Projekt do předmětu MAPV

Detekce významných bodů na Raspberry Pi 2

Vedoucí Ing. Daniel Davídek

Autoři Bc. Pavel Škoda

Bc. David Pařík

Zadání

Cílem projektu je implementovat algoritmy pro detekci významných bodů, jejich popis a korespondenci ve dvou obrazech na výpočetním modulu *Raspberry Pi 2* s využitím alespoň dvou algoritmů. Ve výsledcích práce je nutné provést diskuzi na téma výhod a nevýhod jednotlivých algoritmů a porovnat je s jedním dalším ne nutně implementovaným algoritmem. Diskuze typu a vlastností scény a charakteru předlohy vhodné pro robustní detekci a korespondenci pomocí jednotlivých algoritmy s návazností na výstupy realizovaných algoritmů (ne/invariantnost vůči geometrickým a jasovým transformacím).

Software:

- V rámci linuxové distribuce Raspbian rozchoďte knihovny pro zpracování obrazu OpenCV 3.X [1]
- Jako programovací jazyk využijte C/C++ nebo Python

Vstupy:

- Osvětlovací soustava a snímací aparatura dle vlastního návrhu
- Raspberry Pi 2 s kamerovým modulem

Výstupy:

- Alespoň dva algoritmy pro korespondenci významných bodů obrazu z kamery a předlohy
- Snaha o nízkou výpočetní náročnost - jednotky FPS
- Periodicky vykreslovaný obrazový výstup z kamery rozšířený o obraz předlohy
    - minimum: zobrazené významné body v obou obrazech propojené v složeném obrazu čarou
    - vykreslení perspektivně zkresleného bounding-boxu předlohy do obrazu z kamery

Metody detekce významných bodů

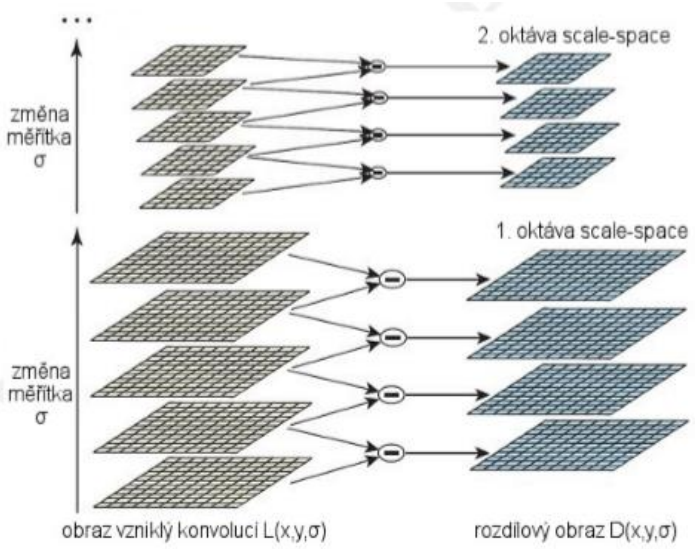
SIFT [1],[4]

Metodu SIFT(Scale-Invariant Feature Transform) můžeme rozdělit do čtyř základních kroků:

Detekce extrémů uvnitř scale-space

Během tohoto kroku je obraz převáděn do různých měřítek. Toho se dosahuje opakovaným vyhlazováním obrazu pomocí Gaussovy funkce, která se chová jako dolní propust. Opakovaným filtrováním obrazu získáváme jednotlivé vrstvy stejného rozměru, které tvoří jednu oktávu scale-space. V ní má nejvyšší vrstva dvojnásobné měřítko oproti nejnižší a je základem pro následující oktávu, která vznikne podvzorkováním na poloviční rozměr.

Dále jsou odečtením jednotlivých vrstev v oktávách získané rozdílové obrazy, ve kterých se prozkoumává 26-ti okolí každého bodu obrazu. Tímto získáváme jednotlivé významné body.



