ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STROJNÍ

ÚSTAV MECHANIKY, BIOMECHANIKY A MECHATRONIKY

Odbor mechaniky a mechatroniky

Bakalářská práce

Model asistenčního systému pro couvání a parkování

Praha, 2019 **Jan Hladík**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hladík** Jméno: **Jan** Osobní číslo: **465355**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav mechaniky, biomechaniky a mechatroniky**

Studijní program: **Teoretický základ strojního inženýrství**

Studijní obor: **bez oboru**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Model asistenčního systému pro couvání a parkování

Název bakalářské práce anglicky:

Model of the reversing and parking assist system

Pokyny pro vypracování:

1) Seznamte se aktuálními asistenčními a kamerovými systémy v automobilech.
2) Připravte simulační (výpočetní) model automobilu pro predikci a plánování trajektorie.
3) Vyberte vhodnou platformu pro realizaci asistenčního systému.
4) Navrhněte rozhraní pro řidiče.
5) Sestavte funkční model pro demonstraci.

Seznam doporučené literatury:

[1] Sekerka, M.:Systém pro plánování složitějších parkovacích manévrů, diplomová práce, ČVUT v Praze, 2018.
[2] Reeds, J. A., Shepp, L. A.:Optimal paths for a car that goes both forwards and backwards, Pacific J. Math. 145, zv. 2, s. 367-393, 1990.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Petr Beneš, Ph.D., odbor mechaniky a mechatroniky FS

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **29.04.2019** Termín odevzdání bakalářské práce: **16.08.2019**

Platnost zadání bakalářské práce: _____

Ing. Petr Beneš, Ph.D. podpis vedoucí(ho) práce	prof. Ing. Milan Růžička, CSc. podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání Podpis studenta

Anotační list

Jméno autora:	Jan Hladík	
Název bakalářské práce:	Model asistenčního systému pro couvání a parkování	
Anglický název:	Model of the reversing and parking assist system	
Akademický rok:	2018/2019	
Obor studia:	Teoretický základ strojního inženýrství	
Ústav/odbor:	Ústav mechaniky, biomechaniky a mechatroniky Odbor Mechaniky a mechatroniky	
Vedoucí bakalářské práce:	Ing. Petr Beneš, Ph.D.	
Bibliografické údaje:	Počet stran:	**44**
	Počet obrázků:	**28**
	Počet příloh:	1 x CD

Klíčová slova:	Parkovací asistent, plánování trajektorie, „bird‘s eye view“ transformace
Keywords:	Parking assist system, trajectory planning, bird’s eye view transformation
Anotace:	Bakalářská práce se zabývá vytvořením modelu parkovacího asistenta, přesněji reverzní kamery s pohledem shora, do nějž je vykreslována predikovaná trajektorie v závislosti na úhlu natočení kol.
Abstract:	Bachelor thesis with a focus on the model of parking assist system, more precisely a reverse camera with a top-view, into which the predicted trajectory depending on the angle of rotation of wheels is drawn.

Čestné prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

V Praze, dne …….. ……

Podpis

Obsah

Anotační list 3

Čestné prohlášení 4

Obsah 5

Seznam obrázků 7

Úvod 9

Cíl práce 10

1. Současný stav zkoumané problematiky 11

1.1. Parkovací sensory 11

1.1.1. Ultrasonické senzory 11

1.1.2. Elektromagnetické senzory 12

1.2. Couvací a 360° kamery 12

1.3. Park assists 13

1.4. Plně autonomní systémy 13

2. Model vozidla 14

2.1. Kinematický model vozidla 14

2.2. Analytický výpočet trajektorie 16

3. Transformace obrazu 18

3.1. Bird's eye view (top-view) transformation 18

3.1.1. Posunutí obrazu 18

3.1.2. Rotace obrazu 19

3.1.3. Homografická projekce 20

3.1.4. Model kamery 21

3.1.5. Pozice kamery 22

3.1.6. Celkový transformační model 23

3.1.7. Zkreslení kamery 25

4. Vývojové platformy 26

4.1. Arduino 26

4.2. Android/IOS 26

4.3. Raspberry pi 26

5. Model pro demonstraci 28
5.1. Rozhraní 28
5.2. Geometrie 28
5.3. Použitý hardware 31
5.3.1. Raspberry Pi 3 Model B+ 31
5.3.2. Raspberry Pi NoIR Camera V2 31
5.3.3. Ostatní hardware 32
5.4. Použitý software 33
5.4.1. Operační systém 33
5.4.2. Programovací jazyk Python 33
5.4.3. Knihovna OpenCV 33
5.4.4. Knihovna Tkinter 34
5.4.5. Knihovna NumPy 34
5.4.6. Knihovna RPi.GPIO 34
5.4.7. Knihovna PIL 34
5.5. Získání transformační matice 35
5.6. Vykreslení trajektorie 37
Závěr 42
Literatura 43

Seznam obrázků

Obr. 1 – Volkswagen Park pilot – převzato z [5] 12
Obr. 2 – Kinematický model vozidla – převzato z [6] 14
Obr. 3 – Rotační pohyb (převzato z [6]) 15
Obr. 4 – Trajektorie parkovacího manévru (převzato z [6]) 16
Obr. 5 – Reprezentace umístění pixlů vstupního obrazu (vlevo), po posunutí (vpravo) 18
Obr. 6 – Definice rotace v levotočivém souřadnicovém systému (převzato z [8]) 19
Obr. 7 – Model dírkové komory 21
Obr. 8 – Umístění kamery v souřadném systému 22
Obr. 9 – Zobrazení zorného pole kamery 23
Obr. 10 – Proces transformace. (**a**) Zdrojový obraz perspektivy; (**b**) Obraz po transformaci; (**c**) obraz po odstranění přebytečných částí; (**d**) přeškálovaný obraz, převzato z [8] 24
Obr. 11 – Geometrické zkreslení obrazu a) nezkreslený; b) poduškovité zkreslení; c) soudkovité zkreslení, převzato z [16] 25
Obr. 12 – Rozměry Škoda Octavia kombi II (převzato z [12]) 28
Obr. 13 – Souřadný systém *X'*, *Y'* transformovaného obrazu z kamery a umístění kamery v rovině *X'*-*Y'* 29
Obr. 14 – Umístění kamery v rovině *Y'*-*Z'* a ukázka zjednodušení tvaru zadní části automobilu 29
Obr. 15 – Rozměry a geometrie modelu zadní části automobilu s uchycením kamery (rozměry v mm) 30
Obr. 16 – Držák kamery vyrobený pomocí 3D tisku 30
Obr. 17 – Rozložení pinů v IDC konektoru na Raspberry PI 3 B+ (převzato z [13]) 31
Obr. 18 – Zapojení spínačů k GPIO (vytvořeno v Circuit Diagram) 32
Obr. 19 Ukázka kódu pracujícího s GPIO 32
Obr. 20 – Perspektivní transformace mezi systémy *U*-*V* a *X'*-*Y'* 35
Obr. 21 – Ukázka transformace, v levé části výstupní obraz z kamery, v pravé části obraz po provedení perspektivní transformace 36
Obr. 22 – Trajektorie v souřadném systému Y'-X' 37
Obr. 23 – Ukázka kódu pro výpočet x-ové souřadnice trajektorie 38
Obr. 24 – Ukázka kódu vytvářejícího pole pro vykreslení trajektorie 39
Obr. 25 – Ukázka výstupu modelu s transformací obrazu a vykreslením trajektorie pro natočení kol $\varphi = 0°$ 40
Obr. 26 – Ukázka výstupu modelu s transformací obrazu a vykreslením trajektorie pro natočení kol $\varphi = 11°$ 40

Obr. 27 – Vstupní obraz parkoviště 41
Obr. 28 – Výstupní obraz parkoviště 41

Úvod

V poslední dekádě začal boom na poli vývoje podpůrných systémů pro jednoduší a komfortnější ovládání automobilů. Parkovací asistenční systémy se stávají čím dál tím více propracované a rozšiřují se do více modelů automobilů.

Cílem práce je seznámit se se stávajícími asistenčními a kamerovými systémy v automobilech, připravit simulační model pro predikci trajektorie, vybrat asistenční systém, provést jeho zjednodušení pro demonstrační model asistenčního systému a zvolit vhodnou platformu, na které se poté bude model realizovat.

Práce je rozdělena do pěti kapitol.

První kapitola obsahuje stručnou historii parkovacích asistenčních zařízení, přehled současného stavu parkovacích asistenčních systémů a příklad nejpoužívanějších z nich.

Ve druhé kapitole práce se věnujeme zavedení zjednodušeného kinematického modelu automobilu a analytickému odvození trajektorie vozidla v závislosti na geometrii vozidla a úhlu natočení kol.

Třetí kapitola práce pojednává teoreticky o procesu generování vrchního pohledu tzv. bird's eye a jednotlivých částí, ze kterých se tento proces transformace skládá. V této kapitole je i zmínka o geometrických zkresleních reálných kamer.

Čtvrtá kapitola práce obsahuje přehled a zhodnocení jednotlivých vývojových platforem, které by připadaly v úvahu pro realizaci modelu asistenčního zařízení.

V páté kapitole je popsán zjednodušený statický model reverzní kamery s Bird's eye transformací obrazu a predikcí trajektorie. Zmiňujeme použitý hardware, software, geometrii demonstračního modelu a její odvození od reálného automobilu. Zde také popisujeme důležitou část demonstračního modelu, kterou je vytvořený zdrojový kód pro postup získání transformační matice, provedení transformace a vykreslení predikované trajektorie do transformovaného obrazu. Model je realizován na platformě Raspberry Pi a naprogramovaný v programovacím jazyku Python s využitím knihovny OpenCv.

Cíl práce

Cílem práce je seznámit se se stávajícími asistenčními a kamerovými systémy v automobilech, připravit simulační model pro predikci trajektorie, vybrat asistenční systém, provést jeho zjednodušení pro demonstrační model asistenčního systému a zvolit vhodnou platformu, na které se poté bude model realizovat.

Částečné cíle k naplnění hlavního cíle:

1. Seznámit se s aktuálními asistenčními a kamerovými systémy v automobilech.
2. Připravit simulační (výpočetní) model automobilu pro predikci a plánování trajektorie.
3. Vybrat vhodnou platformu pro realizaci asistenčního systému.
4. Navrhnout rozhraní pro řidiče.
5. Sestavit funkční model pro demonstraci.

1. Současný stav zkoumané problematiky

S rozvojem polovodičových součástek, obzvláště mikroprocesorů, začaly automobilky užívat čím dál tím více elektronické systémy v automobilech. Počátek tohoto trendu lze pozorovat už v sedmdesátých letech. Prvně se objevily snahy o zvýšení bezpečnosti (ABS, airbag) a snížení škodlivin ve výfukových plynech (elektronické řízení vstřikování paliva, elektronicky řízený katalyzátor). Ve druhé polovině osmdesátých let následoval vývoj směrem ke komfortu užívání vozidla řidičem. Začaly se objevovat dvouzónové klimatizace, elektronické nastavení sedadel, elektronicky řízené samočinné převodovky a posilovače řízení. Jak šel vývoj dopředu, automobily se staly doslova naplněné elektronickými systémy a jejich řídicími jednotkami. Po roce 2000 začal trend užívání elektronických parkovacích asistenčních systémů. [1]

V Současné době se stávají parkovací asistenční systémy čím dále tím více dostupné. Vyskytují se sice v základní výbavě spíše u luxusních automobilů, ale u většiny automobilů z levnější cenové kategorie se dají pořídit v rámci příplatkové výbavy.

Každá automobilka má vyvinuté své vlastní řešení parkovacích asistentů. Ty jsou dohledatelné v rámci katalogů jednotlivých výrobců pod různými názvy. Pojem „Parkovací asistenční zařízení“ je patentován dle patentu, jehož znění lze nalézt v [2]. Dále se pokusíme shrnout nejčastěji používané systémy.

