| | | | |
|---|---|---|---|
| **Aplikace počítačového vidění**<br>**Ústav automatizace a měřicí techniky**<br>FEKT VUT BRNO | Zpracovali<br>Adam Ligocki, Tomáš Lázna | | |
| | Ročník<br>1. | Obor<br>KAM | Skupina<br>M1 KAM/03 |
| Vedoucí<br>Ing. Daniel Davídek | Hodnocení | | |
| Název úlohy<br>**Segmentace textur na Raspberry Pi 2** | | | Odevzdáno |

# Zadání

Cílem projektu je implementovat alespoň dva typy algoritmů pro segmentaci textur bez předlohy na výpočetním modulu *Raspberry Pi 2* a obrazovém vstupu z kamery. Dále vytvořit databázi obrazových předloh (viz níže) čítající alespoň 20 obrazů a dohromady použít alespoň 10 typů textur. Ve výsledcích práce je nutné provést diskuzi na téma výhod a nevýhod jednotlivých segmentačních algoritmů a porovnat je s jedním dalším ne nutně implementovaným algoritmem.

# Teoretický rozbor

V počítačovém vidění je jednou ze základních úloh odseparovat sledovaný objekt od pozadí, a oddělit tak užitečné informace, které obraz nese, od irelevantních dat. Tomuto účelu slouží celá řada metod od jasové segmentace, sledování vzorů, sledování pohybu, apriorní znalost scény, atd. V rámci tohoto projektu byly implementovány a otestovány dva algoritmy, které segmentují vstupní obraz na základě textur, které se v něm vyskytují. Jeden z algoritmů pracuje na bázi fuzzy logiky a segmentuje obraz v jasové doméně barevných kanálů. Druhý algoritmus rozděluje jednotlivé textury podle odezvy kmitočtových pásmových propustí aplikovaných na vstupní data. Tato práce může v budoucnu sloužit jako pomocný nástroj při vytváření aplikací v problematice strojového vidění a zpracování obrazu.

## Gaborovy filtry

Gaborův filtr (GF) je z matematického pohledu frekvenční pásmová propust, specifikovaná vlnovou délkou, orientací a rozptylem. V dvourozměrném prostoru je Gaboruv filtr komplexní sinusoidou lineárně kombinovanou s dvourozměrnou Gaussovou křivkou. Maximální odezvu dává Gaborův filtr v okamžiku, kdy ve filtrovaném obraze vystupují stejné nebo podobné kmitočty jako v Gaborově filtru.

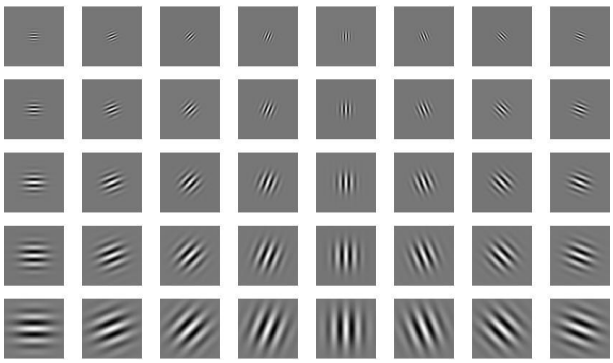
Pro spolehlivou segmentaci obrazu je potřeba filtrovat obrázek celou sadou Gaborových filtrů s různými vlnovými délkami a různou orientací.

Empiricky se pro filtraci používá *n* vlnových délek filtrů,

$$n = \text{floor}\left[\log_2\left(\frac{\sqrt{r^2 + c^2}}{2\sqrt{2}}\right)\right]$$

kde floor je funkce zaokrouhlení směrem dolů,
*r* je počet řádků,
*c* je počet sloupců vstupního obrázku.

Nejkratší vlnová délka se pak volí 22 pixelů, a každá další je dvojnásobkem předchozí. Pro každou vlnovou délku pak volíme krok rotace. Obvykle je v literatuře [1] volen krok 30 a 45 stupňů. Filtrem se pak rotuje od počátečního úhlu s krokem do úhlu 180 stupňů. Ukázka filtrů je na obrázku 1. Ukázka aplikace filtrů na reálnou fotku je na obrázku 2. Princip segmentace je poté znázorněn na obrázku 3. Po filtraci GF následuje filtrace dolní propustí a poté jsou vypočteny deskriptory z okolí pixelu. Na základě deskriptorů je obraz clusterován pomocí K-Means algoritmu.



