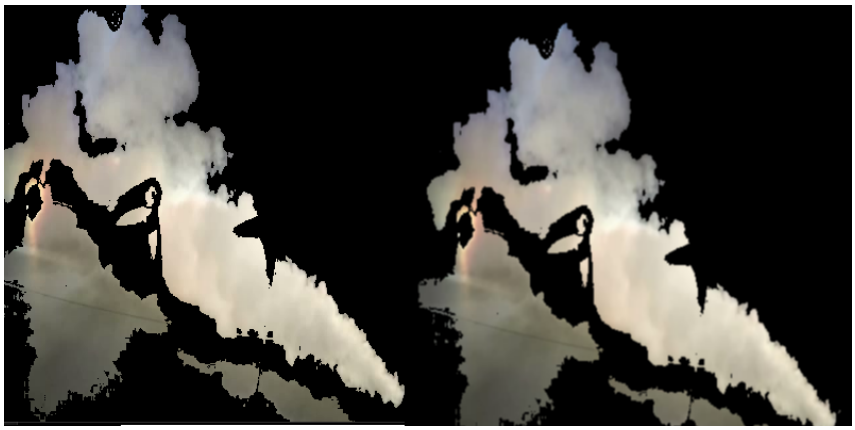
```
Mat reduceNoise() {
   Mat diff;
   absdiff(backgroundImage, frame, diff);
   int th = 90;
   Mat mask(backgroundImage.size(), CV_8UC1);
   mask = Scalar::all(0);
   for (int j = 0; j<diff.rows; ++j) {
      for (int i = 0; i<diff.cols; ++i) {
         cv::Vec3b pix = diff.at<cv::Vec3b>(j, i);
         int val = pix[0] + pix[1] + pix[2];
         if (val>th) {
            mask.at<unsigned char>(j, i) = 255;
         }
      }
   }
   return mask;
}
```

Výpis 1: Metoda pro získání masky popředí odečtem pozadí s tolerancí th pomocí metody `cv:absdiff`

Na obrázku 11 můžeme vidět názornou ukázku při detekování popředí, kde vlevo se nachází extrahovaný obrázek pozadí, uprostřed aktuální snímek a napravo je výsledná maska. Bílé pixely v masce znamenají, že se jedná o popředí. To vše je za pomocí funkce `cv::absdiff`, jak jsme již zmiňovali dříve.

Následně aplikujeme Gaussovské rozostření, abychom eliminovali šum v obraze a zlepšili tím výsledek ověřování barvy, kterému se budeme věnovat v následující kapitole. Ukázku použití Gaussovského rozostření při chodu aplikace můžeme vidět na obrázku 12.

Obrázek 11: Vytvoření masky popředí pomocí funkce `absdiff`



Obrázek 12: Využití Gaussova rozostření v aplikaci na obrázku extrahovaného popředí

3.3 Ověřování barvy

Dalším krokem v naší aplikaci je ověřování barvy pixelů. Zde využíváme dva barevné modely. Jednak je to základní barevný model RGB (2.2.1), ale také barevný model HSV (2.2.2).

Ze začátku hoření, kdy je teplota kouře nízká, bude mít kouř světle šedou barvu, která se bude přiklánět spíše k bílé. Postupem času se bude teplota kouře zvyšovat a kouř začne tmavnout. Barva kouře tedy bude tmavší šedá, až téměř černá. Jak můžete vidět z příkladů kouře na obrázcích 13, většinou je barva kouře šedá. Můžeme tedy barvu kouře specifikovat následujícím způsobem:

$$\begin{aligned} |R(x,y) - G(x,y)| &\leq Th \\ |G(x,y) - B(x,y)| &\leq Th \\ |R(x,y) - B(x,y)| &\leq Th \end{aligned} \tag{8}$$

kde Th je globální proměnná v rozmezí od 15 do 25. Tato rovnice nám v podstatě říká, že rozdíl mezi hodnotami jednotlivých barevných složek je si podobný, a tedy se jedná o šedou barvu daného pixelu na pozici (x, y). Chceme-li tedy odhalit kouř, v co možná nejkratším čase, pak bychom měli hledat nejsvětlejší odstíny šedé, což znamená, že světlost barvy (saturace) by měla být co nejnižší. Touto úvahou můžeme tedy říci, že další pravidlo pro porovnání barvy, tentokrát podle HSV modelu, které spojíme s ověřováním barvy pixelu dle RGB modelu výše.