1.1. Parkovací sensory

Nejjednodušší formou parkovacího asistenčního systému jsou parkovací senzory. Ty jsou obvykle umístěny v náraznících. Senzory jsou připojeny k jednotce, která dává řidiči informaci o vzdálenosti k detekovanému objektu. Vzdálenost je nejčastěji vyjádřena frekvencí pípání. U novějších automobilů je tento systém často kombinován s vizualizací směru, vzdálenosti objektu a couvací či 360° kamerou. Oba následující typy senzorů jsou navrženy tak, aby fungovaly při couvání či jízdě v malých rychlostech. Existují i jiné systémy se stejnou funkcí, jako například Lidar, ale ty nejsou zatím velmi rozšířené, a proto se jimi zde nebudeme zaobírat. [3]

1.1.1. Ultrasonické senzory

Ultrasonické parkovací senzory pracují na principu vysokofrekvenčních zvukových vln. Snímač vyšle vlnu a poté podle času navrácení se odrazu od detekovaného objektu spočte vzdálenost objektu od snímače. Nevýhodou těchto snímačů je, že nejsou schopné detekovat některé tipy objektů a to například: objekty s rovinnými plochami rovnoběžnými či svírajícími ostrý úhel s povrchem vozovky, úzké objekty (například tenké dráty) a objekty z materiálů pohlcujících zvukové vlny. [4]

1.1.2. Elektromagnetické senzory

Elektromagnetické senzory vytváří kolem nárazníku elektromagnetické pole. A reagují na jeho změnu, když se do tohoto pole dostávají objekty. Tyto senzory reagují na objekty různých tvarů. Pouze může docházet k nepatrnému zkreslení vzdálenosti u objektů v závislosti na jeho materiálu. Oproti ultrasonickým senzorům mají kratší detekční vzdálenost. [4]

1.2. Couvací a 360° kamery

První použití couvací kamery bylo již v roce 1956 na konceptu automobilu Buick's Centurion. Širší použití této technologie ale přišlo až kolem roku 2000. Ze začátku se vyskytovaly pouze systémy, které zobrazovaly obraz z kamery umístěné na zadní části auta. Později se přidalo propojení s parkovacími senzory. Následovalo přidání vykreslení budoucí trajektorie vázané na momentální natočení kol. Dnes už se na automobilech vyskytují i systémy s více kamerami obvykle čtyřmi, jejichž obraz je sklopený a vytváří takzvaný ptačí pohled. Takové systémy na informační display řidiči vykreslují pohled seshora na jeho vozidlo a blízké okolí. Do tohoto obrazu velmi často dokreslují i plánovanou trajektorii automobilu a informace z parkovacích senzorů. Ukázka vzhledu parkovacího asistenta je na Obr. 1 . [3]

Obr. 1 – Volkswagen Park pilot – převzato z [5]

1.3. Park assists

Park assist je systém, jehož základní schopností je napomoci s bezpečným zaparkováním (předejití kolize s objekty v okolí vozidla). Je schopný detekce objektů a volných míst k zaparkování. Tento systém sám vypočte trajektorii, jak na zvolené místo zaparkovat. Systém řídí natáčení kol a řidič pouze ovládá brzdu a plyn. První verze tohoto systému byla ohlášena v roce 2003. Z počátku byla funkce omezená a bylo těžké detekovat malé objekty, jako psy, osoby či kočárky. Díky zlepšení softwaru, implementaci detekce obrazu z kamer a dat z parkovacích senzorů se tyto systémy postupně staly schopné detekovat i tyto objekty, a navíc rozpoznávat i dopravní značení. Tento asistent lze využít pouze do modelů s automatickou převodovkou. [3]

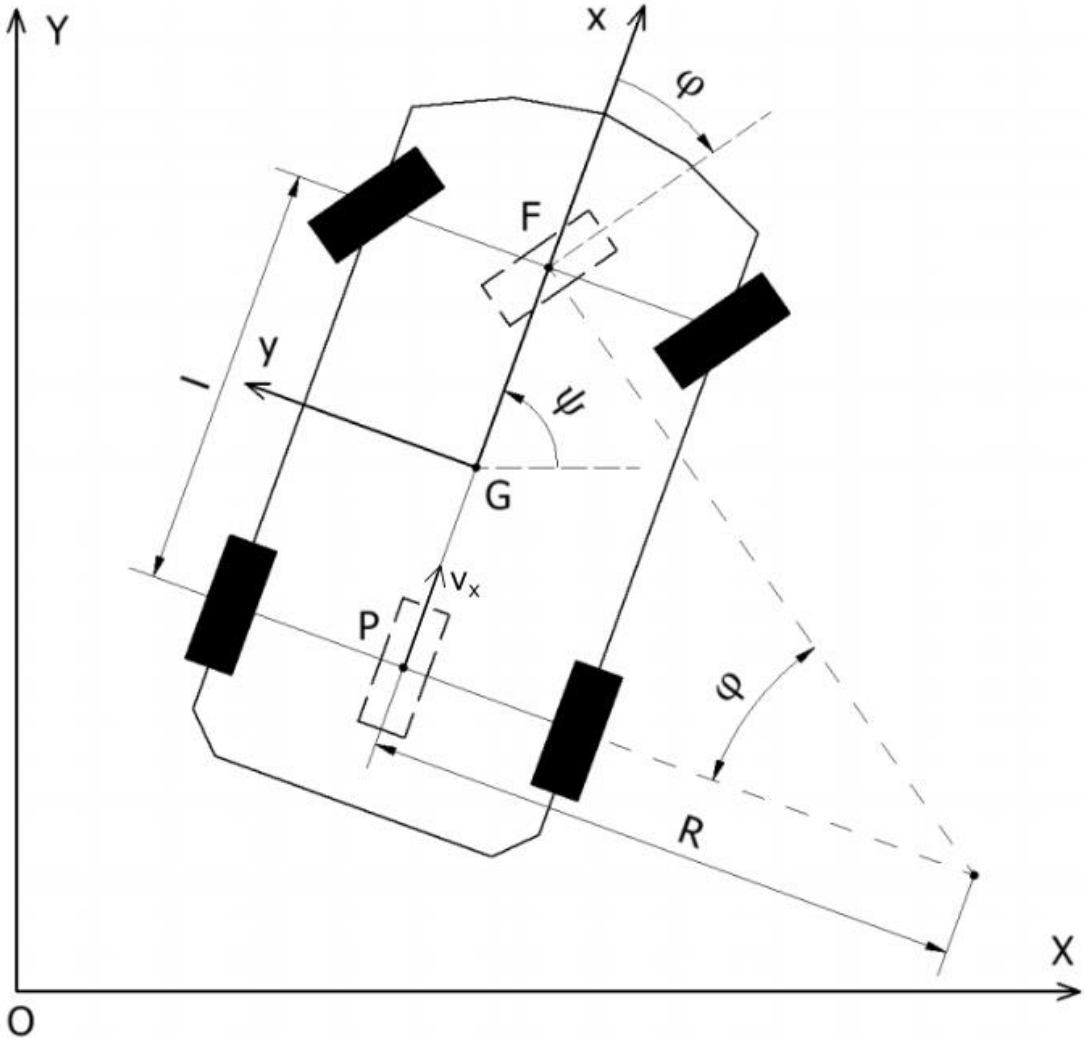
1.4. Plně autonomní systémy

Většina těchto systémů je schopná po zvolení místa sama vůz zaparkovat bez jakéhokoliv zásahu řidiče. Obvyklý úkon je parkování do kolmých či paralelních míst. Tyto systémy jsou dokonce schopny po dokončení parkovacího manévru samy zabrzdit ruční brzdu, aby se vůz zajistil proti rozjetí. Automatické parkování je užitečné a za poslední roky v této oblasti došlo k významnému vývoji, ale i tak stále hrozí riziko, že systém selže díky neočekávané situaci, na kterou by řidič byl schopen reagovat, a navíc ve spoustě států není vyřešena legislativa týkající se autonomního pohybu vozidla. [3]

2. Model vozidla

2.1. Kinematický model vozidla

Pro modelování pohybu automobilu se obvykle využívá převedení z čtyřstopého modelu na dvoustopý náhradou kol na nápravě kolem virtuálním umístěným doprostřed nápravy (viz Obr. 2). Pro tento model není důležitá znalost natočení každého kola, jelikož je možné jej dopočítat z rozchodu kol. Dalším předpokladem pro výpočet je pomalý pohyb vozidla, při kterém nedochází k prokluzu kol.



Obr. 2 – Kinematický model vozidla – převzato z [6]

Vozidlo je popsáno v globálním souřadnicovém systému souřadnicemi bodu G = [X(t), Y(t)] a natočením lokálního souřadnicového systému ke globálnímu ψ(t). Souřadnice globálního systému jsou značeny velkými písmeny, lokálního malými. Model se vztahuje ke geometrickému středu

reprezentovanému bodem *G* ležícímu uprostřed spojnice náprav. Z obrázku dvoustopého modelu je zřejmé, že poloměr oblouku R lze vyjádřit jako (viz Obr. 2):

$$R(\varphi) = \frac{l}{\tan \varphi} \tag{2.1}$$

Rychlost bodu *P* v lokálním souřadnicovém systému lze vyjádřit jako:

$$v_P^{xy} = \begin{bmatrix} v_x \\ 0 \end{bmatrix} \tag{2.2}$$

Poté lze vyjádřit úhlovou rychlost otáčení vozidla ω jako:

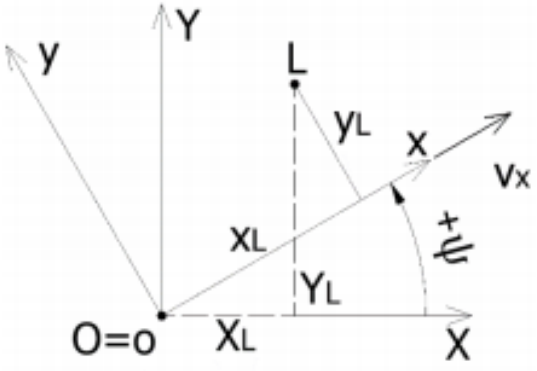
$$\omega = \dot{\psi} = \frac{v_x}{R} \tag{2.3}$$

$$\dot{\psi} = \frac{v_x}{l} \tan \varphi \tag{2.4}$$

Pro transformaci z lokálního do globálního souřadnicového systému lze psát (viz Obr. 3):

$$\begin{bmatrix} X_L \\ Y_L \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \cdot \begin{bmatrix} x_L \\ y_L \end{bmatrix} \tag{2.5}$$

$$\begin{bmatrix} v_X \\ v_Y \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \cdot \begin{bmatrix} v_x \\ 0 \end{bmatrix} \tag{2.6}$$



Obr. 3 – Rotační pohyb (převzato z [6])

Vozidlo koná obecný rovinný pohyb, tudíž rychlost bodu *G* je rozložitelná do unášivé a relativní rychlosti vozidla.

$$v_G^{xy} = v_u^{xy} + v_{rel}^{xy} = v_p^{xy} + \omega^{xy} \times r_G^{xy} = \begin{bmatrix} v_x \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{l}{2} \end{bmatrix} \dot{\psi} = \begin{bmatrix} v_x \\ \frac{l}{2}\dot{\psi} \end{bmatrix} \tag{2.7}$$

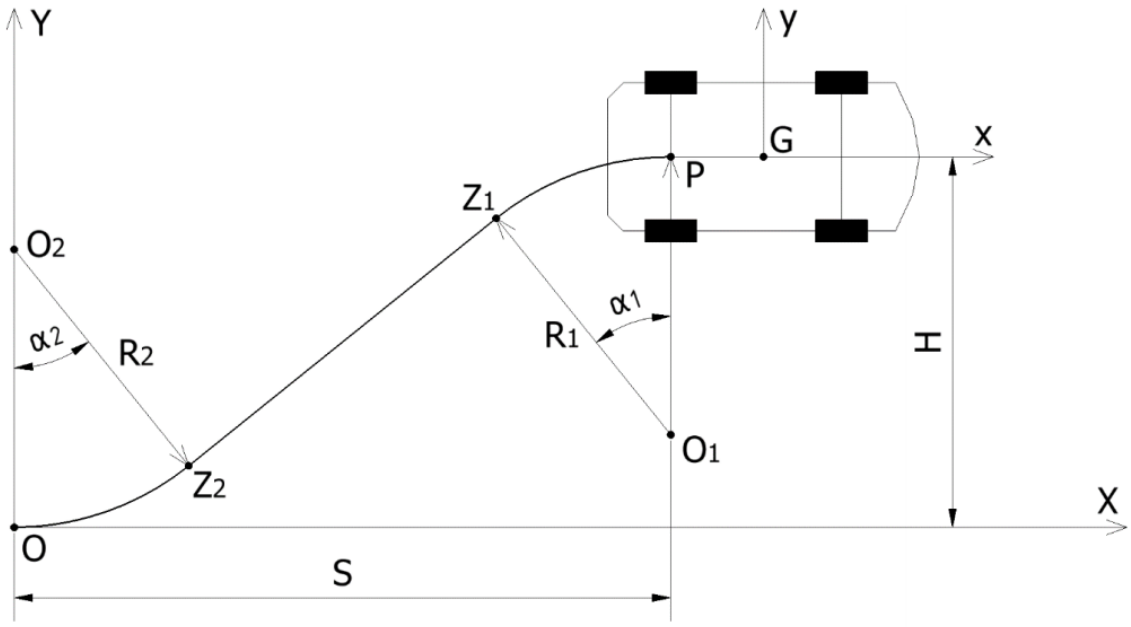
Transformaci do globálního systému lze pak vyjádřit:

$$\begin{bmatrix} X_{GX} \\ Y_{GY} \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \cdot \begin{bmatrix} v_x \\ \frac{l}{2}\dot{\psi} \end{bmatrix} \tag{2.8}$$

Kinematický model je převzat z [6].