Obrázek 1 - Ukázka sady Gaborových filtrů s různou orientaci a různou vlnovou délkou [2].

# Fuzzy C-Means

Jedná se o metodu clusterování, tedy rozdělování množiny dat do několika skupin, clusterů. Při klasickém přístupu (např. v K-Means algoritmu) každý prvek clusteru v souladu s klasickou logikou náleží nebo nenáleží. Ve Fuzzy C-Means je náležení clusteru definováno funkcí příslušnosti, může tedy nabývat libovolné hodnoty od 0 do 1. V důsledku může jeden prvek náležet k více clusterům do určité míry, suma těchto mír pro všechny clustery musí však být 1. Tato vlastnost je velmi výhodná pro některé datové množiny, například právě pro obrazy s texturami. Algoritmus je iterativní a skládá se z následujících kroků [3]:

1. Náhodně zvolíme centra clusterů.
2. Vypočítáme funkci příslušnosti pro každý prvek *i* a cluster *j*.

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{n} \left( \frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}$$

   kde *n* je počet clusterů,
   *x* jsou prvky datové množiny,
   *c* jsou centra clusterů,
   m je parametr „fuzziness“, míra fuzzy.

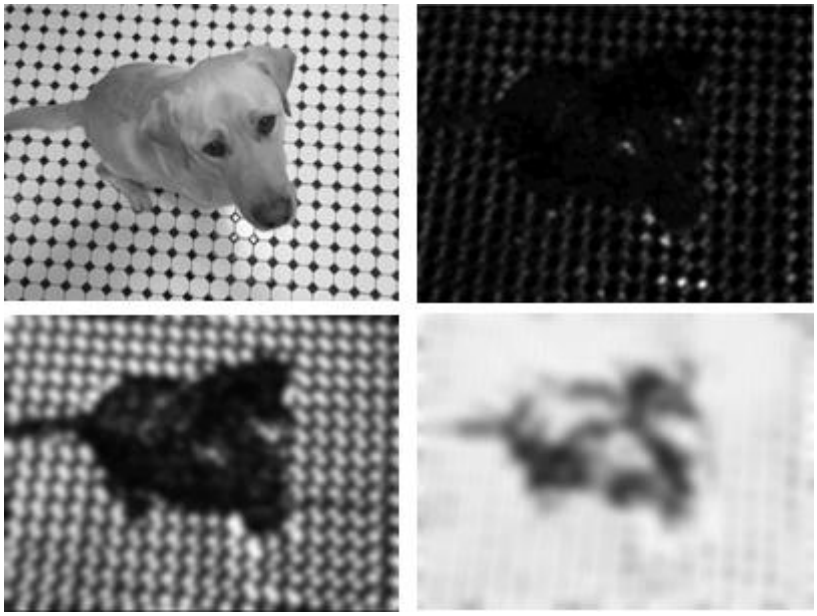
3. Vypočítáme nová centra pro každý cluster *j*.