Zpřesnění polohy významných bodů

Zde dochází k přesnému určení polohy dříve nalezených významných bodů, které jsou popsány souřadnicí a měřítkem. Dále jsou vypuštěny body s nízkým kontrastem nebo body nalézající se v blízkosti hran.

Rozhodnutí o orientaci

Významným bodům je přiřazena dominantní orientace, aby byly invariantní vůči rotaci.

Orientace je určena z výpočtu gradientů v různých směrech. Z nich je dále určen dominantní směr.

Sestavení deskriptoru významných bodů

Deskriptory popisují okolí jednotlivých významných bodů tak, aby byl tento popis nezávislý na geometrických a jasových transformacích. Sestavení deskriptorů probíhá z gradientů v okolí významného bodu. Dá se tedy využít předešlého kroku. Prostor okolí významného bodu je rozdělen na 8 stejných polí. V každém z nich je poté hledána dominantní orientace gradientů.

SURF [1],[4]

Metoda SURF byla navržena tak, aby se co možná nejvíce snížila její výpočetní náročnost. Čerpá z principů používaných v SIFT. Zrychlení bylo dosaženo v oblasti sestavení scale-space a redukci výsledných deskriptorů. SIFT scale-space aproximuje s využitím rozdílů Gaussových funkcí, ty mají ovšem velké odezvy v okolí hran, které je potřeba dodatečně odstranit. Surf tyto dva kroky slučuje, když významné body detekuje pomocí determinantu Hessianovy matice. Pro rychlý výpočet determinantu je využit integrální obraz. Ten je vybudován ze vstupního obrazu podle vztahu:

$$I_{\sum}(\boldsymbol{x}) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i,j)$$

Tento obraz pak umožňuje v libovolně velkém obdélníkovém prostoru spočítat sumu všech pixelů pouhým sečtením čtyř čísel.

Hessianova matice bodu (x,y) v daném měřítku σ je definována:

$$H(x,y,\sigma) = \begin{bmatrix} L_{xx}(x,y,\sigma) & L_{xy}(x,y,\sigma) \\ L_{yx}(x,y,\sigma) & L_{yy}(x,y,\sigma) \end{bmatrix}$$

L_{xx} (x,y,σ) je zde hodnota druhé parciální derivace podle x konvoluce vstupního obrazu I(x,y) s Gaussovou funkcí G(x,y,σ) s měřítkem σ v bodě (x,y).

Po vypočtení determinantu jsou významné body hledány obdobně jako u metody SIFT pomocí 26-ti okolí bodu.

Tvorba deskriptoru

Kvůli nezávislosti deskriptoru na rotaci je každému významnému bodu přiřazena orientace. Ta je spočítána pomocí konvoluce, aby se využilo integrálního obrazu. Jako konvoluční masky se použijí Haarovy vlny ve směru x a y.

Kolem významného bodu je vytvožena čtvercová oblast natočená podle dominantní orientace. Ta je dále rozdělena na 4x4 podoblasti. V každé z nich je určeno pět pravidelně rozmístěných bodů, pro které je vypočtena odezva na Haarovou vlnku ve směru osy x a y. Odezvny jsou značeny d_x a d_y a v každé z podoblastí je vypočtena suma d_x a d_y. Pro zvýšení odolnosti na změnu osvětlení jsou spočteny i sumy $|d_x|$ a $|d_y|$. Tyto čtyři hodnoty pak tvoří vektory pro každou z 16-ti oblastí a všechny tvoří výsledný deskriptor.

ORB [2],[3]

ORB (Oriented FAST and Rotated BRIEF) je alternativou SIFT a SURF vytvořenou týmem pracujícím na OpenCV. Vznikl spojením detektoru FAST a modifikovaného deskriptoru BRIEF. Jeho hlavní výhodou je menší výpočetní náročnost a bezplatnost jeho využívání, přičemž dosahuje obdobných výsledků.

Nalezení významných bodů

Nejprve je k nalezení významných bodů použit detektor FAST. Dále je mezi těmito body nalezeno několik nejlepších s využitím Harrisova detekrotu rohů, kterým se vyloučí nalezené hrany.