2.2. Analytický výpočet trajektorie

Vyjdeme z předpokladu, že úhel natočení kol budeme měnit jen ve chvíli, kdy se vozidlo nepohybuje. Z předcházející kapitoly 2.1 plyne, že se trajektorie vozidla při parkování bude skládat jen z oblouků a přímek (viz Obr. 4).



Obr. 4 – Trajektorie parkovacího manévru (převzato z [6])

Na obrázku Obr. 4 jsou: O_1, O_2 středy oblouků; R_1, R_2 jim příslušné poloměry; Z_1, Z_2 body, kde oblouk tečně navazuje na přímku a bod *O* je počátkem globálního souřadného systému. Na Obr. 4

jsou souřadnice středů oblouků v globálním systému $O_1 = [S; H - R_1]$, $O_2 = [0; R_2]$. Rovnice kružnice má tvar:

$$(x-a)^2 + (y-b)^2 = r^2 \tag{2.9}$$

kde *a*, *b* jsou souřadnice středu O a *r* je poloměr oblouku.

Nás bude nadále nejvíce zajímat trajektorie vozidla z pohledu souřadného systému soustavy, vztažené k vozidlu (souřadný systém xy na Obr. 4), v závislosti na natočení kol φ . Budeme předpokládat, že úhel φ je kladně orientován ve směru hodinových ručiček od osy x. Poté lze dosadit do rovnice (2.9) z rovnice (2.1), kde je souřadnice bodu $O = [-\frac{l}{2}; R(\varphi)]$, odkud dostaneme:

$$(x + \frac{l}{2})^2 + (y - \frac{l}{\tan\varphi})^2 = (\frac{l}{\tan\varphi})^2 \tag{2.10}$$

Z rovnice (2.10) vyplývá, že není definována pro $\varphi = 0$, což je stav, kdy auto jede rovně a trajektorie je přímka s rovnicí *y*=0. Abychom vyjádřili souřadnici *y* jako funkci *x* a φ, je nutné rovnici odmocnit:

$$\left|y - \frac{l}{\tan\varphi}\right| = \sqrt{(\frac{l}{\tan\varphi})^2 - (x + \frac{l}{2})^2} \tag{2.11}$$

Řešení této rovnice má poté následující tvar:

$$\varphi \in \left(0, \frac{\pi}{2}\right) \qquad y(x,\varphi) = \frac{l}{\tan\varphi} - \sqrt{\left(\frac{l}{\tan\varphi}\right)^2 - \left(x + \frac{l}{2}\right)^2} \tag{2.12}$$

$$\varphi \in \left(-\frac{\pi}{2}, 0\right) \qquad y(x,\varphi) = \frac{l}{\tan\varphi} + \sqrt{(\frac{l}{\tan\varphi})^2 - (x + \frac{l}{2})^2} \tag{2.13}$$

V této kapitole jsme odvodili trajektorii pro různé natočení kol, z pohledu soustavy vztažené k autu. Výsledkem jsou tři různé rovnice v závislosti na velikosti φ a to pro kladné φ rovnice (2.12), pro záporné φ rovnice (2.13) a pro $\varphi = 0$ rovnice přímky *y*=0.

3. Transformace obrazu

V další části práce se zabýváme modelem reverzní kamery s predikcí trajektorie, kde z přímého pohledu kamery vytváříme sklopený pohled seshora, a proto zde uvedeme principy, kterých se při této transformaci užívá. Obecně se této oblasti práce s obrazem říká fotogrammetrie.

3.1. Bird's eye view (top-view) transformation

Bird's eye view transformation je technika generování pohledu shora. Jedná se o zpracování obrazu, při němž dochází ke geometrické modifikaci obrazu. Proces se skládá ze tří kroků: posunutí obrazu, poté jeho rotace a přeškálování následnou projekcí do roviny.

K pochopení tohoto procesu je důležitá znalost, jak je obraz ukládán. Obvykle je obraz ukládán do tří barevných kanálů RGB. Každý kanál pak definuje dvojrozměrná matice, kde každá pozice v matici reprezentuje hodnotu od 0 do 255. Pro zjednodušení můžeme dále uvažovat černobílou fotografii, která má jen jeden kanál, kde na každé pozici matice hodnota 0 až 255 představuje stupeň šedi.

3.1.1. Posunutí obrazu

V této první části procesu dochází k posunu jednotlivých pixelů vstupní matice. Tento krok je důležitý, abychom následně mohli provést rotaci obrazu. Nechť každý pixel vstupního obdélníkového obrazu má souřadnice (x, y) a posunutý obraz (X, Y), matice má h řádků a w sloupců, pak je posunutí definováno (viz [7] a Obr. 5):

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \frac{w}{2} \\ \frac{h}{2} \end{bmatrix} \tag{3.1}$$

$$\begin{matrix} (0;\,0) & \cdots & (0;\,w-1) \\ \vdots & \ddots & \vdots \\ (h-1;\,0) & \cdots & (h-1;\,w-1) \end{matrix} \qquad \begin{matrix} (-\frac{h}{2};-\frac{w}{2}) & \cdots & (-\frac{h}{2};\,\frac{w}{2}-1) \\ \vdots & \ddots & \vdots \\ (\frac{h}{2}-1;-\frac{w}{2}) & \cdots & (\frac{h}{2}-1;\frac{w}{2}-1) \end{matrix}$$

Obr. 5 – Reprezentace umístění pixlů vstupního obrazu (vlevo), po posunutí (vpravo)

3.1.2. Rotace obrazu

Rotaci lze provést vynásobením posunuté matice transformační maticí *R*. Pokud označíme (*X, Y, Z*) souřadnice vstupního obrazu a (*p, q, r*) souřadnice výstupního obrazu. Kde Z reprezentuje třetí souřadnici pro 3D obraz. Pro 2D obraz můžeme souřadnici *Z* pro každý pixel položit rovnu nule. Rotaci lze zapsat jako:

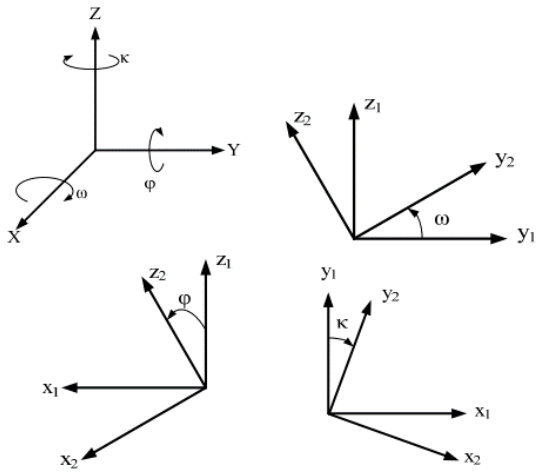
$$\begin{bmatrix} p \\ q \\ r \\ 1 \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.2}$$

kde R je rotační matice 4×4 definovaná v rovnicích (3.3), (3.4) a (3.5), ω, φ, κ jsou záporně orientované úhly vůči směrům os (viz Obr. 6). Jedná se o běžně používané transformační matice, které jsou ovšem pro grafické účely používány v levotočivém souřadnicovém systému. [7]

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\omega) & \sin(\omega) & 0 \\ 0 & -\sin(\omega) & \cos(\omega) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$R_y = \begin{bmatrix} \cos(\varphi) & 0 & -\sin(\varphi) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$R_z = \begin{bmatrix} \cos(\kappa) & \sin(\kappa) & 0 & 0 \\ -\sin(\kappa) & \cos(\kappa) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$



Obr. 6 – Definice rotace v levotočivém souřadnicovém systému (převzato z [8])

3.1.3. Homografická projekce

Posledním krokem je homografická projekce (Homographic mapping method), což je metoda, pomocí které se definuje vztah mezi dvěma odlišnými pohledy na stejnou scénu. Nechť *P* a *P'* jsou průměty stejného bodu ve dvou různých rovinách obrazů. Předpokládejme, že *P* a *P'* mají souřadnice $(x_1, y_1, z_1)^T$ a $(x_2, y_2, z_2)^T$, potom homografická projekce může být vyjádřena jako:

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = H \begin{bmatrix} x_1 \\ y_2 \\ z_2 \end{bmatrix} \tag{3.6}$$

Homografická projekce dvou bodů je rovinná projektivní transformace. Jedná se o lineární transformaci, pro kterou je nesingulární 3x3 matice *H*, homogenní transformační maticí. Podrobněji viz [8].

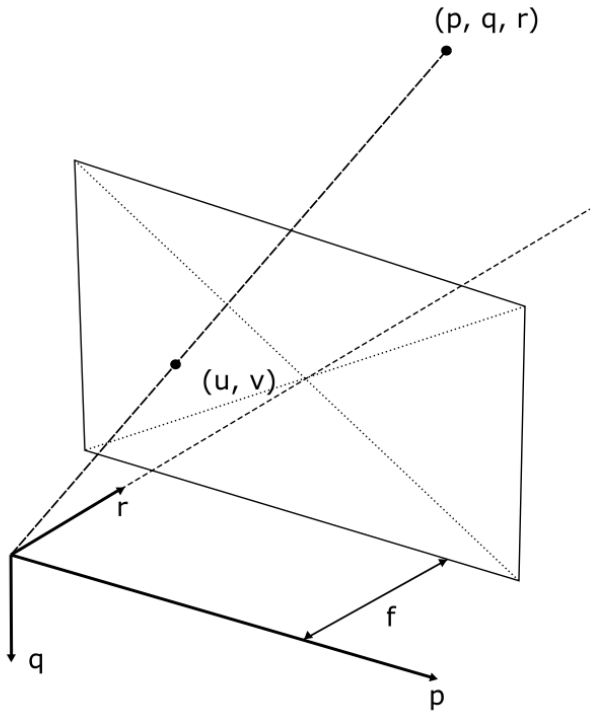
3.1.4. Model kamery

Obraz kamery je 2D rovinná projekce z 3D. Využijeme modelu dírkové komory (pinhole camera), pro kterou lze z podobnosti trojúhelníků odvodit vztahy:

