**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**
**FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

# VIZUÁLNE SYSTÉMY

## DOKUMENTÁCIA: ÚLOHA 1

**Bratislava 2017** **Michal Pavlac, Filip Miklík**

# Zadanie:

**Úloha 1 (40 bodov):**

Navrhnite a naprogramujte program (aplikáciu), ktorá bude schopná pomocou optického toku detegovať prekážky (objekty) pred kamerou a určiť ich približnú vzdialenosť. Pohyb v obraze môžete zabezpečiť pohybom kamery alebo pohybom objektov.
Počas vypracovávania zadania budete prechádzať cez jednotlivé metódy spracovania obrazu, ktoré na seba nadväzujú (konverzie medzi farebnými modelmi, hranové operácie a iné filtrácie obrazu, segmentácia, detekcia objektov, ...). Pre každú túto časť vypracujte analýzu možných riešení a odôvodnite výber Vášho postupu (napr.: prečo ste použili daný farebný model a nie iný, ...).

***Použiť môžete**: knižnicu OpenCV (alebo iné voľne dostupné knižnice), internetové zdroje, skriptá a učebnice.*
***Programovacie prostredia**: Microsoft Visual Studio, Android Studio, Matlab,...*

# Vypracovanie:

Na začiatku sme si vyhľadali informácie o téme ktorú budeme spracovávať. Najprv sme sa zamerali na optický tok. Detekciu prekážok sme implementovali hlavne pomocou tutoriálov na internete.

Pri spustení programu si nastavíme zdroj vstupného obrazu, ktorý je v našom prípade typu *VideoCapture* zo zdroja 0, čo je v našom prípade jediná webkamera notebooku (pri použití externej webkamery prepíšeme zdroj na 1). Detekcia prekážok pomocou optického toku sa vykonáva na každom frame zvlášť, čiže aktuálny snímok z webkamery presmerujeme pomocou c++ streamu do premennej typu *Mat*, tú následne prekonvertujeme do grey modelu, s ktorou už detekčné funkcie následne pracujú.

**Využívali sme nasledovné funkcie:**

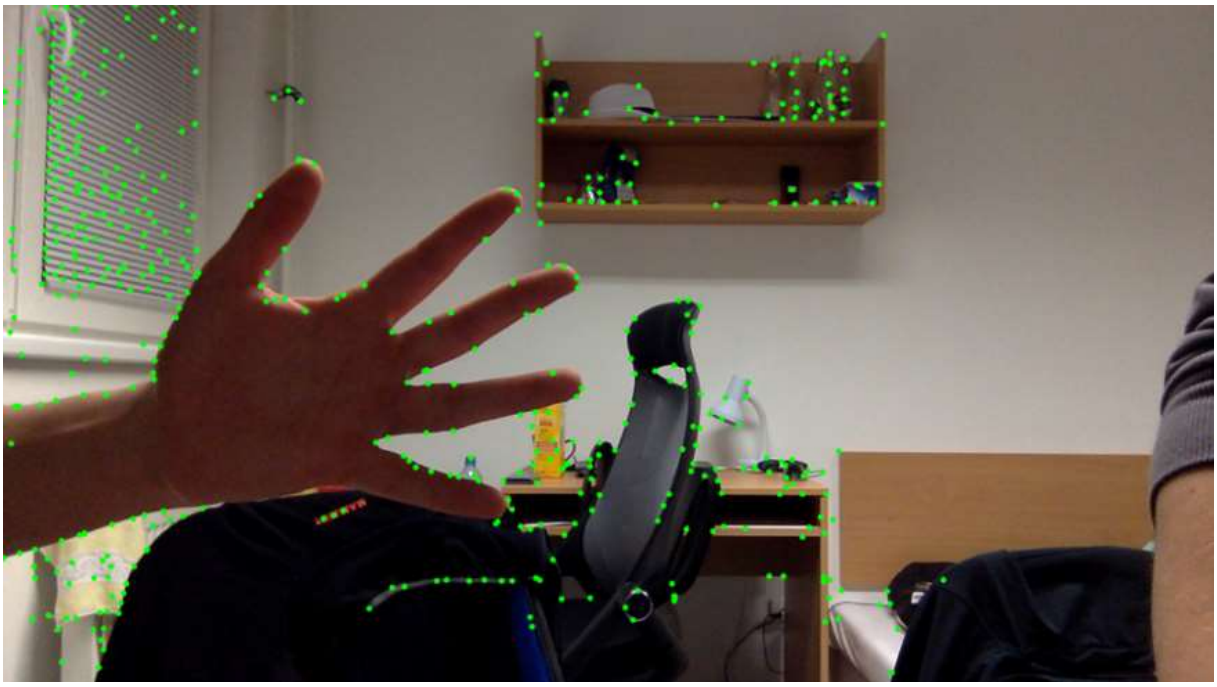
*goodFeaturesToTrack* – nájdenie hrán najmä v oblasti ostrých vrcholov objektov v obrázku

*cornerSubPix* – upresneni polohy rohov na obrázku

*calcOpticalFlowPyrLK* – kalkulácia optického toku pomocou iteratívnej Lucas-Kanade metódy s pyramídami

*Parametre týchto funkcií sme si vyhľadali v dokumentácií opencv a vhodne nastavili

Na nasledujúcom obrázku môžeme vidieť ako vyzerá grafická interpretácia našej detekcie prekážok pomocou optického toku:



Ďalej sme sa zaoberali problémom určovania vzdialenosti predmetu známych rozmerov od kamery. Tak isto sme si vyhľadali na internete spôsoby, akými by tento problém bolo možné riešiť. Rozhodli sme sa na tento účel použiť Houghovu transformáciu, konkrétne pre určovanie kruhov. Odmerali sme si polomer predmetu, v našom prípade CDčka, ktoré malo polomer 6cm. Položili sme ho pred kameru do určitej známej vzdialenosti. Zistili sme si polomer kruhu (CD) v pixeloch. Vďaka týmto parametrom sme dokázali vypočítať ohniskovú vzdialenosť kamery. Vďaka tomu sme dostali poslednú potrebnú premennú do vzorca:

*Ohn. vzdialensoť = (polomer v pixeloch * známa vzdialenosť od kamery v cm)/polomer v cm*

a dokážeme určovať vzdialenosť daného kruhového predmetu od kamery.