Tento algoritmus také využívá scale-space k zajištění nezávislosti na měřítku.

Určení orientace

K určení orientace nalezeného bodu vypočítává těžiště intenzity rohu, jehož výsledkem je pouze jeden dominantní výsledek (na rozdíl od operátoru orientace u SIFT). Moment políčka obrázku s rohem se vypočte jako:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y)$$

Pomocí těchto momentů se vypočte těžiště:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)$$

Poté můžeme určit vektor ze středu rohu O do těžiště C:

$$\theta = \operatorname{atan2}(m_{01}, m_{10}).$$

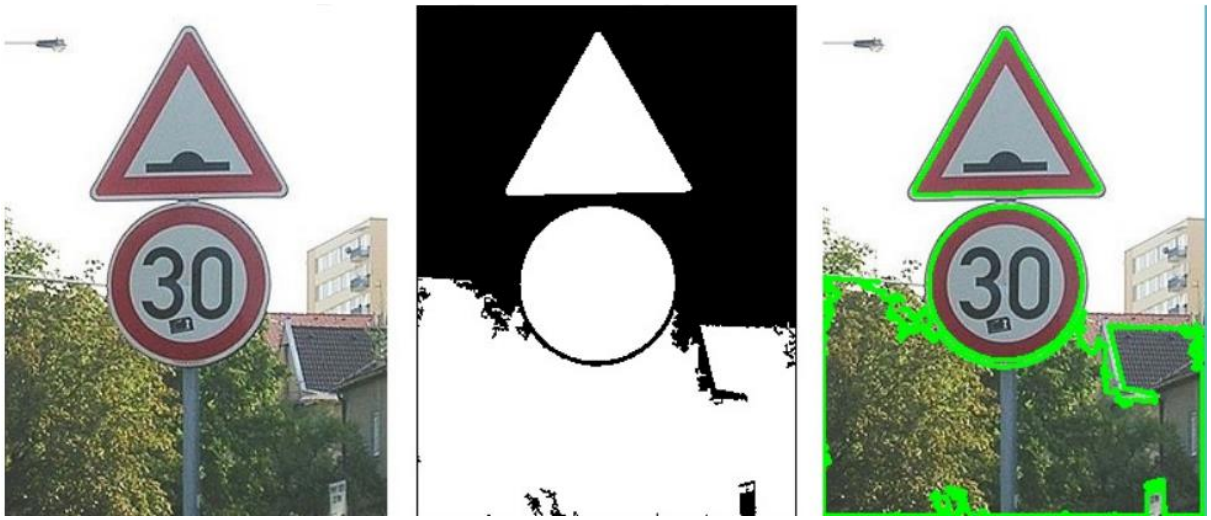
Popsání významných bodů

K popsání významného bodu využívá deskriptor BRIEF, který ovšem funguje špatně při rotaci. ORB tedy používá BRIEF tak, že jej natáčí podle orientace klíčových bodů. Účinnost tohoto deskriptoru je obdobná jako u SIFT, včetně jeho odolnosti vůči změně osvětlení, rozmazání a pokřivené perspektivy. Jedná se o binární deskriptor složený z množiny testů porovnávání dvou pixelů. Zkoumaný obraz je nejdříve vyhlazen kvůli zamezení vlivu šumu.

MSER [1]

Metodu MSER (*Maximally stable extremal regions – Maximálně stabilní extrémní oblasti*) lze využít stejně jako tři předchozí metody k popisu obrazu. Na rozdíl od předchozích metod však MSER neprovádí detekci a popis významných bodů, ale významných oblastí. Tyto oblasti musejí splňovat několik základních vlastností:

- Invariance vůči jasovým transformacím
- Sousednost zůstává zachována i po lineární transformaci obrazu
- Stabilita – extrémní oblasti obrazu s životností v širokém rozsahu prahů
- Detekovatelnost v různých měřítkách obrazu
- Množinu extrémních oblastí lze vyčíslit na O(*n* log log *n*), kde *n* je počet pixelů obrazu