$$u = \frac{f \cdot p}{f - r} + \frac{w}{2} \tag{3.7}$$

$$v = \frac{f \cdot q}{f - r} + \frac{h}{2} \tag{3.8}$$

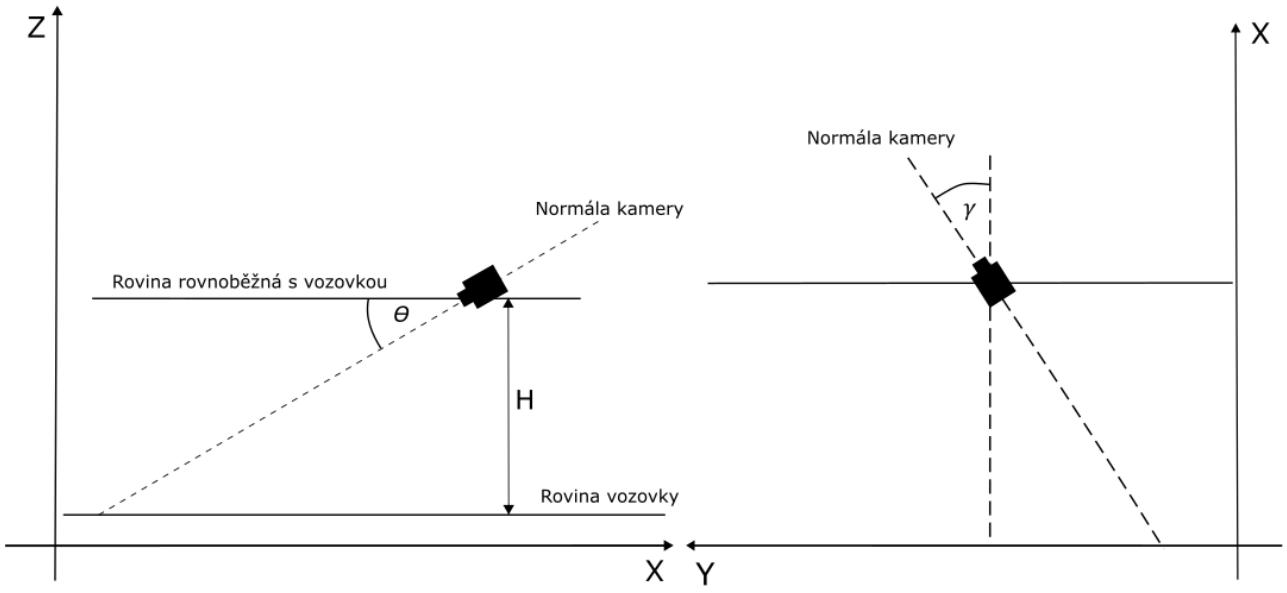
kde (*p, q, r*) jsou souřadnice bodu v 3D prostoru, (*u, v*) souřadnice promítnutého bodu ve výstupním obrazu, f ohnisková vzdálenost dle modelu dírkové komory, *h* výška obrazu, *w* šířka obrazu (viz Obr. 7). [7]



Obr. 7 – Model dírkové komory

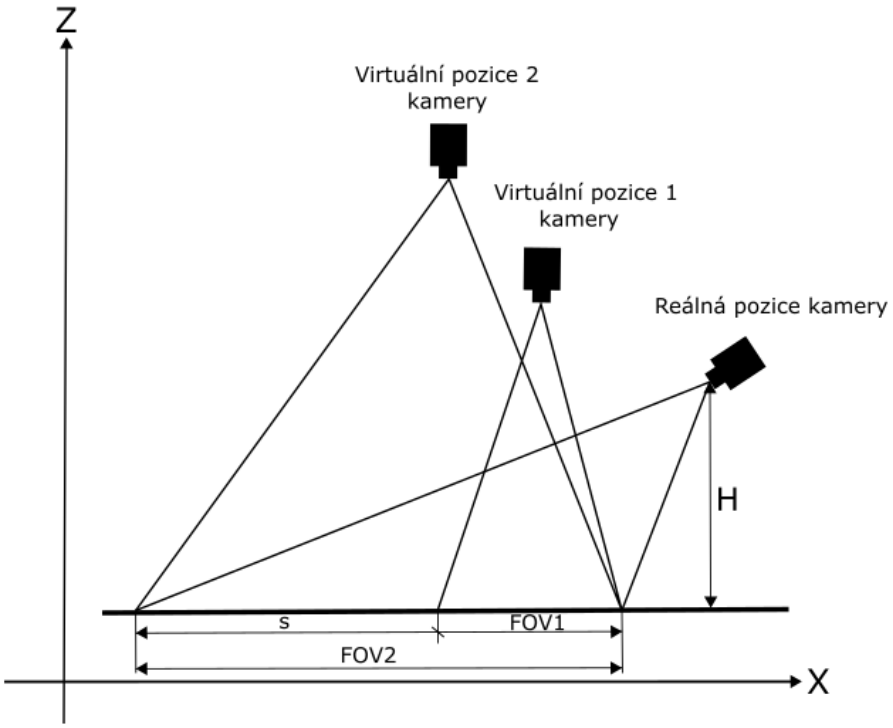
3.1.5. Pozice kamery

V reálné aplikaci se nachází kamera v zadní částí vozidla uprostřed nárazníku. K popisu se užívá pravoúhlý souřadný systém svázaný s vozidlem, kdy osa *X* směřuje ve směru vozidla, osa *Y* směřuje vlevo a osa *Z* kolmo k zemi. Obecné umístění kamery v souřadném systému je zobrazeno na Obr. 8. *H* je výška kamery nad rovinou vozovky, Θ úhel normály kamery k rovině X-Y a γ úhel mezi normálou kamery promítnutou do roviny X-Y a osou *X*. Pokud $\gamma = 180°$, je obraz zachycený kamerou pouhou lineární perspektivní projekcí obrazu prostoru za automobilem.



Obr. 8 – Umístění kamery v souřadnicovém systému

Efektivní zorné pole (FOV) kamery je vymezeno obecně kosým jehlanem s vrcholem ve středu obrazové roviny kamery. Scéna nacházející se v tomto poli je poté promítnuta do kamery. Sklon tohoto jehlanu závisí na *H,* Θ*,* γ. Promítnutá velikost scény, kterou kamera zabírá, do osy *X* se používá k označení zorného pole, např. FOV1 na Obr. 9 značí reprezentaci zorného pole pro virtuální pozici kamery 1.



Obr. 9 – Zobrazení zorného pole kamery

Pokud je kamera na Obr. 9 umístěna ve virtuální pozici 2, pak má stejné efektivní zorné pole jako reálně umístěná kamera. Jelikož je nejdůležitější řidiči zobrazit nejpřesněji prostor blíže k zadní části automobilu, je vhodnější použít spíše virtuální pozici 1, i když se přitom nevyužije část pořízeného snímku. Podrobnější popis lze nalézt v [8].

3.1.6. Celkový transformační model

Na základě kapitol 3.1.1, 3.1.2, 3.1.3, 3.1.4 a 3.1.5 lze popsat celkovou transformaci (viz [8]):

$$x' = H\frac{x\sin(\theta) + f\cos(\theta)}{-y\cos(\theta) + f\sin(\theta)} \tag{3.9}$$

$$y' = H\frac{y\sin(\theta) + f\cos(\theta)}{-y\cos(\theta) + f\sin(\theta)} \tag{3.10}$$

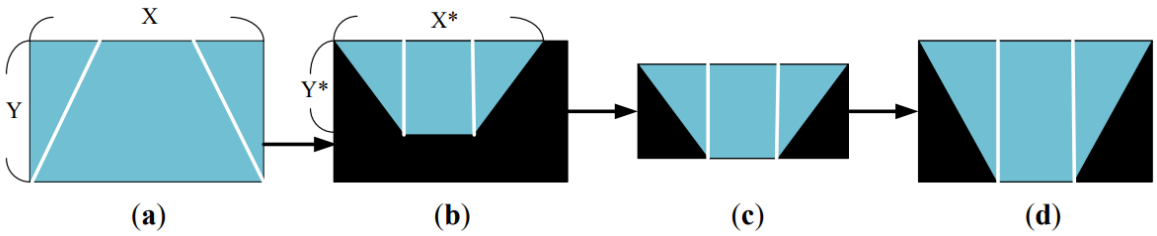
kde (*x, y*) jsou souřadnice původního obrazu a (*x', y'*) souřadnice výstupního obrazu, *H* vzdálenost kamery od země, *f* ohnisková vzdálenost kamery a θ úhel natočení kamery (viz kapitola 3.1.5).

Z rovnic (3.9) a (3.10) vyplývá, že hodnoty souřadnic výstupního obrazu *x', y'* mohou nabývat záporných hodnot a nuly. Abychom zajistili, že *x', y'* budou kladné a zároveň zachována podmínka, že bod z původního obrazu bude mapován do bodu výstupního obrazu, upravíme rovnice zavedením konstanty *d*. Získáme tím rovnice:

$$x' = H\frac{x\sin(\theta) + f\cos(\theta)}{-y\cos(\theta) + f\sin(\theta)} + d \tag{3.11}$$

$$y' = H\frac{y\sin(\theta) + f\cos(\theta)}{-y\cos(\theta) + f\sin(\theta)} + d \tag{3.12}$$

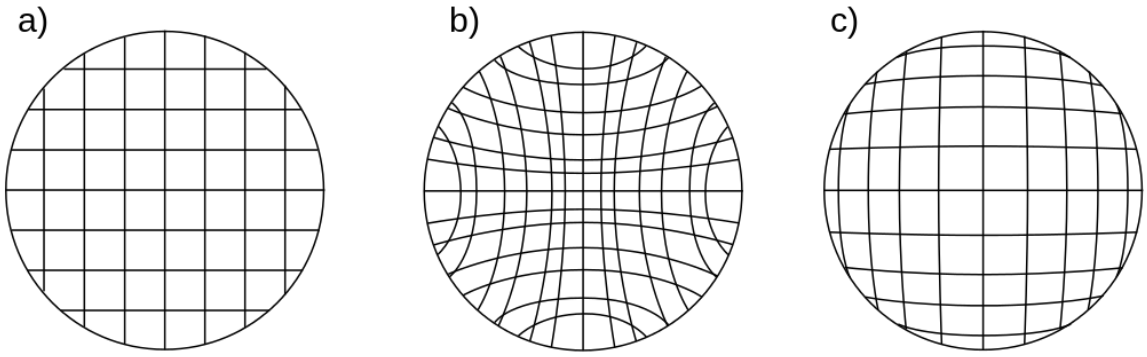
$$d = \left|H\frac{\sin(\theta) + f\cos(\theta)}{-\cos(\theta) + f\sin(\theta)}\right| + 1 \tag{3.13}$$



Obr. 10 – Proces transformace. (**a**) Zdrojový obraz perspektivy; (**b**) Obraz po transformaci; (**c**) obraz po odstranění přebytečných částí; (**d**) přeškálovaný obraz, převzato z [8]

3.1.7. Zkreslení kamery

V předešlých kapitolách jsme uvažovali dokonalou kameru. Reálné optické systémy jsou však nedokonalé a dochází ke geometrickému zkreslení obrazu. Mezi dvě hlavní patří zkreslení soudkovité a poduškovité (viz Obr. 11). Obecně lze říci, že zkreslení je odchylka od přímočaré projekce. Zkreslený obraz poté zavádí další odchylky při užití fotogrammetrických metod.



Obr. 11 – Geometrické zkreslení obrazu a) nezkreslený; b) poduškovité zkreslení; c) soudkovité zkreslení, převzato z [16]

Tato geometrická zkreslení je možné odstranit přenásobením obrazu kompenzační maticí. Kompenzační matice je někdy poskytována výrobcem kamery. Pokud tato není k dispozici, užívá se fotky rovinné šachovnice, z jejíhož zkreslení se algoritmicky dopočte kompenzační matice. Příklad takového postupu lze nalézt v [9] a [10] .

4. Vývojové platformy

V této kapitole shrnujeme současné možnosti na poli vývojových platforem a následně vybereme vhodnou platformu pro realizaci demonstračního modelu, reverzní top-view kamery s predikcí trajektorie.