$$c_j = \frac{\sum_{i=1}^{N} \mu_{ij}^m \cdot x_i}{\sum_{i=1}^{N} \mu_{ij}^m}$$

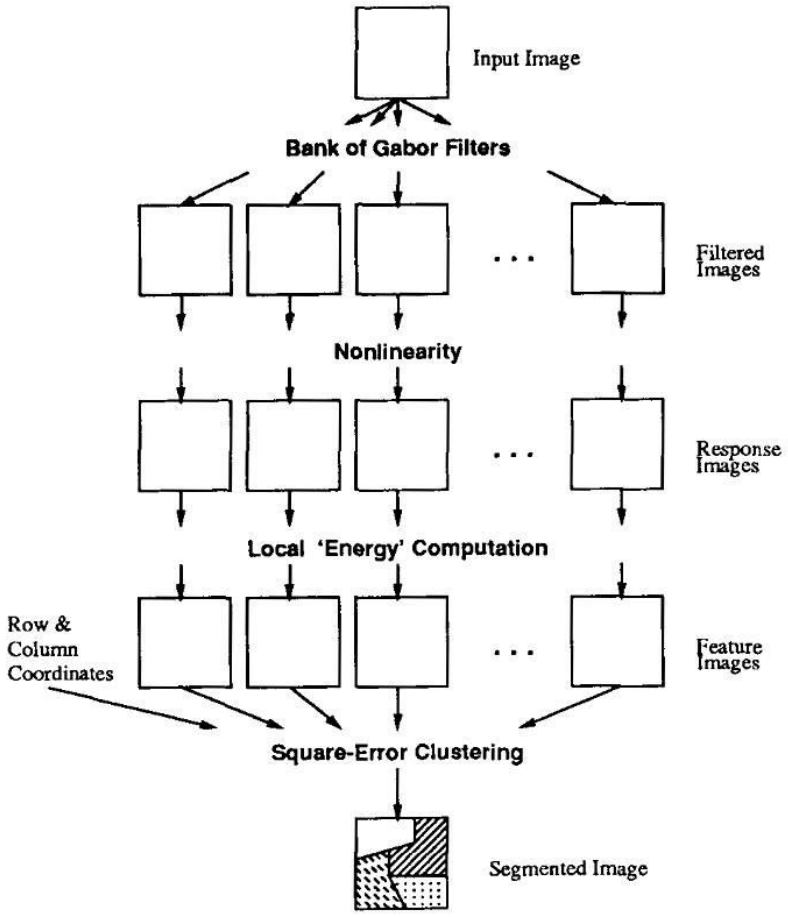
   kde *N* je počet prvků.

4. Pokud je $\left\| C^{(k+1)} - C^{(k)} \right\| < \varepsilon$
   kde *C* je vektor center clusterů,
   $\varepsilon$ je povolená chyba,

   tak skončíme, jinak pokračujeme krokem 2.



Obrázek 2 - Filtrace Gaborovym filtrem.
Vlevo nahoře vstupní obraz [4], vpravo nahoře odezva na vlnovou délku $2\sqrt{2}$ px,
vlevo dole odezva na vlnovou délku $4\sqrt{2}$ px, vpravo dole odezva na vlnovou délku $8\sqrt{2}$ px.




Obrázek 3 - Princip segmentace textur s pomocí deskriptorů vypočtených GF [1].

Použitá norma je Eukleidovská norma $L^2$. Obraz je nutné předzpracovat. Smyslem tohoto kroku je každému bodu obrazu přiřadit hodnotu, která bude vhodná pro clusterování. Obraz je defaultně v barevném modelu RGB, ovšem podle [5] je pro segmentaci textur vžhodnější použití barevného modelu CIE L*u*v. V první fázi předzpracování se tedy provede převod barevného modelu. Dalším krokem je určitý typ filtrace. Prosté jasové hodnoty vhodné nejsou z důvodu poměrně velké heterogenity pixelů v jedné textuře. Jako clusterovatelné hodnoty byl zvolen průměr a směrodatná odchylka v okně. Výpočet probíhá odděleně pro všechny tři barevné kanály, výstupem předzpracování jednoho bodu je tedy vektor o šesti hodnotách. Výstupem předzpracování celého obrazu je matice těchto vektorů. Zvolený postup je úpravou algoritmu uvedeném v [6]. Je logické, že i centra clusterů budou mít podobu šestiprvkových vektorů.

Po skončení Fuzzy C-Means algoritmu je nutné segmentaci vyhodnotit. Cílem tohoto kroku je vytvořit šedotónovou masku dle zadání. Nejprve je nutné vypočítat vektor (jeho velikost je rovna počtu clusterů $n$) jasových hodnot, které rovnoměrně pokrývají jasový rozsah 0–255. Výstupem předchozího kroku je $n$ funkcí příslušnosti. Pro každý bod masky je nutné zjistit, pro který cluster nabývá funkce příslušnosti nejvyšší hodnoty, a následně tomuto bodu přiřadit jasovou hodnotu tomuto clusteru odpovídající.

# Segmentace textur založená na spektrálních histogramech

Metoda je založena na výpočtu histogramů v okně pro každý pixel obrazu. V rámci jedné textury bude tento histogram teoreticky přibližně konstantní. Pokud takový histogram vypočítáme pro okolí pixelu na rozhraní dvou textur, můžeme jej považovat za vážený součet histogramů obou těchto textur. Za předpokladu, že známe histogramy obou textur, můžeme pomocí metody nejmenších čtverců stanovit váhy, přičemž bod bude náležet textuře s větší váhou. Při semiautonomní segmentaci je možné histogramy jednotlivých textur získat na základě manuální volby reprezentativních pixelů.