Obrázek 1 - Využití MSER v obraze. Vstupní obraz (vlevo), prahovaný obraz (uprostřed), vykreslení hranice nalezených objektů (vpravo)

Princip detekce MSERů může být vysvětlen následovně. Šedotónový obraz *I* je postupně prahován všemi možnými prahy. Pixely s intenzitou pod prahovou hodnotou jsou zabarveny černě a pixely nad či rovné hodnotě prahu jsou bílé. Jestliže jsou jednotlivé naprahované obrazy I_t, kde *t* odpovídá hodnotě prahu, použity jako video snímky, bude taková video stopa začínat bílým obrazem, ve kterém se postupně začnou objevovat černé skvrny, jež indikují lokální minima jasové funkce obrazu *I*. V určitém bodě projekce dojde k postupnému slučování jednotlivých černých skvrn, až projekce skončí zcela černým obrazem. Veškeré uzavřené komponenty, které se během takovéto projekce objeví, jsou maximálními regiony. Pokud by bylo cílem detekovat regiony minimální, je nutné celý proces opakovat s inverzní jasovou funkcí obrazu *I*.

Snímací aparatura a osvětlení scény

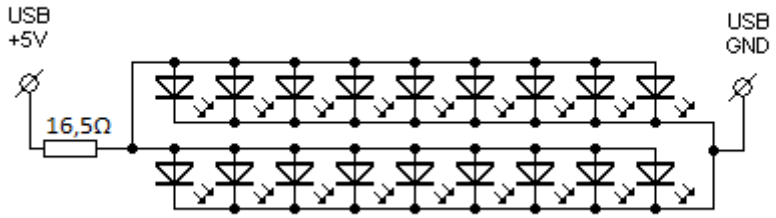
Jako snímací aparaturu jsme využili webkameru Logitech připevněnou na stativu, který má tři tvarovatelné nohy vhodné pro případné připevnění na nerovné místo pro přesné snímání požadované scény.

Kamera se k výpočetnímu modulu Raspberry Pi 2 připojuje pomocí USB 2. Mezi její vlastnosti patří:

- Rozlišení až 1280x1024
- Manuální otření pomocí kolečka na boku kamery
- Manuální tlačítko pro sejmutí obrazu – není využito
- Rozsah nastavení jasové složky 0 - 255
- Rozsah nastavení kontrastu 0 - 255

Pro osvětlení scény jsem vyrobil lampičku, která se k Raspberry PI 2 připojuje stejně jako webkamera – přes konektor USB. Díky tomu má zajištěný stabilní napájecí zdroj o velikosti 5V a maximálně 0,5A.

USB lampička se skládá z 2 krát devíti bílých LED diod získaných ze starých svítilen. Ty jsou zapojeny sérioparalelně a následně přes rezistor o velikosti 16,5Ω připojeny k napájecím kabelům USB konektoru. Díky rezistoru se snížil proud protékající led diodami na přibližně 80mA, přičemž jas klesl pouze minimálně. Z důvodu špatně zvoleného materiálu pro tvarovací drát do nohy světýlka (ocelový drát) a jeho délce, je nejlepší pozice, ve které je přísvit stabilní tzv. „obranné postavení kobry“, kdy je za USB konektorem noha lampičky zatočená do kruhu, který se opírá o podložku (při připojení k Raspberry PI2 o stůl) a následně vztyčená vzhůru. Díky tomuto nastavení má přísvit vlastní podporu a zbytečně nezatěžkává USB konektor.



Obrázek 2 - Schéma zapojení USB Led přísvitu

Provedení projektu

Seznámení a nastavení

Prvním úkolem bylo seznámení se s výpočetním modulem Raspberry Pi 2 obsahujícím systém Raspbian, protože nikdo z nás s ním, ani s operačním systémem založeném na Linuxu nepracoval. Po prvotním zjištění jak tento malý počítač funguje, bylo naším úkolem zprovoznit vzdálený přístup přes síť pro jednodušší práci a nainstalovat do něj knihovny OpenCV, abychom mohli využívat jejich funkce pro splnění zadání projektu.

