Detekce cesty a plánování dráhy pro RC autíčko

Petr Dvořáček xdvora0n@stud.fit.vutbr.cz
Oliver Lelkes xlelke00@stud.fit.vutbr.cz

23. prosince 2014

1 Úvod

Tento dokument popisuje implementaci projektu do předmětu Robotika. Cílem projektu bylo za použití Raspberry PI (verze A) a kamery detekovat a naplánovat cestu RC autíčka. Cesta je definována dvěma pruhy jeden černý, který se nachází vlevo, a druhý bílý, který se nachází na pravé straně. Tyto pruhy je potřeba detekovat a snažit se, aby je RC autíčko nepřejelo.

2 Návrh systému

Pořízený obrázek O nejprve upravíme do odstínů šedi $G = greyscale(O)$. Pro každý pixel $g_{i,j}$ obrázku G aplikujeme dvakrát prahování

$$t_{i,j} = \begin{cases} B & g_{i,j} < BLACK\_THRESHOLD \\ W & g_{i,j} > WHITE\_THRESHOLD \\ \_ & else \end{cases}$$

Jeden práh $WHITE\_THRESHOLD$ je pro bílou a druhý $BLACK\_THRESHOLD$ pro černou barvu, protože víme, že pruh je černý respektive bílý. Vznikne tak matice T, jejíž jednotlivé prvky mohou nabývat tří hodnot – černá B, bílá W anebo šum $\_$ (podtržítko čili prázdná hodnota). Tuto matici (obrázek) konvertujeme na menší matici T_s. Tento krok je poměrně důležitý jednak se odstraní případný šum černé či bílé barvy, jednak nám pomůže urychlit výpočet skóre pro zatáčení.

Podle matice T_s a jejich hodnot B resp. W sečteme hodnoty v skórovacích maticích Q_B resp. Q_W. Skóre pro otočení doprava je pak suma prvků z Q_B, které nabývají hodnoty B v matici T_s. Příklad můžete vidět v tabulce 1. Skóre pro otočení doleva se vypočte analogicky. Pro zjednodušení můžeme Q_W vytvořit vertikálním překlopením Q_B.

Zatáčení je realizováno pomocí pulzů standardního PWM řízení pro serva. Rámec PWM má velikost $20ms$ a šířka pulzu (duty cycle) je v rozmezí hodnot $1ms$ a $2ms$. Experimenty, pomocí souboru `pwm_ctrl.py`, bylo zjištěno, že hodnota $1ms$ interpretuje odbočení doprava a $2ms$ odbočení doleva. Hodnota $1.5ms$ značí, že autíčko pojede rovně. Přepočet skóre na šířku pulzu můžeme realizovat třeba pomocí trojčlenky nebo libovolné vhodné funkce. V našem projektu jsme použili funkci skokovou.

0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7
0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8
0	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8
0	1	2	3	3	4	4	5	5	6	6	7	7	8	8	9
0	1	2	3	4	4	5	5	6	6	7	7	8	8	9	9
0	1	2	3	4	4	5	5	6	6	7	7	8	8	9	9
0	1	2	3	4	5	5	6	6	7	7	8	8	9	9	10
0	1	2	3	4	5	5	6	6	7	7	8	8	9	9	10
0	1	2	3	4	5	6	6	7	7	8	8	9	9	10	10
0	1	2	3	4	5	6	7	7	8	8	9	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabulka 1: Příklad výpočtu skóre pro odbočení doprava podle matice Q_B. Barvou jsou znázorněny černé pruhy získané prahováním. Skóre pro odbočení potom odpovídá hodnotě 54. Při tvorbě skorovací matice jsme chtěli dát vyšší váhu barvě, která se nachází blíž, než té, která se nachází dále od autíčka. Rovněž jsme zanedbali poslední dva řádky, kde se může nacházet stín autíčka nebo samotné autíčko.