Histogramy založené na jasu mají pouze omezenou vypovídající hodnotu pro popis textury. Proto je vhodné na obraz nejprve aplikovat filtr, lze použít např. LoG nebo Gaborův filtr. Histogramy se poté vypočítají pro různé odezvy na filtry, jejich spojením dostaneme tzv. spektrální histogramy. Ty lze s obecně lepším výsledkem využít ve výše zmíněném postupu pro segmentaci. Metodu lze použít i pro autonomní segmentaci textur. Odhad reprezentativních histogramů je proveden na základě singulárního rozkladu matice, pomocí kterého je vstupní obraz transformován. Jelikož implementace tohoto algoritmu není cílem projektu, technické podrobnosti nebudou rozebírány. Podrobnější popis celé metody a zdrojové kódy pro MATLAB je možné najít v [7].

# Srovnání algoritmů

Pro implementaci byly zvoleny dva algoritmy. První převede obraz na šedotónový a aplikuje na něj Gáborovy filtry. Předzpracovaný obraz poté clusteruje pomocí K-Means algoritmu. Obraz je před zpracováním transformován na danou velikost. To jednak snižuje výpočetní náročnost (obraz je prakticky vždy zmenšován), jednak zajišťuje invarianci vůči škálování. Algoritmus není obecně invariantní vůči rotaci, závisí na volbě orientace Gáborových filtrů. Výpočetně nejvíce náročný krok je právě výpočet filtrace, samotné clusterování je poměrně rychlé.

Druhý algoritmus pracuje s barevnými složkami v modelu CIE L*u*v, v rámci předzpracování ovšem počítá pouze průměry a směrodatné odchylky. Pro clusterování využívá Fuzzy C-Means algoritmus. Obdobně je využito zmenšení obrazu před zpracováním. Algoritmus pracuje pouze s lokálním okolím

pixelů, což teoreticky zajišťuje aspoň částečnou invariantnost vůči rotaci. Předzpracování obrazu je v tomto případě nenáročné, ovšem fuzzy clusterování je výpočetně náročné.

Obecně lze říct, že první algoritmus pracuje s plošnými frekvencemi v obraze, zatímco druhý pracuje primárně s barvami. První se tak lépe hodí na segmentaci textur, které mají odlišnou strukturu, má ovšem problém rozlišit podobné textury lišící se barvou. Pro druhý algoritmus to platí přesně naopak. Společnou nevýhodou obou algoritmů je to, že jako vstup potřebují znát počet textur, resp. clusterů, na které mají být obrazová data rozdělena. Tuto nevýhodu lze potlačit tím, že v rámci předzpracování je odhadnut počet textur v obraze, s kterým následně algoritmus počítá.

Třetí algoritmus, který byl zvolen pro srovnání, využívá spektrálních histogramů a rozkladu matic. Podobně jako druhý algoritmus pracuje s lokálním okolím, má tedy podobnou míru invariance vůči rotaci. Proklamovaná výpočetní rychlost je 1 sekunda pro obraz o velikosti 300 × 300 pixelů. Jeho výhodou oproti předchozím dvěma je to, že nevyužívá žádné clusterování, které obecně vyžaduje a priori počet clusterů. Tento algoritmus je tedy zcela autonomní.

# Implementace algoritmů

Zvolené algoritmy byly implementovány v jazyce C++ s využitím knihovny OpenCV. Oba algoritmy byly zapouzdřeny do samostatných tříd s jednoduchým API. Nejzásadnější částí implementace je volba konstant, se kterými algoritmy pracují.

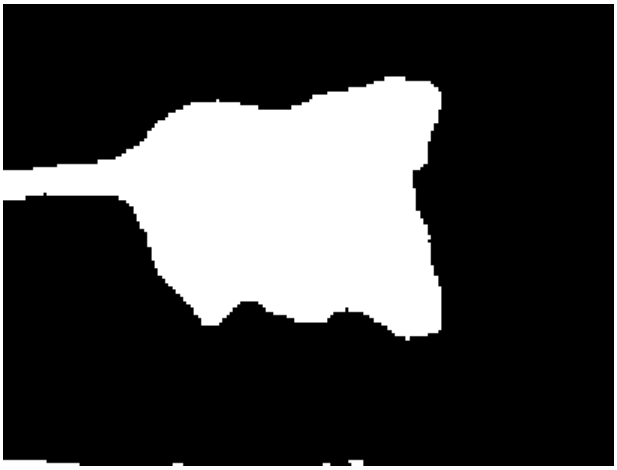
Demonstrační program, který je popsaný v jedné z následujících kapitol je zkompilován a běží na platformě minipočítače Raspberry PI. Jedná se o velmi populární platformu s velkou komunitou vývojářů a s velkým zázemím v rámci podpory ze strany tvůrců softwaru a hardwaru. Konkrétně demonstrační program běží na linuxové distribuci Raspbian, která je derivátam distribuce Debian. Pro snímání živého obrazu je použit oficiální kamerový modul prodávaný k Raspberry PI.