**Použili sme tieto funkcie:**

*houghCircles* – vstupom je obraz v tzv. gray modeli (čierno-biely) a výstupom je pole 3-zložkových vektorov, kde prvé 2 zložky sú súradnice stredu a tretia súradnica je polomer kruhu (všetko v pixeloch)

*medianBlur* – rozostrenie obrazu, zlepšuje detekciu kruhov, znižuje pravdepodobnosť detekcie nepravého kruhu

*circles* – vykreslí do vstupného obrazu kruh s určeným polomerom a farbou

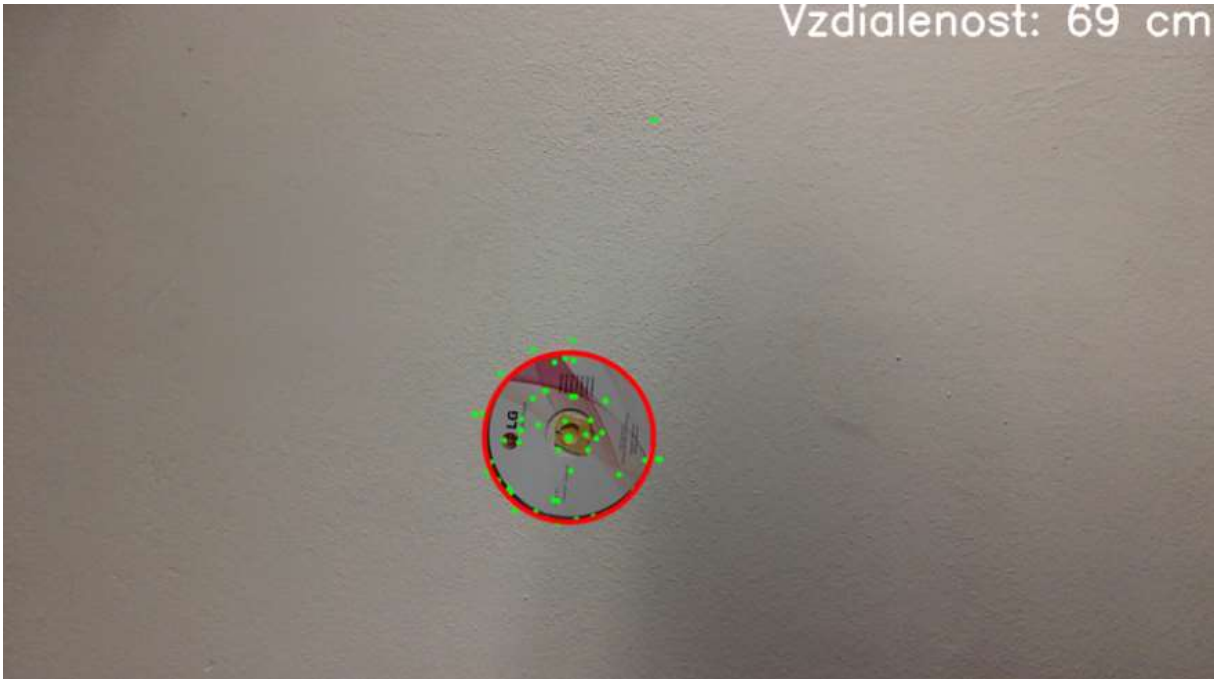
*putText* – vypíše do výstupného obrazu text (v našom prípade zistenú vzdialenosť)

*cvtColor* – konverzia medzi farebnými modelmi

*Parametre týchto funkcií sme si vyhľadali v dokumentácií opencv a vhodne nastavili

Na nasledujúcom obrázku môžeme vidieť ako vyzerá grafická interpretácia nášho určovania vzdialenosti:


Detekcia prekážok spolu s určením vzdialenosti vyzerá takto:



**Zaujímavosti a postrehy pri riešení zadania:**

- Experimentovali sme so zvyšovaním jasu a saturácie obrazu, čo sa ale nepreukázalo ako nápomocné pri zlepšení detekcie kužníc, preto sme túto úpravu obrazu nepoužili.
- Ďalej sme skúšali farebné maskovanie obrazu (AND maskovanie). Dospeli sme k názoru, že farebné maskovanie je reálne účinné, avšak detekovaná prekážka (kruh) musí mať zväčša tú farbu, ktorú sa snažíme maskovaním zvýrazniť, čiže toto riešnie je nevhodné pre viacfarebné snímané objekty.
- Pri hľadaní spôsobov implementácie optického toku sme narazili na jednom fóre na funkciu Night Mode, ktorú sme zo zaujímavosti použili v našom kóde. Táto funkcia demonštruje, že detekcia môže byť vykonávaná na inom než zobrazovanom obraze (*Mat* resp. *Image*). V praxi to funguje tak, že aktuálny obraz zo vstupu sa skonvertuje na obraz typu grey a uloží ako iná premenná s ktorou následne pracujú detekčné funkcie a pôvodný farebný obraz sa funkciou *Scalar::all* celý začierni. Na konci po výpočte sa detekované prekážky resp. kruhy vykreslia do začierneného obrazu ale v skutočnosti reprezentujú prekážky zachytené v grey obraze.