4.1. Arduino

Arduino je open-source platforma založená na snadno použitelném hardwaru a softwaru. Programová část je založena na programovacím jazyku Arduino (vychází z Wiring) s vlastním Arduino IDE (vývojové prostředí odvozené z Processing) určeným k výuce programování. Arduino vzniklo v Institutu integračního designu v Ivrea jako jednouchý nástroj pro prototypování, zaměřený na studenty bez hlubší znalosti elektroniky a programování. Vývojové desky se postupem času staly velmi oblíbenými a vznikla kolem nich rozsáhlá komunita. [11]

Srdcem desek Arduino je procesor od firmy Atmel, který se liší dle konkrétního modelu vývojové desky. Desky jsou většinou osazovány 8-bitovými procesory s taktem 8-30 MHz. Deska většinou obsahuje USB-ART převodník, skrze který se programuje procesor. Tato platforma se ukázala jako nevhodná, jelikož by bylo problematické připojit kameru a nemožné na málo výkonném procesoru provádět transformaci obrazu v reálném čase.

4.2. Android/IOS

Ačkoliv se přímo nejedná o vývojovou platformu, bylo by možné model realizovat na mobilním telefonu s operačním systémem Android či IOS. Dnešní mobilní telefony mají fotoaparáty dostatečné kvality i dostatečný výkon k transformování obrazu v reálném čase. Jediným problémem je velmi složité programování takové aplikace na mobilním zařízení. Jelikož se nejedná o typicky řešenou problematiku na mobilních zařízeních, není zde velká podpora knihoven vhodných pro náš model. Vzhledem k této nízké podpoře by realizace modelu byla obtížná a přesahující rámec této práce.

4.3. Raspberry pi

Raspberry pi je název pro takzvané single-board počítače (všechny komponenty jsou připájeny k jedné desce) vyráběné firmou Raspberry Pi Foundation. Jedná se o levný počítač řízený operačním systémem Linux s USB a HDMI porty. Desky jsou osazovány GPIO piny (general purpose input/output), které umožňují připojení vstupních, výstupních periferií a ovládání elektronických komponent. Tyto počítače dále obsahují port pro připojení podporovaných kamer a mají dostatečný

výkon procesoru i dostatek operační paměti k tomu, aby zvládly transformovat obraz v reálném čase, a proto se tato platforma jeví jako vhodná pro naši aplikaci.

5. Model pro demonstraci

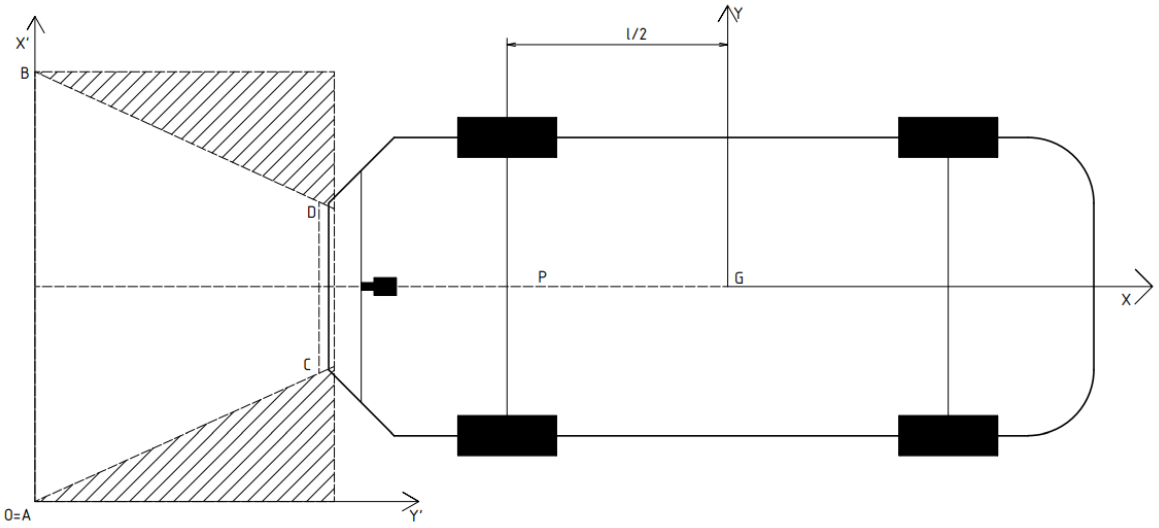
5.1. Rozhraní

Jako vhodný demonstrační model jsme zvolili realizaci reverzní (couvací) kamery s pohledem seshora a vykreslením predikované trajektorie couvání v závislosti na úhlu natočení kol. Uživatelské rozhraní modelu je zobrazeno na Obr. 25.

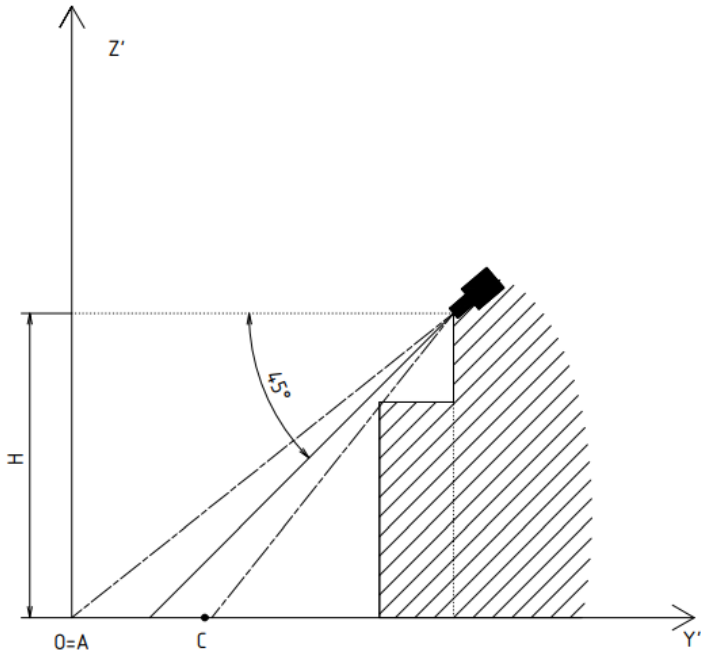
5.2. Geometrie

Vytvořili jsme statický model s umístěním kamery v zadní části automobilu, v měřítku 1:10 k reálné geometrii automobilu. Jako vzor jsme využili geometrii a rozměry Škoda Octavia druhé generace ve verzi kombi (viz Obr. 12). Model simuluje situaci, kdy by tento automobil měl v osvětlení nad SPZ zabudovanou kameru skloněnou pod úhlem 45° směrem k vozovce. Jelikož se jedná jen o demonstrační model, geometrie zádě je silně zjednodušena (viz Obr. 14). Důležité jsou informace o umístění kamery a poloze horní hrany nárazníku, která by měla být v zorném poli kamery, aby uživatel dostával informaci o nejzadnější části vozidla.

Obr. 12 – Rozměry Škoda Octavia kombi II (převzato z [12])

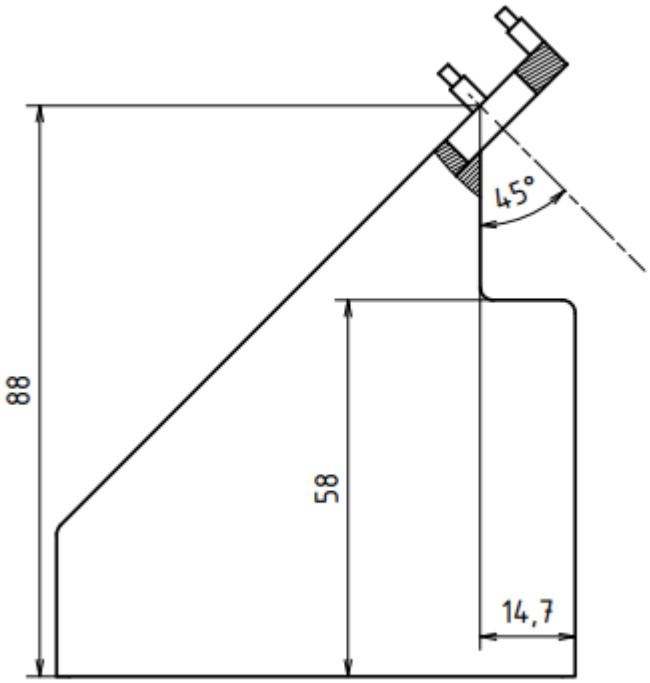


Obr. 13 – Souřadný systém *X', Y'* transformovaného obrazu z kamery a umístění kamery v rovině *X'-Y'*

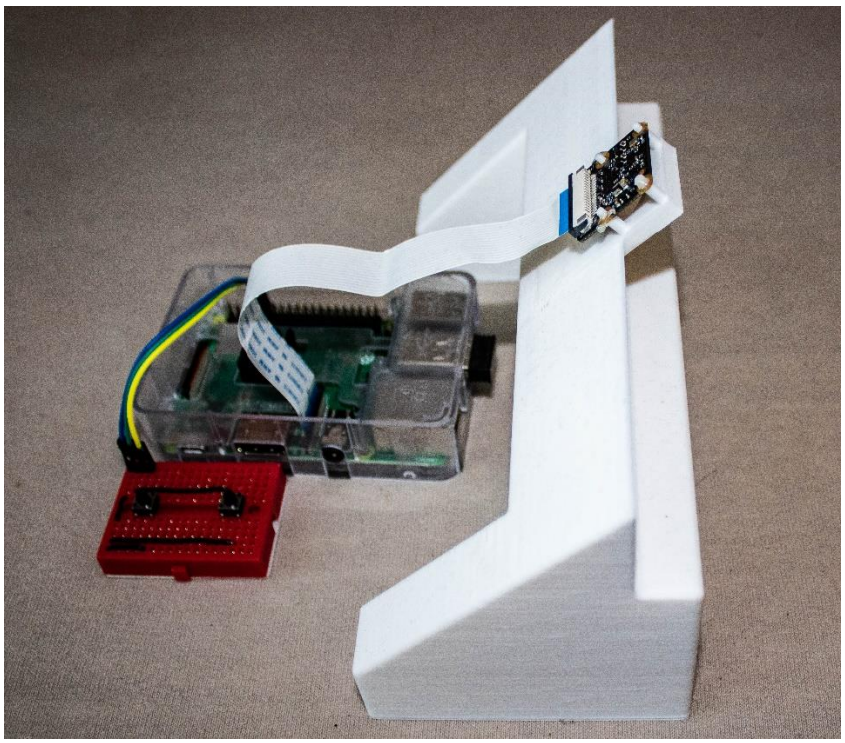


Obr. 14 – Umístění kamery v rovině *Y'-Z'* a ukázka zjednodušení tvaru zadní části automobilu

Po odečtení reálných rozměrů vozidla a přepočtu 1:10 jsme navrhli držák kamery se zjednodušeným tvarem hrany nárazníku, jak ukazuje Obr. 15. Poté jsme tento držák vyrobili za pomoci 3D tisku.



Obr. 15 – Rozměry a geometrie modelu zadní části automobilu s uchycením kamery (rozměry v mm)

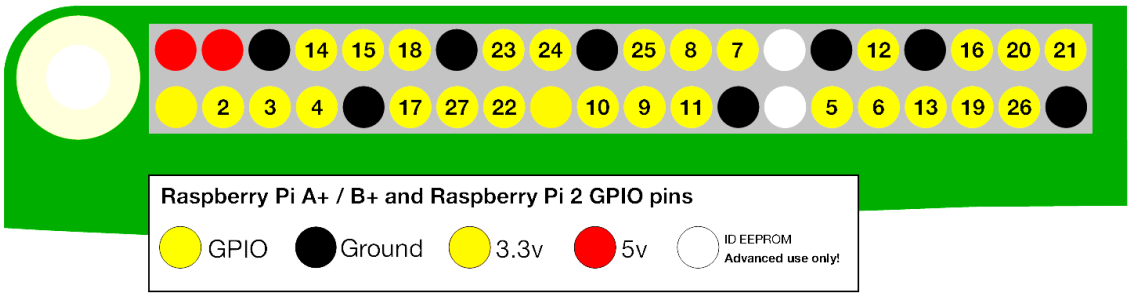


Obr. 16 – Držák kamery vyrobený pomocí 3D tisku

5.3. Použitý hardware

5.3.1. Raspberry Pi 3 Model B+

Jak bylo již zmíněno v kapitole 4.3, jedná se o single-board PC. Tento konkrétní model disponuje 1GB SDRAM a procesorem Cortex-A53(ARMv8) 64-bit na taktu 1.4 GHz. Na zařízení je také wifi modul, 4xUSB, HDMI, LAN port, CSI (kamera port) a DSI (display port). Dále obsahuje 40-pinový GPIO IDC konektor s rozložením (viz Obr. 17).