## Volba parametrů pro Fuzzy C-Means

Pro odladění parametrů algoritmu byla použita fotka psa z obrázku 2. Jedná se o obraz, který není úplně triviální na zpracování, a to jednak kvůli nerovnoměrnému osvětlení, jednak kvůli vysoké míře heterogenity pozadí. Experimentálně byly jako nejvhodnější zjištěny tyto parametry:

- velikost okna 11,
- fuzziness 2,
- epsilon 5,
- rozměr obrazu 128.

Poslední jmenovaný parametr zaslouží vysvětlení. Obraz je zmenšen tak, aby jeho kratší strana měla zadanou délku. Důvodem tohoto kroku je zmenšení výpočetní náročnosti clusterování. Experimenty ukázaly, že při zmenšení obrazu je dosaženo kvalitativně stejných výsledků za významně kratší dobu. Pro výpočet hodnoty pixelů ve zmenšeném obraze je využitu bilineární interpolace. Výsledek segmentace s výše uvedenými parametry je v podobě masky na obrázku 4. Je nutné poznamenat, že výsledek segmentace se při každém průchodu algoritmem liší z důvodu náhodného inicializování center clusterů, které je odvozeno od aktuálního času.



Obrázek 4 - Výstup algoritmu s Fuzzy C-Means algoritmem.

# Volba parametrů pro segmentaci Gaborovými filtry

S ohledem na implementaci algoritmu na procesoru ARM je nezbytné redukovat výpočetní náročnost na minimum. Proto se vstupní obraz vždy zmenšuje na maximálně 400 × 300 px. Podle výše zmíněného postupu bylo vypočteno *n* pro počet vlnových délek 7. Krok rotace byl zvolen 30 stupňů. To dává ve výsledku 42 filtrů. Při optimalizaci výpočtů však bylo zjištěno, že největší vlnové délky mají jen velmi malý přínos pro zpřesnění segmentace, přičemž využívají největší část výpočetního výkonu. V rámci projektu je tak hranice *n* stanovená na 5. To dává ve výsledku maximálně 30 konvolučních filtrů.

Po provedení segmentace jsou rozměry výstupní mapy textur nastaveny na hodnoty rozměrů vstupního obrázku. Výstup algoritmu je na obrázku 5.

Obrázek 5 - Výstup algoritmu s Gaborovymi filtry.

# Estimátor počtu textur v obraze

Zvolili jsme přístup založený na barvách. Textura má obvykle jednu dominantní barvu, např. tráva zelenou. Pokud vypočítáme histogramy jednotlivých barevných složek obrazu, lze předpokládat, že textury v něm budou tvořit výrazné peaky. Stejně jako při samotné segmentaci, i zde se osvědčil barevný model CIE L*u*v více než klasický RGB. V L*u*v modelu jsou pouze dvě barevné složky, stačí tedy vypočítat dva histogramy, pro kanál u a pro kanál v. Histogramy jsou filtrovány konvolučním

průměrovacím filtrem o délce 7, což vede k jejich vyhlazení. To je výhodné především pro „rozkmitané“ histogramy.

Odhad počtu textur se získá tak, že se sečtou peaky v obou histogramech, jejichž výška je alespoň třetina maximální hodnoty daného histogramu a jejich vzájemná vzdálenost je minimálně 15 binů. Tyto konstanty byly získány empiricky. Na obrazové předlohy ve vytvořené databázi tato metoda funguje s průměrnou chybou odhadu asi 17 %.