Pro instalaci jsme zvolili nejnovější verzi OpenCV 3.1.0 s tím, že jsme instalaci prováděli podle manuálu na webových stránkách (adresa 1). Jediný rozdíl oproti tomuto postupu byl v tom, že jsme pro naši práci potřebovali navíc přidat ještě externí knihovny obsahující podstatné funkce, hlavně pak knihovnu xfeatures2d (adresa 2). Nejvíce času zde zabral překlad knihoven pro jejich následné použití, který trval přibližně 3 a ¾ hodiny a kvůli chybějícím externím knihovnám při prvním překladu jsme jej museli provádět vícekrát. Poté bylo nutné provést podobný postup i na PC, na kterém jsme programovali zdrojové kódy (adresa 3).

Programování zdrojových kódů probíhalo v operačním systému Microsoft Windows 10, v prostředí Microsoft Visual studio, kde jsme vždy nejprve naprogramovali, otestovali a odladili část kódu, následně ji přenesli do Raspberry Pi 2 a zkomprimovali pomocí programu Cmake. Výsledkem této procedury je kód, který lze spustit jak v systému Windows, tak v Linuxu. Jediný rozdíl, který je při přenosu kódu nutné kompenzovat je výpočet času a fps, kdy funkce *clock()* vrací v každém systému jinak dlouhý řetězec. V případě spuštění kódu na Raspberry Pi je nutné zařadit do výpočtu navíc dělení 1000, aby se výsledná hodnota zobrazovala ve správných jednotkách.

Popis kódu

Samotný kód je rozdělen na několik samostatných funkcí, které můžou a nemusí být ve výsledném kódu zapsány. První z nich je funkce *obshulaKlavesnice*, která slouží k výběru vyhledávaného obrazu (klávesa „t“), změně metody, která bude zpracovávat snímky (klávesa „m“) a ukončení programu (klávesa „q“). Další je *obsluhaMysi* sloužící pro výběr části obrazu při stisku klávesy „t“. Třetí funkcí je *wait,* ta je využívána po inicializaci kamery, kdy se občas stávalo, že byl program příliš rychlý a kamera nestíhala načíst první snímek. Poslední funkcí je *identifikace* sloužící jak už název napovídá k identifikaci základních vlastností aktuálně připojené kamery.

Za těmito samostatnými funkcemi následuje samotné tělo programu provádějící detekci významných bodů. Pro náš projekt jsme si vybrali metody detekce významných bodů SURF a ORB. V první části je proto nastaveno, aby se program uživatele zeptal, kterou metodu detekce významných bodů chce využívat a následně, jestli chce nastavit vlastnosti kamery jako je jas, kontrast a rozlišení.

Program následně pomocí zvolené metody provede detekci významných bodů na předloze, jejich popis uložení pozice rohů obrazu pro pozdější ohraničení nalezené oblasti. Dále

následuje nekonečná smyčka, ve které se nejprve načte obraz z připojené USB kamery, tento obraz se převede do šedotónového obrazu a následně se provádí vyhledávání bodů a jejich popis.

Jakmile jsou nalezeny a popsány významné body, dojde k jejich porovnání s významnými body předlohy, díky čemuž dostaneme pouze body, které mají na druhém obraze svého dvojníka.

Pro zvýšení přesnosti se z těchto bodů vyberou ještě ty, jejichž vzdálenost se příliš neliší (správně nalezené body by měly mít vzdálenosti přibližně stejně dlouhé). Spojnice těchto bodů se následně vykreslí do okna.