Proto volíme: $S(x, y) \leq 0.1$. Na obrázku 13 níže můžeme vidět ukázky kouřů z praxe.

Obrázek 13: Příklady kouře při požárech [10] [11] [12] [13] [14] [15]

3.4 Ověřování pohybu

Vývoj kouře v průběhu požáru, a tedy v každé ze čtyř fází požáru, až do jeho zániku mívá různé charakteristiky. Fáze požáru jsou důležitou charakteristikou popisující vlastnosti požáru při jeho volném rozvoji, tzn. v případě, že není hašen. Fáze požáru jsou následující:

1. fáze je určena časem od vzniku požáru rozhoření prvních hořlavých předmětů. V praxi se uvažuje čas 10 minut. Tato fáze je charakterizována nízkými teplotami a malou výměnou plynů. Tuto fázi rovněž označujeme jako *fázi rozhořívání*. V této fázi bude kouř velice

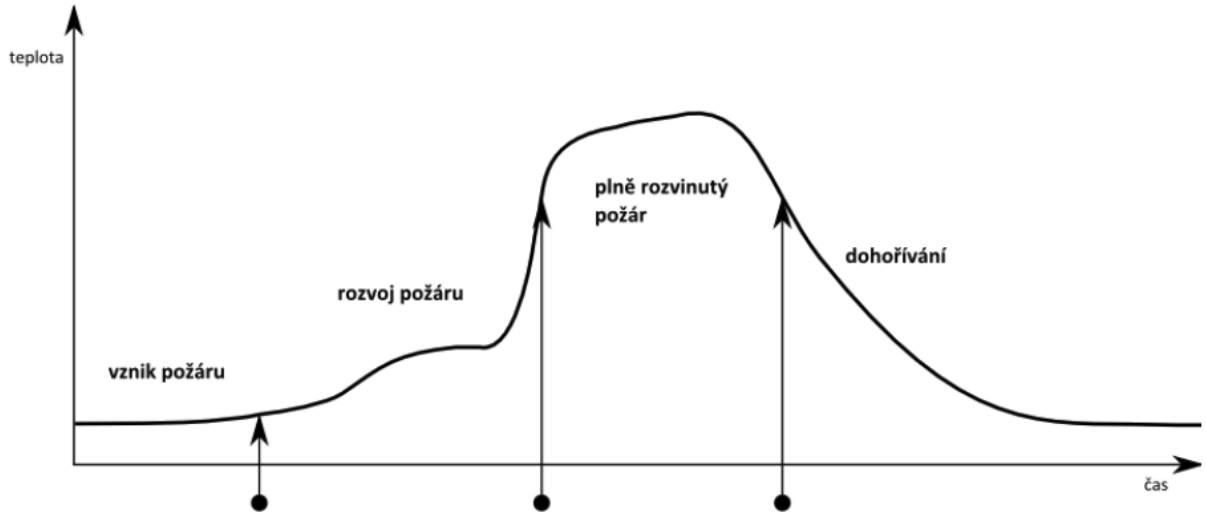
světlý až bílý, jak bylo uvedeno v kapitole 3.3. Tato fáze nás zajímá nejvíce, jelikož nám jde o rozpoznání kouře, tedy zachycení požáru při rozhořívání

2. fáze je charakterizována prudkým nárůstem teploty a plochy požáru, zejména v souvislosti s celkovým vzplanutím (též flashover) je označení pro jev, při kterém téměř současně vzplanou všechny hořlavé látky v místě požáru (v uzavřeném prostoru)

3. fáze je období, kdy požár je stabilizován, probíhá intenzivní hoření a požárem jsou zachváceny všechny hořlavé předměty v prostoru.

4. fáze je charakterizována nedostatkem hořlavého materiálu a postupným snižováním intenzity hoření.

Na grafu níže na obrázku 14 jsou znázorněny všechny čtyři fáze požáru. Jelikož se v aplikaci zabýváme detekcí kouře při samotném prvopočátku, pak nás bude zajímat pouze první fáze požáru, kde můžeme, jak z grafu vyplývá, vidět, že kouř při první fázi požáru bude mít nízkou teplotu, která se časem pomalu zvyšuje a celková energie proudění vzduchu v okolí zdroje hoření nebude velká, ani se razantně měnit. Z toho vyplývá, že nás zajímá světlý kouř, který se bude pomalu šířit a zakouřená oblast na obrázku se bude zvětšovat. V dalších fázích požáru by detekce byla složitější, ale také bezpředmětná. Jde přece o co nejrychlejší detekci.