# Popis API

Software je napsaný v objektově orientovaném jazyve C++. Toho bylo využito při tvorbě rozhraní a programátor při vytváření vlastní aplikace, ve které používá naší implementaci algoritmů, interaguje pouze se zapouzdřovacími třídami GaborTextureSegmenter a FuzzySegmenter konkrétně pak s public metodami.

```
Mat GaborTextureSegmenter::segmentImageByNumberOfTextures(Mat img, int
numberOfTextures);

Mat FuzzySegmenter::SegmentImage(Mat I, int clusterCount);
```

Vstupními parametry obou metod je obrazová matice knihovny OpenCV a počet clustrů do kterých požadujeme obrázek segmentovat a výstupem funkce je nově vzniklý nesegmentovaný obraz.

# Demonstrační program, popis funkce main()

Pro demonstrační účely byl vytvořen program, který kromě načítání obrázků z databáze uložené v souborovém systému počítače načítá také obraz z kamery a následně jej segmentuje. Program pracuje v několika režimech, které se volí jeho jednotlivými vstupními parametry.

Prvním vstupním parametrem programu je --camera pro jednorázovou segmentaci kamerového vstupu, --continous pro kontinuální snímání a segmentaci vstupu kamery, nebo zadání cesty k obrázku v rámci souborového sytému.

Druhým parametrem je --fuzzy pro filtraci v barevné doméně a --gabor pro filtraci ve frekvenční doméně.

Třetím parametrem je počet textur, na který požadujeme obrázek segmentovat. V případě absence tohoto parametru program provede odhad počtu textur výše zmíněným estimátorem.

Demonstrační program také přijímá argument --help nebo -h po kterém vypíše stručnou nápovědu.

Příklad volání programu:

```
./TextSeg --camera --gabor 4

./TextSeg ~/images/myImage.jpg --fuzzy 3

./TextSeg --continous --fuzzy
```

# Závěr

Je možné konstatovat, že zadání projektu bylo splněno. Prvním implementovaným algoritmem byla kombinace Gaborových filtrů a K-Means algoritmu (dále Gaborova metoda). Druhým algoritmem je poté aplikace Fuzzy C-Means algoritmu na průměry a směrodatné odchylky jednotlivých barevných kanálů (dále Fuzzy metoda). Základní rozdíl ve výsledcích poskytovaných oběma algoritmy je patrný z obrázku 6. Gaborova metoda nerozliší dvě stejně strukturované textury o různé barvě, zatímco Fuzzy metoda nerozliší dvě stejně barevné textury.

Srovnání výstupu obou algoritmů na vybraných předlohách z vytvořené databáze je na obrázku 7. Je patrné, že Fuzzy metoda poskytuje celkově lepší výsledky. Je to dáno mj. tím, že v obrazových předlohách se textury barevně liší. Co se výpočetní náročnosti týče, rychlejší je Gaborova metoda. Navzdory pomalejšímu předzpracování Gaborovými filtry zabere celá segmentace přibližně 5 sekund. U Fuzzy metody je doba výpočtu závislá na náhodné volbě počátečních center clusterů, ale obvykle trvá 7 až 8 sekund, kvůli značné výpočetní náročnosti Fuzzy C-Means. Je zřejmé, že výsledek obou algoritmů je špatný v případě nesprávného odhadu počtu textur v obraze. I z toho důvodu je možné tento počet zadat manuálně.