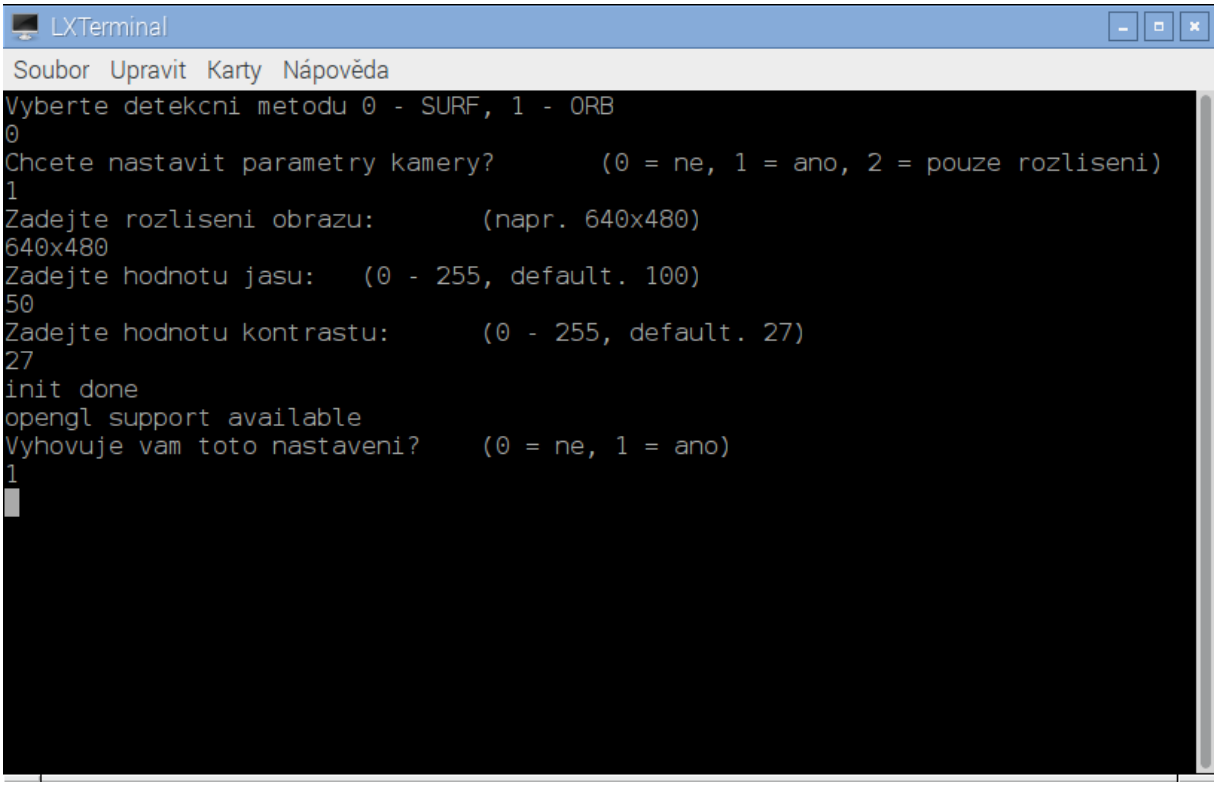
Následuje nalezení okraje objektu ve snímané scéně a provedení transformace perspektivy pomocí rohových bodů hledaného obrazu. Pokud je nalezený počet dobrých významných bodů dostatečný, dojde k vykreslení okrajů objektu do obrazu pomocí čtyř zelených čar.

Poslední částí programu je výpočet doby, kterou program potřeboval pro nalezení a vykreslení zadaného obrázku ve snímané scéně. Tato zaznamenaná doba se následně vypíše do výstupního okna jak ve formě času v sekundách tak ve formě fps (snímků za sekundu).