Obr. 17 – Rozložení pinů v IDC konektoru na Raspberry PI 3 B+ (převzato z [13])

5.3.2. Raspberry Pi NoIR Camera V2

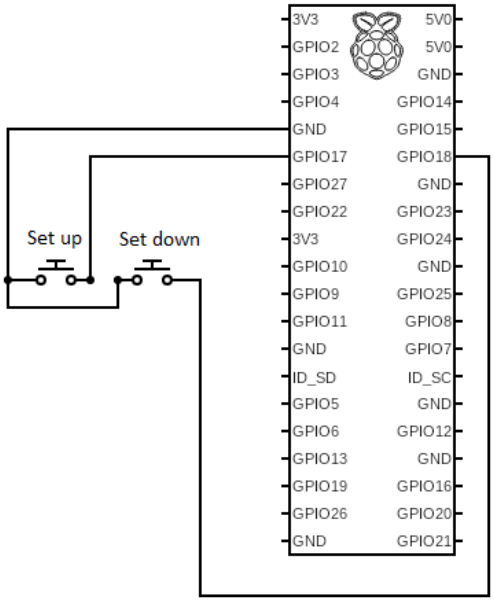
Jako zdroj obrazu jsme použili Raspberry Pi NoIR Camera V2, kterou jsme připojili přes CSI port. Přesné specifikace kamery jsou uvedeny v Tabulka 1.

Tabulka 1 Technické parametry Raspberry Pi NoIR Camera V2

Technické parametry:	
Typ senzoru	Sony IMX219PQ Color 8-megapixel
Velikost senzoru	3.674 x 2.760 mm
Rozlišení	3280 x 2464 (Aktivních pixlů) 3296 x 2512 (Celkových pixlů)
Velikost pixlu	1.12 x 1.12 um
Ohnisková vzdálenost	f=3.04 mm
Zorné úhly	62.2° x 48.8°
Video módy:	
Rozlišení videa do 60fps 4:3	640x480 px
Rozměry:	
Velikost desky	25 x 23.86 x 9 mm
Montážní otvory	4x D=2.20 mm, rozteč 12.5 x 21.0 mm

5.3.3. Ostatní hardware

Jako vstup pro nastavení úhlu natočení kol jsme zvolili dva mikrospínače typu TACT, kterými uživatel může konfigurovat úhel natočení kol (tlačítka Set up a Set down). Zapojení k desce Raspberry PI (viz kapitola 5.3.1) je zobrazeno na Obr. 18. Ukázka námi vytvořeného kódu ovládajícího GPIO vstup je zobrazena na Obr. 19.



Obr. 18 – Zapojení spínačů k GPIO (vytvořeno v Circuit Diagram)

```
def _get_tilt(self):

    #Writing to a local variable last state of tilt
    tilt = self.tilt

    #Get state of GPIO
    input_state0 = GPIO.input(17)
    input_state1 = GPIO.input(18)

    #Change the tilt
    if input_state0 == False:
       tilt=tilt+1
    if input_state1 == False:
       tilt=tilt-1

    # Limitation of maximum angle
    if tilt > 30 or tilt< -30:
       return tilt

    # Writing to a global variable new state of tilt
    self.tilt=tilt
    return tilt
```

Obr. 19 Ukázka kódu pracujícího s GPIO

5.4. Použitý software

5.4.1. Operační systém

Jako operační systém řídicí Raspberry Pi jsme zvolili Raspbian stretch with desktop and recommended software [13]. Jedná se o odlehčenou verzi Linuxového systému Debian s nainstalovaným uživatelským rozhraním a doporučeným softwarem nejen pro výuku programování a automatizace. Bitovou kopii operačního systému lze stáhnou z [13] a poté nahrát na Micro SD kartu (například přes aplikaci balenaEtcher), ze které Raspberry Pi provede při zapnutí boot.

5.4.2. Programovací jazyk Python

K samotnému programování jsme se rozhodli využít programovací jazyk Python 3. Jedná se o multiplatformní vysokoúrovňový skriptovací programovací jazyk, který nabízí dynamickou kontrolu datových tipů a podporuje různá programovací paradigmata (jedná se o hybridní jazyk), včetně objektově orientovaného, imperativního, procedurálního a funkcionálního. Jedná se o open source projekt. V současné době se používají verze 2.x (starší verze s rozsáhlejší základnou knihoven) a 3.x (novější ucelenější verze nekompatibilní s verzí 2.x). Pro naší aplikaci jsme využili novější verzi 3.x. K hlavním výhodám jazyku Python patří jeho jednoduchost z hlediska učení, což je dáno jeho čistotou a jednoduchostí syntaxe. Výkonem patří Python k pomalejším jazykům. Samotný výkon aplikací vytvořených v Pythonu je však dobrý, jelikož výkonově kritické funkce v knihovnách jsou psané v jazyce C, se kterým je schopný Python velmi efektivně spolupracovat. Informace převzaty z [14].

5.4.3. Knihovna OpenCV

OpenCV (Open source computer vision) je knihovna funkcí zaměřených především na zpracování informací z videa v reálném čase (real-time computer vision) a strojové učení. Jedná se o nenativní knihovnu Pythonu, proto je potřeba ji doinstalovat. Instalace na Raspberry Pi dle oficiálního zdroje [10] je poměrně zdlouhavý a složitý proces. Proto jsme použili tento postup popsaný v [15]. Z této knihovny jsme především použili funkce k získání transformační matice, provedení transformace a převedení formátu obrazu. Mimo jiné také používáme funkci z této knihovny k samotnému získání obrazu z webkamery. Na Raspberry Pi nemá při spuštění systému Python oprávnění pracovat s kamerou, proto pro správnou funkci je potřeba udělit oprávnění v terminálu příkazem: „*sudo modprobe bcm2835-v4l2*“.

5.4.4. Knihovna Tkinter

Tkinter je nativní knihovnou Pythonu pro vytváření uživatelského rozhraní GUI, a to především uživatelských oken, k čemuž tuto knihovnu také používáme.

5.4.5. Knihovna NumPy

NumPy Python knihovna je základním balíčkem funkcí pro vědecké výpočty. Mezi její hlavní využití patří vytváření n-rozměrných objektů, nástroje pro implementaci C/C++ kódu, výpočty lineární algebry a generování náhodných čísel. Z této knihovny jsme použily funkci k výpočtu tangenty, odmocniny a zavedení datového typu float32.

5.4.6. Knihovna RPi.GPIO

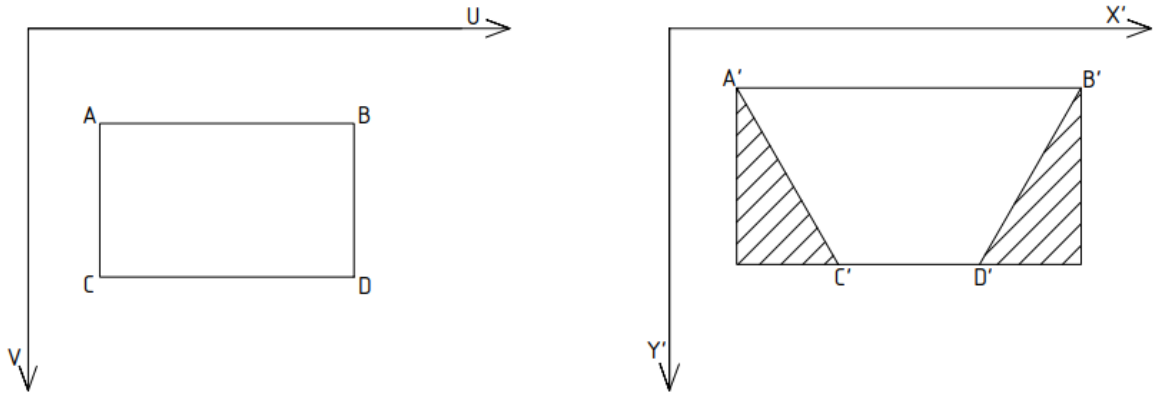
RPi.GPIO je knihovna, která Pythonu umožňuje ovládat GPIO piny na Raspberry Pi. Tuto knihovnu používáme k ovládání mikrospínačů, viz kapitola 5.3.3. Pro toto zapojení spínačů je důležité nastavení, které zařídí, že při čtení vstupu GPIO pinu je v Raspberry Pi softwarově zařazen tzv. pull-up rezistor. Toto nastavení zapsané například pro pin 17 má následující tvar: „*GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)*“.

5.4.7. Knihovna PIL

PIL (Pillow), Python Imaging Library („zobrazovací knihovna“) je nenativní knihovnou Pythonu. Tuto knihovnu používáme jako přemostění mezi výstupem obrazu z knihovny OpenCV a vstupem funkce canvas.create.image() z knihovny Tkinter.

5.5. Získání transformační matice

Transformační matici lze odvodit z geometrie a vlastností kamery, jak je popsáno v kapitole 3. Vzhledem k tomu, že náš model má geometrii s $\gamma = 180$ (viz kapitola 3.1.5) a obraz zachycený kamerou je pouhou lineární perspektivní projekcí prostoru za automobilem, mohli jsme využít výrazně přímočařejší metodu k získání transformační matice, která používá funkci *getPerspectiveTransform()* z knihovny OpenCV k získání transformační perspektivní matice. Tato funkce má jako vstup dvě pole a jako výstup samotnou 3x3 transformační perspektivní matici. Ve vytvořeném zdrojovém kódu je funkce volána následovně: *cv2.getPerspectiveTransform(src, dst),* kde *src* je pole souřadnic vrcholů ve zdrojovém obrazu a *dst* je pole souřadnic odpovídajících čtyřúhelníku v cílovém obrazu. Velmi důležité je pořadí souřadnic bodů v polích *src* a *dst*. Pokud bod A má souřadnice $A=(u_A, v_A)$, pak pole *src* má tvar *src=[A, B, C, D]* a obdobně *dsc=[A', B', C',D']*, viz Obr. 20.

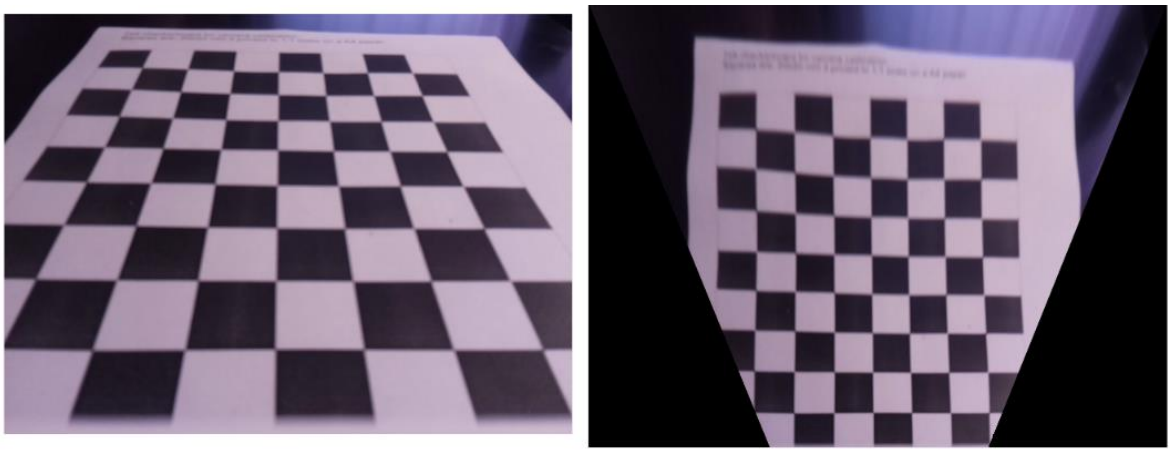


Obr. 20 – Perspektivní transformace mezi systémy *U-V* a *X'-Y'*

V našem případě odečteme souřadnice pixlů obdélníku na výstupním obraze z kamery pro levý horní roh obrazu, pravý horní roh obrazu, levého a pravého bodu za hranou nárazníku (místo, kde kamera nepromítá nárazník, ale vozovku). Kamera má rozlišení 640x480 px. Po odečtení můžeme zapsat *src = np.float32([[0, 0], [640, 0], [0, 450], [640, 450]]).* Pro odečtení souřadnic pixlů v obraze a kalibraci transformace jsme užili námi vytvořený skript *get_transform.py* (viz příloha na CD).