Obrázek 14: Graf čtyř fází vývoje požáru

Vezmeme-li v úvahu, že detekujeme kouř v uzavřených prostorách, pak by měl dým postupovat směrem vzhůru a oblast, na které se kouř objeví by se měla postupně zvětšovat. Problém může nastat, pokud se jedná o detekci kouře v různých průmyslových halách, kde se můžou skladovat tlakové lahve, či nádoby s hořlavými kapalinami.

Tlakové lahve, skladované v průmyslových halách představují problém, kdy při proražení pláště neopatrným zacházením hořlavá látka uniká pod tlakem do boku. V případě vzplanutí

takovéto látky by potom kouř nestoupal pozvolna vzhůru, nýbrž prudce do boku a jeho oblast by se zvětšovala pouze v prvních pár okamžicích, poté se oblast kouře téměř ustálí, jelikož nejsme na otevřeném prostranství. Chování kouře však závisí na mnoha aspektech, díky kterým bychom v praxi museli naši aplikaci poupravit, aby fungovala správně v závislosti na místě, kde by byla umístěna.

V této práci se snažíme najít řešení pro haly v obchodních domech, či školách a dalších veřejných prostorech, kde jsou velikosti místností vcelku velké a my tak máme více prostoru v obraze pro detekování kouře, a tím je naše aplikace přesnější.

Z práce [6] jsme převzali myšlenku, kdy z nalezené oblasti, kterou jsme detekovali pomocí ověřování barvy pixelů, vypočteme obdélník, ohraničující tuto oblast a velikost oblasti (počet pixelů v oblasti obrázku, které mají šedivou barvu podle ověřování barev v kapitole 3.3).

Uchováváme hodnoty $H(t), H(t-1), \ldots, H(t-k)$ těchto obdélníků a oblastí, kde k je velikost pole, uchovávajícího tyto hodnoty (v našem případě $k = 10$), kdy každý prvek tohoto pole je vektor typu vlastní struktury `Rectangle`, kterou jsme si vytvořili pro snadnější a rychlejší porovnávání aktuální oblasti a jejího ohraničujícího obdélníku s posledními podobnými detekovanými a uloženými hodnotami, ale také abychom měli všechna data pohromadě, tedy ohraničující obdélník i oblast šedivých pixelů zároveň.

```
struct Rectangle
{
   // Ohraničující obdélník
   cv::Rect rect;
   // Oblast šedivých pixelů
   double area;
   // Proměnná, udávající, zda-li se jedná o poslední přidanou strukturu
   bool last;
};
```

Výpis 2: Vlastní struktura Rectangle

V každém snímku procházíme všechny detekované oblasti, které dostaneme pomocí OpenCV metody `findContours(InputOutputArray image, OutputArrayOfArrays contours, int mode, int method, Point offset=Point())`, která vyhledá kontury[4] z masky popředí. Vstupem této metody jsou parametry: zdrojový, 8-mi bitový jednokanálový snímek, ze kterého budeme obrysy detekovat, výstupní pole, kde se budou ukládat nalezené kontury, kdy každá kontura je uložena jako vektor bodů, volitelný výstupní vektor obsahující informace o topologii obrazu a režim vyhledávání obrysů.

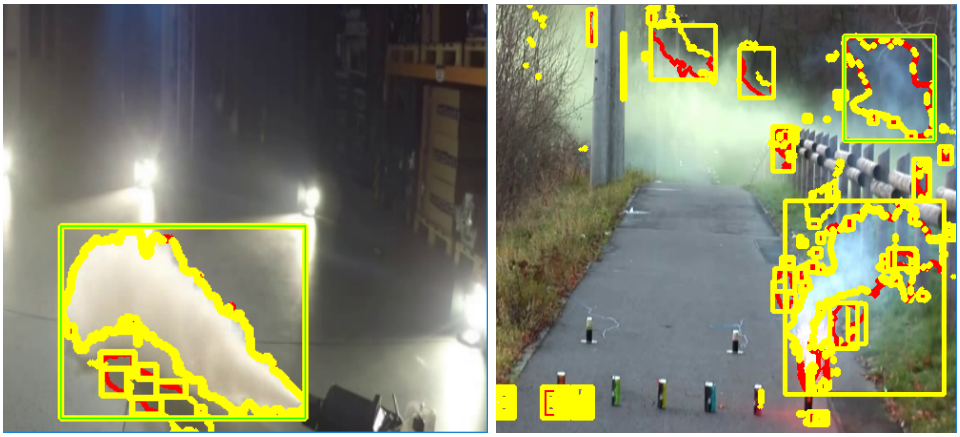
Z těchto kontur dostaneme pomocí OpenCV metody `boundingRect(InputArray points)`, jak už z názvu metody vyplývá, ohraničující obdélník této kontury.