3 Impelmentace systému

Výrobce Raspberry PI dodává kromě desky i vlastní kameru, pro níž jsou oficiální knihovny pro Python a Bash. Proto jako programovací jazyk byl zvolen jazyk Python. Bohužel použití originální kamery od výrobce přináší několik problémů, se kterými jsme se v projektu potýkali. Prvním z nich je ten, že kamera nemá široký úhel záběru, takže na výsledném obrázku nemusíme vidět pruhy cesty, což jsme vyřešili navrženým algoritmem. Druhý problém je počet pořízených snímků za sekundu a rychlost jejich zpracování. Dalším je ten, že kamera nefotí v odstínech šedi.

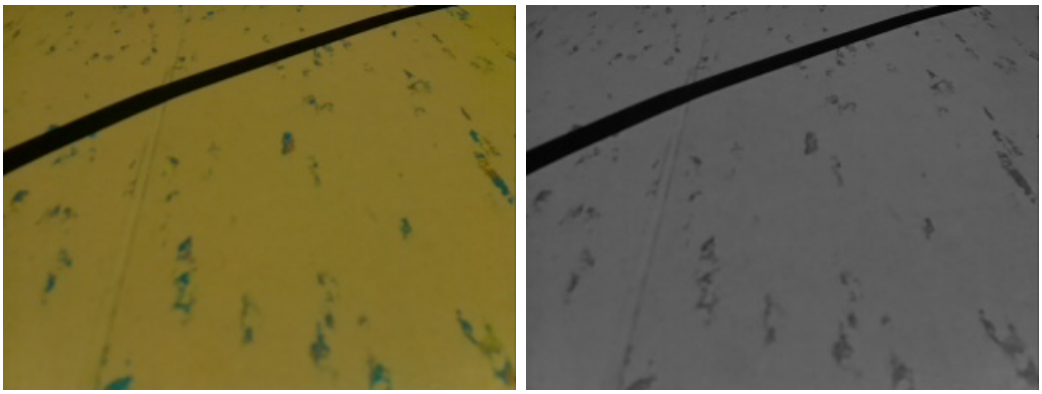
Pro snažší zpracování obrazu je totiž vhodnější zkonvertovat barevný obrázek do odstínů šedi a dané pruhy detekovat pomocí jejich odstínu. Tuto operaci jsme provedli softwarově za pomocí knihovny OpenCV[1]. Na tento obrázek v odstínech šedi jsme aplikovali algoritmus uvedený v předchozí kapitole.

Druhý problém jsme vyřešili snížením rozlišení kamery na 320x240 čímž jsme získali 17 snímků za sekundu. Tato rychlost zpracování obrazu je podobná lidskému zraku. Jediný problém s tým byl, že přístup do pole v jazyce Python přináší velmi velkou režii. Provedení průchodu matice o velikosti 320x240 spotřebovalo cca. 24 sekund. Tento průchod je pro detekci objektů v reálném čase nežádoucí a nepřijatelný. Z tohoto důvodu byl použit jazyk Cython[2], který odpovídá syntaxi jazyku Python a je navíc obohacen o statické typy a ukazatele. Jazyk Python je tedy podmnožinou jazyka Cython. Jeho princip funguje následovně, nejdříve se kód v Cythonu přeloží do jazyka C, který se zkompiluje a vznikne tak knihovna pro Python. Ta se pak může spouštět v interpreteru Pythonu.

3.1 Instalace

Je nutností míti na Raspebrry PI nainstalovaný modul pro kameru a v nastavení povolenou kameru. Dále pro instalaci je potřeba míti nainstalovaný Python2 a Cython. Přeložení kódu provedete pomocí příkazu `python setup.py build_ext --inplace` čímž vlastně vytvoříte knihovnu

[1] `http://opencv.org`
[2] `http://cython.org`



Obrázek 1: Vpravo původní obrázek z kamery. Vlevo obrázek zkonvertovaný do odstínů šedi

pro Python. Pomocí příkazu `sudo python run.py` spustíte algoritmus. Je potřeba spouštět skript jako superuživatel, neboť pracujeme s GPIO portama RPi. Na port 11 resp. 12 nastavíme PWM vodič pro otáčení resp. ovládání rychlosti.