V rovině *X'-Y'* jsme zjistili vzájemnou vzdálenost bodů A*', B', C', D'* v milimetrech a bod A umístili do počátku souřadného systému. Ze znalosti, že úsečka $\overline{CD}$ přejde v úsečku $\overline{C'D'}$ a úvahy o tom, že zachováme počet pixlů této úsečky v obraze i vzoru, jsme získali škálování 1 mm≈2px. Odtud poté získáme *dst = np.float32([[0, 0], [652, 0], [202, 495], [452, 495]]).*

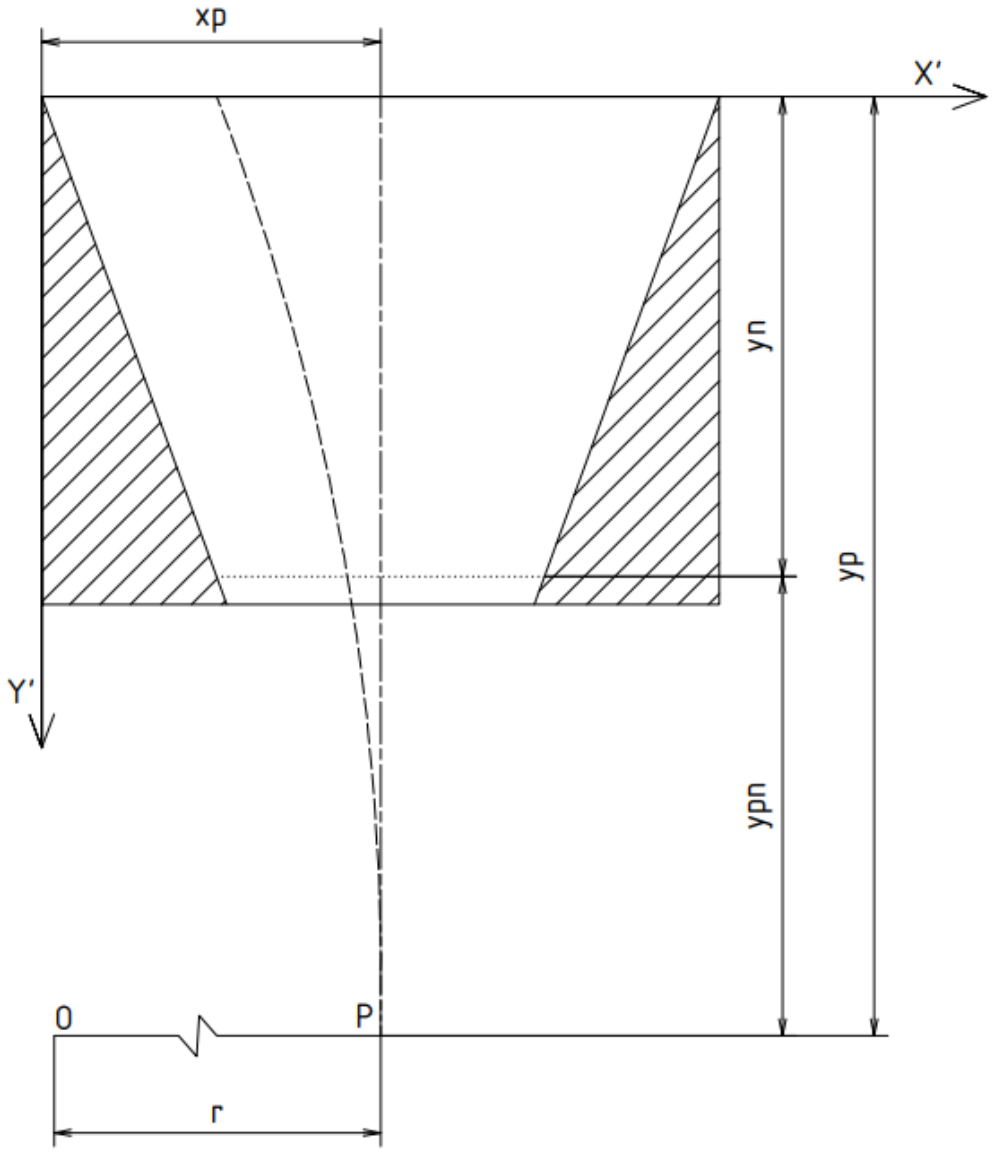
O samotnou transformaci obrazu se poté stará funkce *warpPerspective()* z knihovny OpenCV. Ukázka výsledku námi provedené transformace je zobrazena na Obr. 21.



Obr. 21 – Ukázka transformace, v levé části výstupní obraz z kamery, v pravé části obraz po provedení perspektivní transformace

5.6. Vykreslení trajektorie

Jak je odvozeno v kapitole 2.2, predikovaná trajektorie vozidla má tvar kružnice se středem v bodě O, viz Obr. 22. Pro nás je důležité znát polohu bodu P v souřadném systému. Protože je kamera umístěna uprostřed automobilu, ze symetrie vyplývá, že jeho x-ová souřadnice *xp* je rovna polovině šířky transformovaného obrazu. Složku *yp* spočteme ze znalosti vzdálenosti bodu D‘ od počátku systému a od kraje nárazníku (na Obr. 22 je hrana nárazníku zobrazena tečkovanou čarou) a ze vzdálenosti kraje nárazníku od zadní nápravy. Opět přepočteme rozměry reálného vozidla na rozměry modelu a poté na pixly systému X‘-Y‘ (1 mm v modelu ≈ 2px v transformovaném obrazu). Velikost poloměru r je pouze funkcí úhlu natočení kol φ.



Obr. 22 – Trajektorie v souřadném systému Y’-X’

Získané hodnoty dosadíme do obecné rovnice kružnice (5.1) a postupujeme jako v kapitole 2.2.:

$$(x-a)^2+(y-b)^2=r^2 \tag{5.1}$$

kde pro náš model spočteme:

$$a=326-\frac{530}{\tan\varphi} \tag{5.2}$$

$$b=759 \tag{5.3}$$

$$r=\frac{538}{\tan\varphi} \tag{5.4}$$

Pro bod se známou hodnotou souřadnice y dopočteme z rovnice (5.1) hodnotu souřadnice x, jak je popsáno v ukázce zdrojového kódu na Obr. 23 a v rovnici (5.5). Opět je zde důležité ošetřit dělení nulou a znaménko $\mp$ v rovnici (5.5), pro hodnoty úhlu natočení kol φ (viz kap 2.2.).

```
"""
   Calculates coordinate x
"""
def _coordinate_x(self):

      #Division by zero treatment
      if self.tilt == 0:
         x=326
         return x

      #Calculating a point on a circle
      a=326-(538/np.tan(self.tilt*(np.pi/180)))
      b=759
      r=538/np.tan(self.tilt*(np.pi/180))
      if self.tilt > 0:
         x=a+np.sqrt(r**2+(self.y-b)**2)
      if self.tilt < 0:
         x=a-np.sqrt(r**2+(self.y-b)**2)
      return x
```

Obr. 23 – Ukázka kódu pro výpočet x-ové souřadnice trajektorie

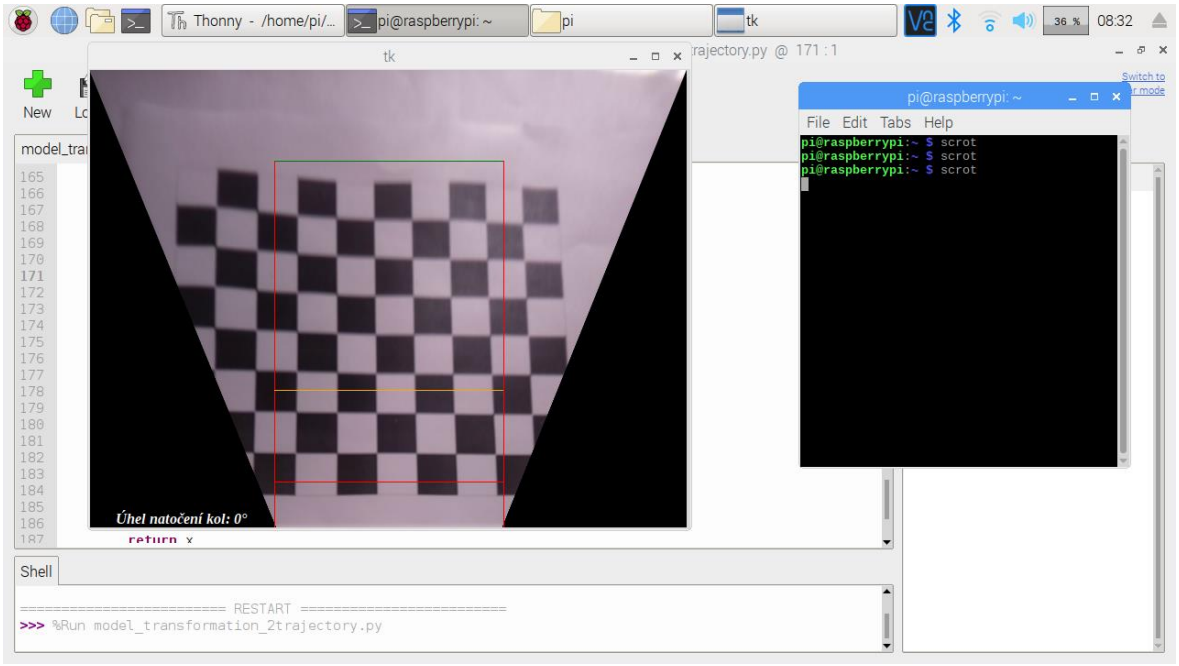
$$x(y, \varphi) = a \mp \sqrt{r^2 - (y - b)^2} \tag{5.5}$$

K vykreslení trajektorie používáme funkci *create_line(),* z knihovny Tkinter, která má jako vstup pole souřadnic bodů. Toto pole načítáme *for*-cyklem, viz Obr. 24.