[4] kontury nebo také obrysy

Dále potřebujeme získat velikost oblasti. Pro tuto potřebu jsme využili opět metodu z knihovny OpenCV `contourArea(InputArray contour, bool oriented=false)`, kde je vstupním parametrem metody množina bodů 2D, uložená v `std::vector` nebo `cv::Mat` a volitelný parametr značka orientace. Je-li `true`, funkce vrací hodnoty orientovaných obrysů, a to ve směru hodinových ručiček nebo proti směru hodinových ručiček.

Konečně se dostáváme k ověřování změny pohybu ohraničujícího obdélníku a změny velikosti oblasti, které provádíme tak, že si procházením pole najdeme ve struktuře `Rectangle` a její položce `cv::Rect rect` shodu x-ové a y-ové souřadnice ohraničujícího obdélníku s tolerancí `t = sqrt(input.size().height * input.size().width) * 0.05;`, kde `input` je vstupní snímek, pouze za předpokladu, že ověřovaný obdélník je námi poslední ověřovaný a uložený. To zjistíme pomocí položky `last` struktury `Rectangle` a `t` je tolerance ohraničujícího obdélníku na x-ové a y-ové ose. Tuto konstantu vypočteme jako 5% z druhé odmocniny šířky snímku, vynásobenou výškou snímku ($t = \sqrt{height * width} * 0.05$).

Nalezneme-li shodný obdélník, pak ověřujeme ještě velikost aktuální oblasti, která musí být větší, než poslední velikost oblasti, detekovaná v minulém snímku s tolerancí 25% své plochy a to kvůli struktury kouře, která má svou specifickou texturu a která se stále mění, tudíž při testování na videosekvencích a vývoji, oblast s kouřem jen nerostla, ale v rozsahu 25% kolísala. Detekce kontur a ohraničujících obdélníků z masky popředí ve snímku je znázorněna na obrázku 15.



Obrázek 15: Ukázka hledání kontur a jejich ohraničujících obdélníků

4 Testování

Takto aplikované řešení jsme testovali na pěti videosekvencích. Pro testování jsme si vybrali pět videí, na kterých se kouř objevoval a jedno video, kde byl pohyb na scéně, ale kouř se zde neobjevoval. Videosekvence máme ze statické kamery, převážně z vnitřních prostor.

4.1 Video 1 - motokára

První videosekvence (obrázek 16) zachycuje jezdce v motokáře na dráze v hale, kde je pohyb na scéně vcelku velký a také samotná scéna různobarevná s šedou podlahou, která by mohla vézt k falešným detekcím. V tomto videu se kouř nevyskytuje. Aplikace na videu správně nerozeznala žádný kouř a nedošlo zde ani k falešným detekcím.

Obrázek 16: Záběry z aplikace při testování prvního videa bez kouře

4.2 Video 2 - diskotékový kouř

Na druhém testovacím videu je vyvíječ kouře pro diskotéky a různé společenské akce (obrázek 17). Tento kouř sice není nijak závadný pro lidský organismus a okolí, ale jedná se o kouř, takže nás zajímalo, zda aplikace tento kouř detekuje.

Při testování této videosekvence se kouř objeví (jak můžeme vidět na prvním snímku vlevo nahoře na obrázku 17), avšak aplikace díky ověřování pohybu a vývoje oblasti obsahující kouř tuto oblast detekuje až se zpožděním $k = 15$ snímků.

Na tomto videu byl kouř úspěšně detekován ve všech snímcích, kromě prvotních k - snímků, které následovaly po prvotním objevení kouře na scéně, a které jsou potřeba k analýze pohybu a eliminaci falešných detekcí - jedná se tedy o toleranci aplikace.