3.2 Modul

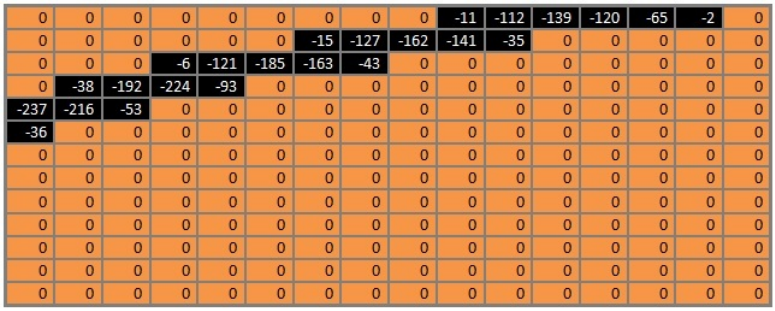
V této sekci naleznete užitečné příkazy pro používaní navrženého modulu.
`from raspberry import car` – importuje vytvořenou třídu
`c = car()` – nová instance třídy
`c.run()` – spuštění běhu algoritmu
`c.black_threshold = 42` – nastavení thresholdu pro černou
`c.white_threshold = 42` – nastavení thresholdu pro bílou
`c.speed = 7.96` – rychlost autíčka v % duty cyclu
`c.weight_matrix` – váhovací matice o velikosti 16x12

4 Experimentální vyhodnocení navrženého systému

Při testování systému v domácím prostředí jsme museli autíčko interpretovat, neboť nám nemohlo býti zapůjčeno. Raspberry PI jsme posouvali po dráze a podle výpisů ze standardního výstupu terminálu jsme naše imaginární autíčko otáčeli. Tento proces je ilustrován na obrázcích 1 a 2.

Pomocí experimentů bylo zjištěno, že zatáčení funguje správně jen pro specifické osvětlení. Takže když třeba naše imaginární autíčko vjelo do stínu, zatáčelo furt doprava, neboť snímalo hodně černé barvy. Tento problém jsme chtěli řešit pomocí posouvání prahů podle histogramu obrázku.

Podle experimentů na reálném autíčku bylo nastaveny správné hodnoty PWM signálu pro určitá skóre. Avšak při testování rychlosti jízdy nastal závažnější problém. Pro korektní fungování algoritmu se nesmí připustit rozmazání obrázků z kamery, která pro tyto účely není stavěná. To se dá zajistit tak, že rychlost jízdy musí býti pomalá – v centimetrech za sekundu. Bohužel motorek autíčka je silnější než jsme očekávali. Pulzně šířková modulace pro snížení rychlosti motorku nemohla být použitá, neboť servro autíčka je velice citlivé. Drobné zvýšení impulzu vedlo ke změně rychlosti v řádech metru za sekundu. Řízení motorku tedy odpovídá nespojité skokové funkci.



Obrázek 2: Informace z obrázku po prahování a zmenšení ($white\_threshold = 110$, $black\_threshold = 27$). Hodnoty menší než 64 se zanedbají a získané skóre se pak přepočte podle skorovací matice, vizte tabulku 1. Výsledné skóre pro otočení doleva a doprava odpovídá hodnotám 0 a 54. Autíčko by se tedy mělo otočit doprava.

5 Závěr

Výsledkem této práce je modul pro Python, který umožňuje detekci cesty podle výše zmíněného systému pro Raspberry PI a PiCameru. Rovněž řeší zatáčení autíčka. Při testování výsledného algoritmu nastal neočekáváný problém s nevyhovujícím motorkem autíčka – jelo příliš rychle. Experimenty pomocí zvětšování a zmenšování duty cyclu nebyla zjištěna vhodná hodnota pro pohyb v centimetrech za sekundu.

Možné pokračování projektu vidíme ve třech krocích. První je použití vhodnějšího motorku pro lepší demonstraci jízdy autíčka. Druhým krokem je zahrnutí posouvání prahů černé a bílé podle histogramu obrázku. Třetím krokem je použití logiky pro přejetí dráhy a následného zotavení se z chyby.