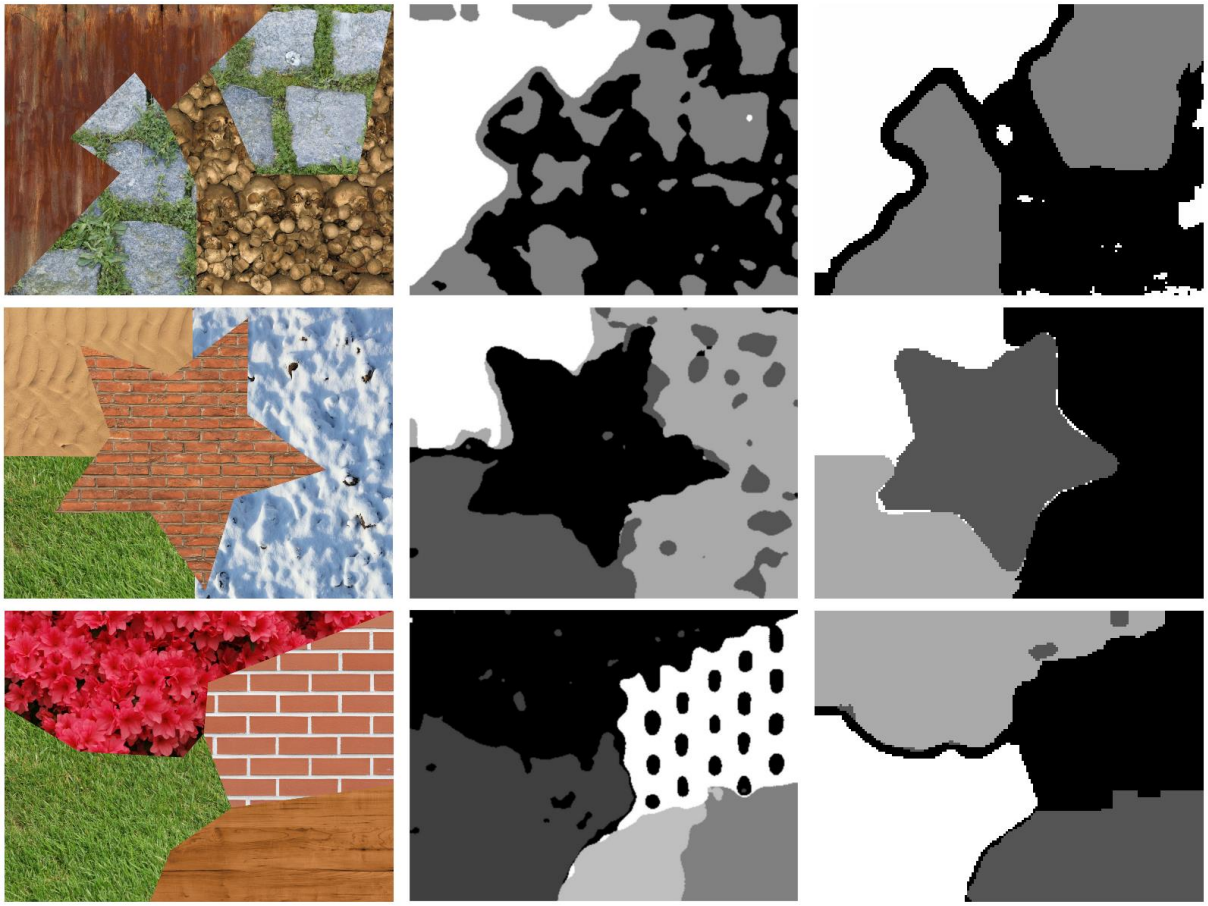
Obrázek 6 – Srovnání Gaborovy metody (uprostřed) a Fuzzy metody (vpravo) [9].

Velmi patrnou chybou jsou malé regiony jedné textury uprostřed jiné oblasti. To by teoreticky bylo možné odstranit pomocí morfologické operace uzavření nad jednotlivými složkami výsledné masky. Bohužel implementace této funkce se v rámci projektu nestihla.

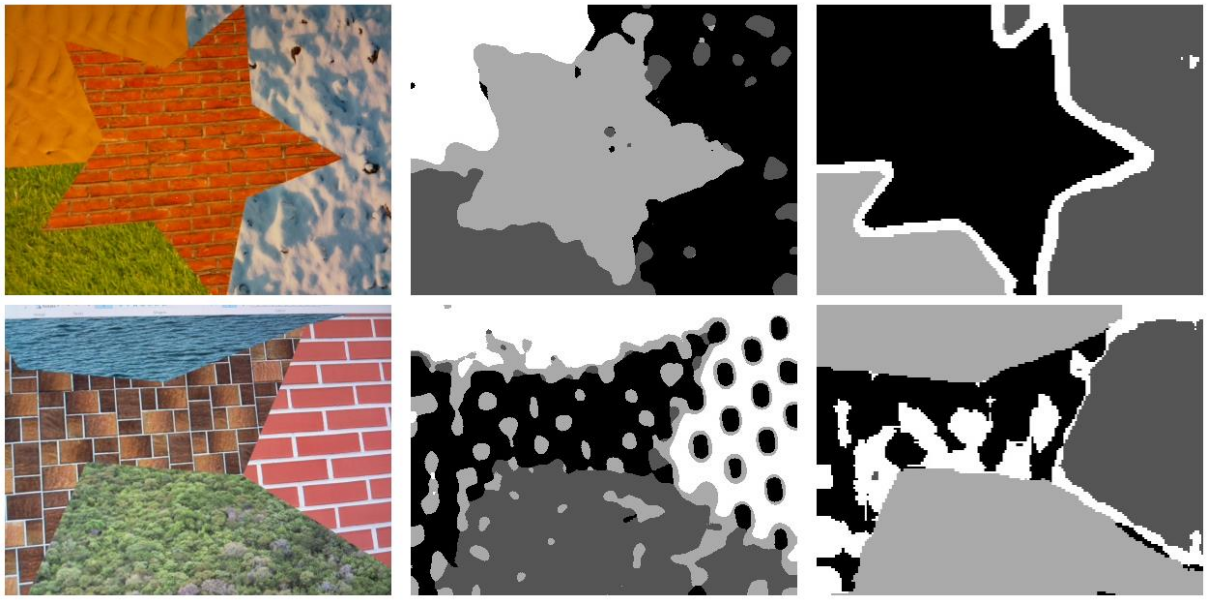
Výstupy segmentace při použití obrazu z kamery jsou na obrázku 8. Dosažené výsledky jsou dle očekávání horší než na obrazových předlohách načtených ze souboru. Na vině je nižší kvalita optiky, nerovnoměrné osvětlení, apod. Přesto určitou míru úspěšnosti stále mají.