Obrázek 17: Testování videa s diskotékovým kouřem

4.3 Video 3 - požár odpadkového koše v truhlářské dílně

Třetí video je z praxe, kdy v truhlářské dílně došlo k požáru koše (obrázek 18). V tomto videu jsou světelné podmínky velice špatné a rozlišení je malé. I přes tuto skutečnost fungovala aplikace při testování dobře. Na videu jde vidět, že se nejprve objevuje oheň, který celou detekci výrazně ovlivňuje, jelikož ozařuje oblasti, s potencionálním výskytem kouře. Samotný kouř by v těchto podmínkách byl jen stěží detekovatelný.

Z videa je patrné, že odpadkový koš hoří. My však pouhým okem jen těžko zjistíme, zda-li na snímku 135 (první obrázek vlevo nahoře) kouř ve videosekvenci je. Na snímku 228 (vpravo nahoře) můžeme na pohled vidět první kouř, ale naše aplikace kouř detekuje až na snímku 257 (vlevo dole), kdy je kouř detekován ve všech snímcích až do snímku číslo 449 (vpravo dole). Ke konci videosekvence nám detekci velice ztěžuje rozvíjející se požár a výrazné kolísání barevnosti a osvětlení scény.

Obrázek 18: Testování videa - požár koše v dílně

4.4 Video 4 - bílá dýmovnice venku

Čtvrté video) je natočeno venku na chodníku se zábavnou pyrotechnikou - dýmovnicí (obrázek 19. Dýmovnice není nijak zbarvená, jedná se o bílou dýmovnici. Na této videosekvenci je problém při detekování pozadí, jelikož autor videosekvence natáčel nejspíše více dýmovnic v krátkém časovém sledu po sobě, které rozdělil do více videosekvencí a tak se v pozadí kouř vyskytuje již od začátku.

Obrázek pozadí, extrahovaný z videosekvence tedy obsahuje již od začátku oblasti, v nichž se vyskytuje kouř. To může být problém při odečítání aktuálního snímku od pozadí v případě, kdy se oblast s potenciálním výskytem kouře objeví na místech, kde se již od začátku vyskytuje kouř, který bude obsažen v obrázku extrahovaného pozadí, a tedy by mohla být barva pixelu na témže místě velice podobná až totožná na aktuálním snímku i na pozadí. Tento případ by vedl k ignorování oblastí, které jsou adepty na místa s výskytem kouře ve videosekvenci.

Výsledek testu je překvapivý. Byť se v pozadí nacházely oblasti s kouřem, naše aplikace přesto rozpoznala již drobný kouř na začátku videosekvence, který se vyskytoval již od prvopočátku.

Obrázek 19: Testování videa - bílá dýmovnice venku

4.5 Video 5 - simulovaný kouř v laboratoři

Další videosekvencí je simulovaný kouř v laboratoři (obrázek 20). V této videosekvenci jsem objevil falešnou detekci, která je způsobená mužem, oděným v šedé košili, který prochází v blízkosti kamery. Kouř aplikace rozpoznala, ale chvílemi oblast s kouřem vynechávala.

Obrázek 20: Testování videa - simulovaný kouř v laboratoři

4.6 Video 6 - nehoda v tunelu

Poslední testovanou videosekvencí je nehoda automobilu s následným požárem v dálničním tunelu (obrázek 21). V této videosekvenci je spousta projíždějících aut, které mají šedivou až bílou barvu. Tento fakt vede k falešným detekcím, v případech, kdy nákladní automobil vjíždí do scény nebo se pozadí aktualizuje když jiný automobil projíždí scénou a ten po této aktualizaci obrázku pozadí, která zachytí jako pozadí i právě projíždějící automobil, odjede. Falešné detekce se v této videosekvenci objevují a je to z velké míry dáno aktualizováním pozadí v čase, kdy projíždí auto, které se stane součástí obrázku pozadí.

Obrázek 21: Testování videa - video z bezpečnostní kamery v dálničním tunelu

4.7 Shrnutí dosažených výsledků

Testováním jsme došli k výsledkům uvedeným v tabulce 1 níže. Všechny hodnoty kromě úspěšnosti a falešných detekcí jsou ve snímcích. Falešné detekce jsou v počtu falešných detekcí za videosekvenci a úspěšnost, jak je uvedeno je v procentech.