Použití programu

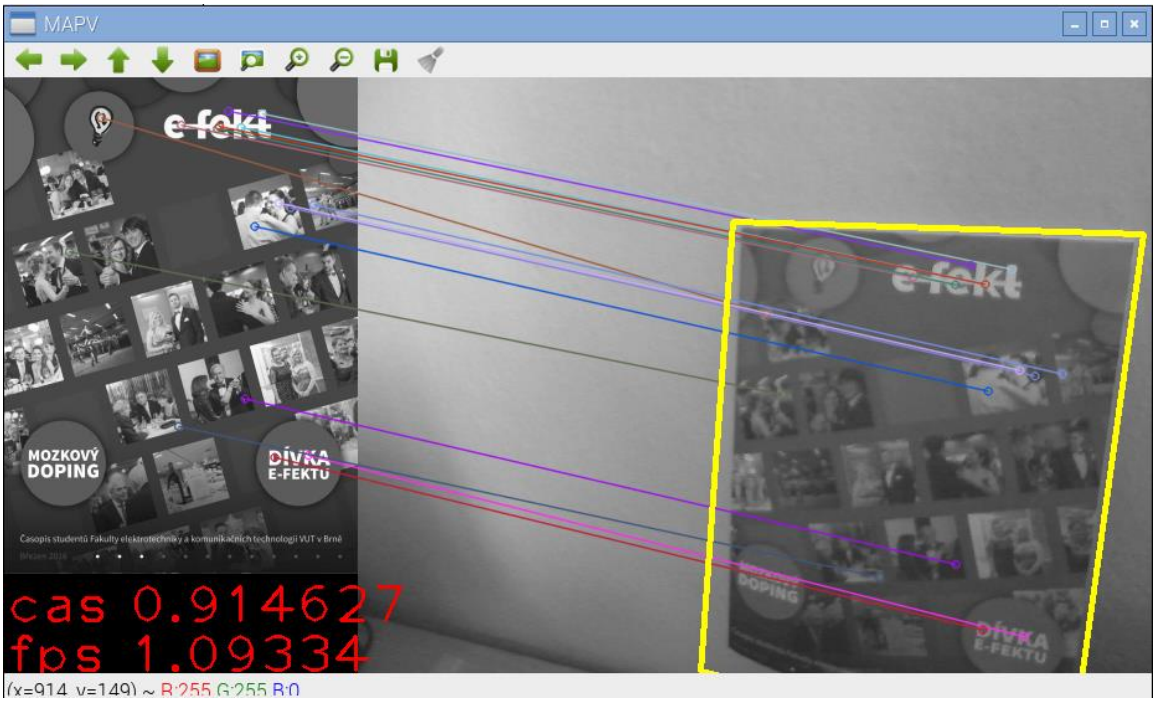
Program je nutné spouštět buďto v terminálu, nebo poklikáním na spouštěcí ikonu a výběrem spuštění programu v terminálu. Je to z důvodu výpisu informací do terminálového okna. Po spuštění se program uživatele dotáže, kterou z dvou metod (SURF, SIFT) chce využít pro detekci významných bodů. Vybírání je řešeno zápisem čísla, u něhož je napsaná zvolená možnost. Po zvolení metody detekce se objeví nastavení parametrů kamery, kdy má možnost toto nastavení přeskočit, nastavit pouze rozlišení kamery (při nastavení nepodporovaného rozlišení se kamera přepne do rozlišení, které mu je nejblíže svým rozsahem), nebo nastavit rozlišení, jas a kontrast (u jasu a kontrastu jsou zobrazeny hraniční hodnoty). V případě volby nastavení všech možností se po dokončení nastavení zobrazí ukázkové okno s aktuálně nastavenými hodnotami a uživatel je dotázán, zda je to to co žádal, nebo zda chce svou volbu změnit. Při potvrzení se spustí hlavní proces programu a spouští se detekce základního obrazu ve scéně.

V průběhu detekce má uživatel možnost pomocí klávesy „m“ měnit metodu detekce významných bodů. Pomocí klávesy „t“ se uživatel dostane do okna, v němž si pomocí kurzoru myši vybere oblast, která bude následně vyhledávána a pomocí klávesy „q“ lze program ukončit.



Obrázek 3 - ukázka nastavení hodnot připojené kamery

Pro využití této detekce objektu ve scéně je důležité, aby byla scéna vhodně nasvícena (přídavný osvětlovací USB modul). Dalším požadavkem na snímanou scénu je co nejvíce omezit rušivé vlivy, tedy aby byl hledaný objekt na pokud možno jednobarevném pozadí bez přebytečných barevných přechodů.