Celkově výsledky projektu považujeme za uspokojivé, vzhledem k tomu, že segmentace textur nepatří mezi triviální úlohy počítačového vidění. V rámci pokračování projektu by bylo vhodné spojit oba algoritmy do jednoho a využít tak všech výhod, které nabízejí.

Zdrojem textur pro databázi obrazových předloh byl [8].



Obrázek 7 - Porovnání segmentačních algoritmů na obrazech ze souboru.
Vpravo originál, uprostřed Gaborova metoda, vlevo Fuzzy metoda.



Obrázek 8 - Porovnání segmentačních algoritmů na obrazech z kamery
Vpravo originál, uprostřed Gaborova metoda, vlevo Fuzzy metoda.

# Literatura


[1] Anil K. Jain a Farshid Farrokhnia. *Unsupervised Texture Segmentation Using Gabor Filters* [online]. Department of Computer Science Michigan State University, 1990 [cit. 2016-05-03]. Dostupné z: http://www.ee.columbia.edu/~sfchang/course/dip/handout/jain-texture.pdf

[4] *Mathworks Documentation* [online]. [cit. 2016-05-03]. Dostupné z: http://www.mathworks.com/help/examples/images/TextureSegmentationUsingGaborFiltersExample_01.png

[2] *Stackexchange* [online]. [cit. 2016-05-03]. Dostupné z: http://dsp.stackexchange.com/questions/25040/whats-the-optimal-filter-size-for-a-2d-gabor-filter

[3] Fuzzy c-means clustering algorithm. *Data Clustering Algorithms* [online]. [cit. 2016-04-22]. Dostupné z: https://sites.google.com/site/dataclusteringalgorithms/fuzzy-c-means-clustering-algorithm

[5] DESHMUKH, Kanchan S. Texture Image Segmentation using FCM. In: *Proceedings of 2012 4th International Conference on Machine Learning and Computing* [online]. Singapore: IACSIT Press, 2012 [cit. 2016-04-24]. Dostupné z: http://www.icmlc.org/icmlc2012/025_icmlc2012.pdf

[6] SAYADI, Mounir, Lotfi TLIG a Farhat FNAIECH. A New Texture Segmentation Method Based on the Fuzzy C-Mean Algorithm and Statistical Features. In: *Applied Mathematical Sciences* [online]. 2007, s. 2999-3007 [cit. 2016-04-24]. Dostupné z: http://www.m-hikari.com/ams/ams-password-2007/ams-password57-60-2007/sayadiAMS57-60-2007.pdf

[7] YUAN, Jiangye. *Factorization-Based Texture Segmentation* [online]. [cit. 2016-05-02]. Dostupné z: https://sites.google.com/site/factorizationsegmentation/

[8] *Textures.com* [online]. 2016 [cit. 2016-04-24]. Dostupné z: https://www.textures.com/

[9] *Cargo Gallery* [online]. 2016 [cit. 2016-05-03]. Dostupné z: http://payload276.cargocollective.com/1/15/509033/7830335/mosaic.jpg