Tabulka 1: Tabulka dosažených výsledků z testování

videosekvence	**správně**	**falešné detekce**	**celkem snímků**	**úspěšnost**
motokára	365	0	365	100%
diskotékový kouř	185	0	200	92.5%
odpadkový koš	505	0	651	77.57%
bílá dýmovnice venku	163	0	233	69.95%
kouř v laboratoři	596	1	668	48.95%
tunel	507	5	654	61.77%
celkem	2321	6	2771	83.76%

5 Závěr

Pro detekci kouře jsme v této práci použili Gaussův smíšený model, který je využíván v OpenCV funkci `BackgroundSubtractorMOG2` k extrakci pozadí a popředí. Následné rozeznávání a porovnávání barev v RGB modelu společně s HSV modelem s sebou nese jistá úskalí.

Jak jsme zjistili při testování, samotné ověřování barvy pro rozpoznání kouřových oblastí nestačí. Tato technika samotná je použitelná, pokud se v okolí nevyskytuje mnoho předmětů (osob či automobilů), které mají taktéž šedou barvu. Takovýto výskyt nežádoucích předmětů vede k falešným detekcím.

Tento problém je částečně možné řešit pomocí analýzy pohybu a velikosti oblasti s potenciálním výskytem kouře. Při vývoji této aplikace jsme však narazili na to, že se oblast s kouřem né vždy zvětšuje. Tento fakt ovlivňuje několik aspektů. Jednak je to tvar kouřového oblaku, ale také úhel šíření kouře vůči úhlu snímání kamery. Kouř může v nevýhodném postavení kamery zabírat pouze malou část obrazu, zatímco se bude rozšiřovat po ose zorného pole kamery, tedy od kamery do prostoru. Takovýto kouř je pro nás hůře, až nemožně detekovatelný. V praxi se takovéto problémy však řeší (také kvůli rozpoznání tváří zlodějů) umístěním několika kamer v různých rozích místností, pod různými úhly atp.

Během vývoje jsme porovnávali různé barevné modely a nakonec dospěli k závěru, že pro naše využití v aplikaci byla nejlepší kombinace HSV a RGB. Jak bylo zmiňováno v kapitole 2.2 HSL může mít při téměř bílé barvě 100% sytost, což v tomto případě nechceme. YUV barevný model je možné použít namísto HSV modelu, jelikož má první složku jasovou obdobně jako HSV saturaci.

Různý materiál hoří různým způsobem. Některý produkuje hustý černý kouř, jiný zas téměř žádný. Osvětlení objektu také přispívá k výsledné barvě kouře a ovlivní detekci změn v popředí. Bezpečnostní kamery v budovách a podobných objektech, často snímají jen určitou část prostoru, kde se případný hořlavý materiál dá předem určit a otestovat. Podobně osvětlení může být stálé a rovnoměrné během celého dne, což vede k ideálnímu případu nasazení systému.

Jiná situace nastává na veřejných prostranstvích, kde se vyskytuje velké množství různých objektů, často i proměnlivého charakteru, což můžou být kupříkladu jedoucí auta po silnici. Světelné podmínky se také během dne pochopitelně dramaticky mění, proto je nasazení sytému do takového prostředí (v našem případě prosklené haly letišť, obchodních domů atp.) komplikovanější. V noci je prakticky nemožné detekovat kouř, zde by přišla na řadu nejspíše detekce ohně.

5.1 Zhodnocení přínosu a návrhy k rozšíření

Ve vývoji detekce, založené na popsaném systému, se dají použít další rozšířené techniky, které by vedly k co největší kvalitě výstupu algoritmu. Jedním rozšířením by mohlo být rozšíření implementace i o vstup, který není snímaný jen statickou kamerou. Problémem však je, že

detekce pohybu kouře způsobem ověření pohybu a barevnosti vyžaduje, aby v 15 snímcích za sebou byla splněna kritéria pro detekci, která by se tímto způsobem nedala možná dosáhnout.

Analýza samotného pohybu a vývoje kouře byla spíše experimentální, což vedlo k falešným detekcím, či nedetekovanému kouři. Tento nedostatek by se dal vylepšit metodou strojového učení, kdy by se analýzou datasetu kouřů model naučil obvyklé tvary a texturu kouře i s barevností. Tato metoda by nejspíš vedla k výraznému zlepšení dosažených výsledků a mohla by zmenšit počet falešných detekcí.