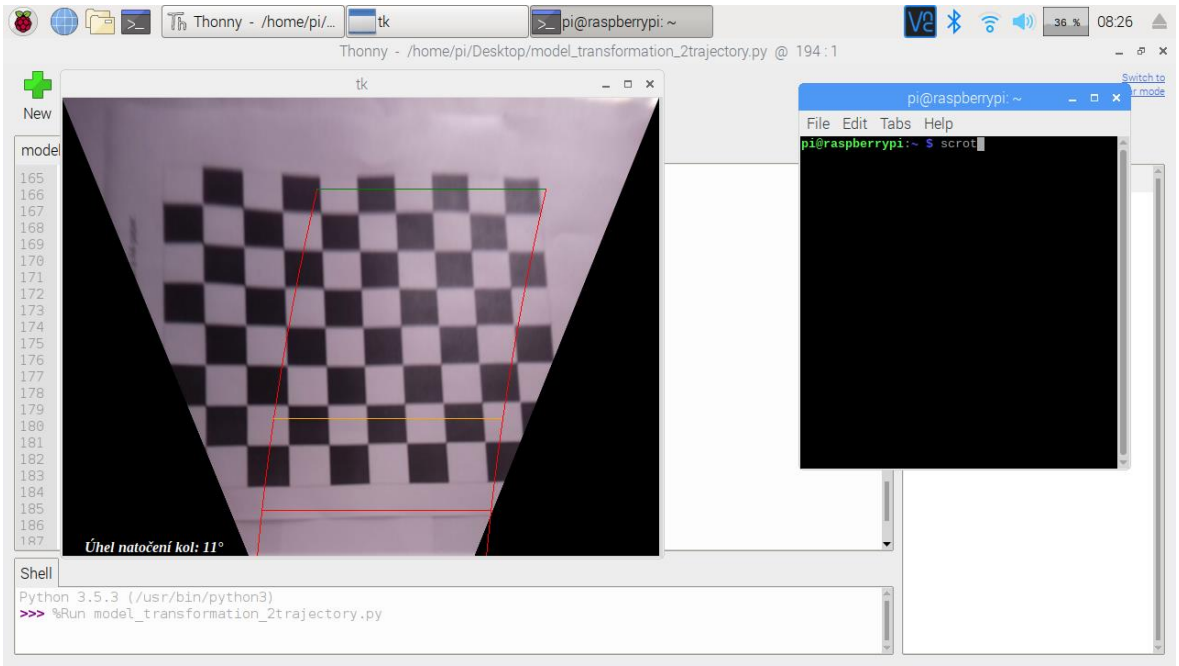
```
    """
      Calculates new trajectory lines
    """
    def _get_trajectory(self):

        #Get a new tilt position
        self._get_tilt()

        #Create new empty array (list)
        trajectory=[]

        #Cycle filling array for right line
        for self.y in range (100, 501):
          trajectory.append(int(round(self._coordinate_x()))+125)
          trajectory.append(self.y)

        #Writing to a global variable new state of tilt
        self.trajectory1 = trajectory

        #Create new empty array (list)
        trajectory=[]

        #Cycle filling array for left line
        for self.y in range (100, 501):
          trajectory.append(int(round(self._coordinate_x()))-125)
          trajectory.append(self.y)

        #Writing new trajectory to a global variable
        self.trajectory2 = trajectory
        return
```

Obr. 24 – Ukázka kódu vytvářejícího pole pro vykreslení trajektorie

V této kapitole jsme se doposud zabývali trajektorií bodu *P* (viz Obr. 22). Pro uživatele asistenčního systému je ovšem mnohem zajímavější informace o budoucí trajektorii krajních bodů vozidla. Proto v našem zjednodušeném modelu vykreslujeme dvě posunuté trajektorie bodu *P.* Jedna trajektorie je posunuta o půl šíře vozidla vlevo a druhá vpravo (viz Obr. 25 a Obr. 26). V případě nulového natočení kol na Obr. 25 lze dobře vidět, že výsledné zkreslení šachovnice je větší s rostoucí vzdáleností od zádi vozidla. Toto je přesně v duchu způsobu definice transformace diskutované v kapitole 3.1.5. Celkově však považujeme dosažený výsledek za přijatelný. Celý skript našeho modelu se nachází v příloze na CD jako soubor *model_transformation_2trajectory.py*.

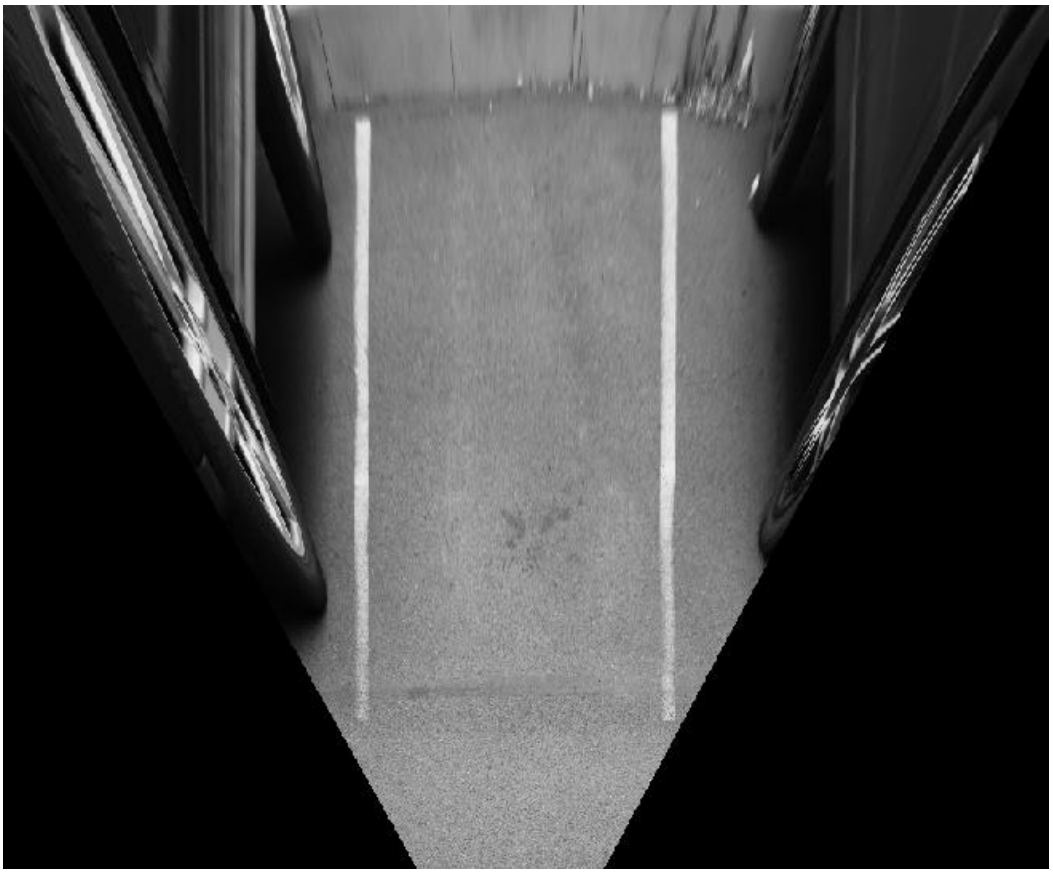


Obr. 25 – Ukázka výstupu modelu s transformací obrazu a vykreslením trajektorie pro natočení kol $\varphi = 0°$



Obr. 26 – Ukázka výstupu modelu s transformací obrazu a vykreslením trajektorie pro natočení kol $\varphi = 11°$

Obr. 27 – Vstupní obraz parkoviště



Obr. 28 – Výstupní obraz parkoviště

Závěr

Cílem práce bylo seznámit se se stávajícími asistenčními a kamerovými systémy v automobilech, připravit simulační model pro predikci trajektorie, vybrat asistenční systém, provést jeho zjednodušení pro demonstrační model asistenčního systému a zvolit vhodnou platformu, na které se poté model bude realizovat.

V první části práce byla stručně diskutována historie a současný stav na poli asistenčních systémů pro couvání a parkování, od parkovacích senzorů po plně autonomní systémy.

Ve druhé části práce byl popsán zjednodušený kinematický model automobilu a analytické odvození trajektorie vozidla v závislosti na geometrii vozidla a úhlu natočení kol. Tento zjednodušený kinematický model byl následně využit při realizaci zvoleného demonstračního modelu.

Třetí část práce se teoreticky zabývala procesem generování vrchního pohledu tzv. Bird's eye view transformací a jednotlivými dílčími transformacemi a projekcemi, ze kterých se tento proces skládá. Dále byl zmíněn model dírkové komory a zkreslení reálných kamer.

Čtvrtá část práce se zabývala současnými možnostmi na poli vývojových platforem a následně výběrem vhodné platformy pro realizaci demonstračního modelu reverzní kamery s vrchním pohledem a s predikcí trajektorie.

V rámci páté části práce byl zvolen a realizován zjednodušený statický model reverzní kamery s Bird's eye view transformací obrazu a predikcí trajektorie. Byl použit hardware Raspberry Pi 3 Model B+. Zjednodušená geometrie modelu byla odvozena od reálného automobilu v měřítku 1:10. Model byl vytvořen pomocí 3D tisku. Jako programovací jazyk byl použit Python 3 a z knihoven hlavně OpenCV, která obsahuje mimo jiné i funkce k provedení lineární perspektivní transformace obrazu kamery. Výsledný výstup modelu byl otestován zobrazením horního pohledu na šachovnici s vykreslením trajektorie závislé na úhlu nastavení kol. Zkreslení šachovnice se zvětšuje s rostoucí vzdáleností od zádi vozidla, což je přesně v duchu způsobu definice transformace a v důsledku nekorigovaného zkreslení použité kamery. Celkově však lze považovat dosažený výsledek za velmi přijatelný.

Náměty pro dodatečné vylepšení případně rozpracování dané problematiky jsou následující: 1) Optimalizace umístění kamery na vozidle, její sklon a parametry; 2) Přesnější vykreslení predikce trajektorie jako obálky bodů vozidla a ne jako posunuté trajektorie středu zadní nápravy; 3) Přidání spolupráce s parkovacími senzory, či dokonce s automatickým rozpoznáváním objektů a predikce kolize s nimi ve snímaném obraze.

Literatura

[1] T. Dusil, „auto.cz,“ 25. 07. 2015. [Online]. Available: https://www.auto.cz/technika-elektronika-v-autech-postupuje-a-neceka-88380. [Přístup získán 05. 05. 2019].

[2] M. TOMOZAWA a N. IMAI, „PARKING ASSISTANCE DEVICE“. Patent EP3135564 (B1), 03. 04. 2019.

[3] N. Andreev, „Confused.com,“ 13. 06. 2018. [Online]. Available: https://www.confused.com/on-the-road/gadgets-tech/parking-technology-brief-history. [Přístup získán 11. 05. 2019].

[4] QuANz459f, „Quanz auto body,“ 12. 06. 2018. [Online]. Available: https://www.quanzautocare.com/parking-sensor/. [Přístup získán 11. 05. 2019].

[5] „Šmucler magazín,“ [Online]. Available: https://www.smucler.cz/blog/parkpilot/. [Přístup získán 11. 05. 2019].

[6] M. Sekerka, Systém pro plánování složitějších parkovacích manévrů, Praha: CVUT, Fakulta strojní, diplomová práce, 2018.

[7] M. Venkatesh a P. Vijayakumar, „Transformation Technique,“ *International Journal of Scientific & Engineering Research,* pp. Volume 3, Issue 5, May 2012.

[8] L. Chien-Chuan a W. Ming-Shi, „A Vision Based Top-View Transformation Model for a Vehicle Parking Assistant,“ Sensors[online], 2012.

[9] N. Falaleev, „Self-Driving Cars Lab,“ 05. 06. 2017. [Online]. Available: https://nikolasent.github.io/. [Přístup získán 18. 06. 2019].

[10] „OpenCV documentation,“ OpenCV, [Online]. Available: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html. [Přístup získán 18. 06. 2019].

[11] „ARDUINO,“ Arduino, [Online]. Available: https://www.arduino.cc/en/Guide/Introduction. [Přístup získán 18. 06. 2019].

[12] „MOTORINFO,“ MOTORINFO, 29. 04. 2013. [Online]. Available: http://www.motorinfo.cz/nova-skoda-octavia-combi-prijizdi-s-pohonem-prednich-i-vsech-ctyr-kol.html. [Přístup získán 18. 06. 2019].

[13] „Raspberrypi,“ Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.org/. [Přístup získán 19. 06. 2019].

[14] W. contributors, „Python (programming language),“ Wikipedia, The Free Encyclopedia., 18. 05. 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Python_(programming_language)&oldid=902393741. [Přístup získán 20. 06. 2019].

[15] B. Nuttall, „piwheels,“ piwheels, 27. 08. 2018. [Online]. Available: https://blog.piwheels.org/new-opencv-builds/. [Přístup získán 20 06. 2019].

[16] M. Králová, „Techmania Science Center,“ Eudoportál, [Online]. Available: https://edu.techmania.cz/cs/encyklopedie/fyzika/svetlo/opticke-zobrazovani/vady-optickych-soustav. [Přístup získán 18. 06. 2019].