Obrázek 4 - Příklad detekce časopisu e-fekt metodou SURF

Závěr

Cílem našeho projektu bylo implementovat algoritmy pro detekci významných bodů, jejich popis a korespondenci ve dvou obrazech na výpočetním modulu Raspberry Pi 2 s využitím alespoň dvou algoritmů. Jako první algoritmus jsme zvolili SURF (Speed-Up Robust Features), který vychází z pomalejší metody SIFT a očekávali jsme tedy, že bude vhodné jej využít pro detekci významných bodů ve videu. Po dokončení implementace metody SURF do zdrojového kódů byla nejvyšší naměřená rychlost detekce kolem jedné sekundy a tedy i kolem jednoho fps. Největší výhodou této metody jsou její dobré detekční a popisové vlastnosti a stabilita, kdy při nalezení zadaného obrazu se množství nalezených bodů ustálilo na nějaké hodnotě a už se příliš neměnilo.

Druhou zvolenou metodou je volně dostupný ORB, který je vyvíjený přímo vývojáři OpenCV. Jeho největší výhoda spočívá v rychlosti, kdy se v závislosti na počtu vykreslovaných spojnic pohybujeme až okolo 0,17 sekund na snímek (až 6fps). Trpí však jedním problémem, který se nám nepodařilo přes všechnu snahu nijak vhodně eliminovat a to postupným snižováním počtu vykreslovaných bodů až zmizí úplně. Tento problém spočívá v postupném snižování minimální vzdálenosti mezi nalezenými body a tím dochází k eliminaci více a více spojnic. Jako nejjednodušší řešení tohoto problému jsme zvolili vyresetování minimální vzdálenosti spojnic, pokud dojde ke snížení počtu spojnic pod určitou mez. Z toho důvodu je občas při detekci vidět náhlý nárůst spojnic z několika pár na několik desítek.

Jedním ze způsobů jak nejvíce ovlivnit výpočetní rychlost programu je změna rozlišení kamery, kdy při nejvyšším rozlišení se čas potřebný pro zpracování obrazu dostane až na několik sekund, zatímco při nejnižším se například pro SURF dostaneme na rychlost kolem 6 snímků za sekundu. Nevýhodou přílišného snižování rozlišení je snížení počtu bodů, ve kterých se vyhledávají významné body a tím dochází ke zhoršení detekce.

Zdroje

[1] HRÚZ, Marek. *SIFT, SURF, MSER* [online]. Plzeň, 2015 [cit. 2016-05-01]. Dostupné z: http://www.kky.zcu.cz/uploads/courses/mpv/04/materialy04.pdf

[2] RUBLEE, Ethan, Vincent RABAUD, Kurt KONOLIGE a Gary BRADSKI. *ORB: an efficient alternative to SIFT or SURF*. Kalifornie. Dostupné také z: https://www.willowgarage.com/sites/default/files/orb_final.pdf

[3] ORB (Oriented FAST and Rotated BRIEF). *OpenCV* [online]. 2014 [cit. 2016-05-03]. Dostupné z: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html

[4] TRÁVNÍČEK, Vojtěch. *POROVNÁVÁNÍ VÝZNAMNÝCH BODŮ PRO DETEKCI OBJEKTŮ V OBRAZE*. Brno, 2013. Dostupné také z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=69136 . Bakalářská práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedoucí práce Ing. Vratislav Harabiš.

Webové stránky

(1) http://forums.trossenrobotics.com/showthread.php?7419-Camer-is-built-but-were-to-plug-it-in/page4

(2) https://github.com/Itseez/opencv_contrib

(3) http://forums.trossenrobotics.com/showthread.php?7419-Camer-is-built-but-were-to-plug-it-in/page4

Seznam přístrojů

Raspberry Pi 2

Webkamera Logitech s tvarovatelným stativem Braun

Přisvětlovací USB lampička