Literatura

[1] OpenCV dokumentace: `https://docs.opencv.org`

[2] Ing. Jan Gaura, Ph.D.: Mixtura Gaussiánů

[3] Hongda Tian, Wanqing Li, Lei Wang, and Philip Ogunbona: *Smoke Detection in Video: An Image Separation Approach.* Int. J. Comput. Vision 106, 2 (January 2014), pp. 192-209, 2014

[4] Simone Calderara, Paolo Piccinini and Rita Cucchiara: Vision based smoke detection system using image energy and color information, Mach. Vision Appl. 22, 4 (July 2011), pp. 705-719, 2011

[5] Turgay Çelik, Hüseyin Özkaramanl and Hasan Demirel: FIRE AND SMOKE DETECTION WITHOUT SENSORS: IMAGE PROCESSING BASED APPROACH, *15th European Signal Processing Conference (EUSIPCO 2007), Poznan, Poland, September 3-7*, 2007.

[6] DongKeun Kim, Yuan-Fang Wang: Smoke Detection in Video *2009 WRI World Congress on Computer Science and Information Engineering*, ISBN: 978-0-7695-3507-4, pages 759–761. IEEE, 2009.

[7] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780, 2006.

[8] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. *In Pattern Recognition, 2004. ICPR 2004.* Proceedings of the 17th International Conference on, volume 2, pages 28–31. IEEE, 2004.

[9] OpenCV, Background Tutorial Scheme: `https://docs.opencv.org/3.1.0/Background_Subtraction_Tutorial_Scheme.png`

[10] WEST SEATTLE NEWS. *West Seattle Blog* [online]. [cit. 26.4.2018]. Dostupný na WWW: `http://zoneone.wf7woxluljv.maxcdn-edge.com/blog/wp-content/uploads/2011/07/IMG_2700.jpg`

[11] WUNC.ORG. *WUNC North Carolina radio* [online]. [cit. 26.4.2018]. Dostupný na WWW: `http://mediad.publicbroadcasting.net/p/wunc/files/styles/medium/public/201611/PartyRockFire.jpg`

[12] ALBAZ, Jenny. *Mount Laurel NJ Real Estate Blog* [online]. [cit. 26.4.2018]. Dostupný na WWW: `http://c0263062.cdn.cloudfiles.rackspacecloud.com/content/images/sized/prevent-house-fire-manassasfire2_3x2_90fcc7696d243db678977cc072239a3c_jpg_300x200_q85.jpg`

[13] THE ARGUS. *The Argus* [online]. [cit. 26.4.2018]. Dostupný na WWW: http://www.theargus.co.uk/resources/images/2719343/

[14] SUO, Steve. *The Oregonian/OregonLive* [online]. [cit. 26.4.2018]. Dostupný na WWW: http://media.oregonlive.com/wildfires/photo/treesjpg-efba77eaecc0b5efjpg-ea66d494ffd472b8.jpghttp://image.oregonlive.com/home/olive-media/width620/img/wildfires/photo/treesjpg-efba77eaecc0b5efjpg-ea66d494ffd472b8.jpg

[15] VOA. *VOA* [online]. [cit. 26.4.2018]. Dostupný na WWW: https://gdb.voanews.com/6F5D166E-8315-4AA3-AE4F-16667B4D4252_cx0_cy6_cw0_w1023_r1_s.jpg

[16] DAVID, Bařina. *Wikimedia Commons* [online]. [cit. 26.4.2018]. Dostupný na WWW: https://upload.wikimedia.org/wikipedia/commons/thumb/5/51/Denoise_-_gaussian_matrix.svg/600px-Denoise_-_gaussian_matrix.svg.png

[17] Quark67. *Wikimedia Commons* [online]. [cit. 26.4.2018]. Dostupný na WWW: https://upload.wikimedia.org/wikipedia/commons/thumb/e/e8/AdditiveColorMixiing.svg/512px-AdditiveColorMixiing.svg.png

[18] TONYLE. *Wikimedia Commons* [online]. [cit. 26.4.2018]. Dostupný na WWW: https://upload.wikimedia.org/wikipedia/commons/f/f9/YUV_UV_plane.svg

[19] SHARKD. *Wikimedia Commons* [online]. [cit. 26.4.2018]. Dostupný na WWW: https://upload.wikimedia.org/wikipedia/commons/thumb/5/5e/HSL_color_solid_sphere_munsell.png/800px-HSL_color_solid_sphere_munsell.png

[20] SHARKD. *Wikimedia Commons* [online]. [cit. 26.4.2018]. Dostupný na WWW: https://upload.wikimedia.org/wikipedia/commons/thumb/4/4e/HSV_color_solid_cylinder.png/800px-HSV_color_solid_cylinder.png