Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Aplikácia na meranie dĺžky a šírky chodidla

Bakalárska práca

2017
Izabela Dobšovičová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Aplikácia na meranie dĺžky a šírky chodidla

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	RNDr. Zuzana Berger Haladová, PhD.

Bratislava, 2017
Izabela Dobšovičová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Izabela Dobšovičová
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: aplikovaná informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Aplikácia na meranie dĺžky a šírky chodidla z fotografie
Foot measuring application

Anotácia: Aplikácia pre Android, ktorá z jednej fotografie z kalibrovanej kamery, obsahujúcej známy objekt a obrys chodidla/chodidlo zistí jeho dĺžku a šírku. 1. Naštudovanie problematiky "ground calibration" a metód počítačového videnia na extrakciu kontúr 2. Špecifikácia 3. Implementácia Android apliikácie

Vedúci: RNDr. Zuzana Berger Haladová, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 11.09.2017

Dátum schválenia: 23.10.2017 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné prehlásenie: Čestne prehlasujem, že som túto bakalársku prácu vypracovala samostatne s použitím uvedenej literatúry.

V Bratislave dňa: .....

.....
Izabela Dobšovičová

Poďakovanie: Ďakujem mojej rodine a priateľom za stále pretrvávajúcu podporu a trpezlivosť počas celej tvorby tejto bakalárskej práce. Najviac vďačím však mojej školiteľke RNDr. Zuzane Berger Haladovej, PhD., za jej zapálenie pre vec, návrhy a postrehy k práci, stále dobrú náladu a podporu a za moje rozvíjanie a vedie v tejto oblasti. Veľa som sa naučila a to veľmi fascinujúcou formou. Ďakujem.

Abstrakt

Cieľom tejto bakalárskej práce je vytvorenie aplikácie pre platformu Android, ktorá dokáže z jednej snímky z kalibrovanej alebo nekalibrovanej kamery, obsahujúcej obrys chodidla na papieri A4, zistiť dĺžku a šírku chodidla. Je tu opísaná potrebná teória na hlbšie pochopenie danej problematiky. Ďalej je opísaný návrh aplikácie, podľa ktorého sme vytvárali našu aplikáciu a aj samotný vývoj aplikácie. Čitateľ je postupne oboznámený s postupmi spracovania obrazu a potrebnými knižnicami, ktoré sme využívali počas vývoja našej aplikácie.

Kľúčové slová: meranie veľkosti chodidla, Android aplikácia, spracovanie obrazu

Abstract

The goal of this bachelor thesis is creating an Android application for measuring length and width of foot. It can be done using one image from calibrated or noncalibrated camera, containig the outline of a foot drawn on an A4 sized paper. This thesis contains the description of the theory needed for deeper understanding of the problem. Specification and implementation itself is described in the thesis as well. A reader will be introduced to image processing techniques, used during creating the application.

Keywords: foot size measure, Android application, image processing

Obsah

Zoznam obrázkov

Úvod

Každý z nás už zažil situáciu, kedy sme si skúšali topánky u iného predajcu ako sme boli zvyknutí, alebo sme si objednávali topánky z internetu a zistili sme, že nie je veľkosť ako veľkosť. Najistejším spôsobom, ako čo najpresnejšie zistiť svoju veľkosť topánky je preto zistiť svoju veľkosť chodidla v centimetroch a podľa toho sa ďalej orientovať.

Zaujímavým spôsobom zisťovania veľkosti chodidla by bolo pomocou svojho smartfónu. Preto sme sa rozhodli vytvoriť takúto aplikáciu pre Android. Na začiatku si ale musíme uvedomiť, že keď snímame obrázok z rôznej výšky, veľkosť chodila v pixeloch sa pri inej výške stále mení. Taktiež sa nám chodidlo rôzne perspektívne deformuje keď snímame chodilo z rôznych uhlov. Preto potrebujeme mať v obraze určitý predmet, ktorého veľkosť je štandardizovaná a známa. Tento predmet musí ležať v rovnakej rovine ako merané chodidlo, aby sme vedeli zadefinovať rovinu v 3D priestore. Vďaka tomuto objektu budeme vedieť určiť aká je naša hľadaná veľkosť chodidla.

V prvej časti tejto práce sú vysvetlené teoretické poznatky potrebné na pochopenie danej problematiky a na tvorbu našej aplikácie. Tiež sú v nej spomenuté technológie, ktoré sme využívali počas vývoja aplikácie. Sú tu opísané aj riešenia, či už z predchádzajúcich bakalárskych a diplomových prác, alebo aplikácií z internetového obchodu Google Play, ktoré aspoň z časti súvisia s našou problematikou.

V nasledujúcej kapitole sme predložili čitateľovi návrh na riešenie nášho problému, podľa ktorého sme postupovali pri tvorbe aplikácie. Samotná tvorba je opísaná v nasledovnej časti *implementácia*. Nachádzajú sa v nej opisy postupov, ktoré boli úspešné alebo nás naviedli na správny smer.

V posledných kapitolách je zhrnutie práce a ukázané namerané výsledky našou aplikáciou a ich porovnanie so skutočnými hodnotami.

Kapitola 1

Prehľad problematiky

V tejto kapitole je opísaný základný prehľad vedomostí, ktoré sú potrebné na realizáciu práce. Taktiež tu nájdeme odkazy na staršie bakalárske práce, ktoré sa aspoň z časti týkali danej problematiky. Nakoniec si v tejto kapitole ešte ukážeme prehľad technológií využívaných pri riešení danej problematiky.

1.1 Potrebná teória

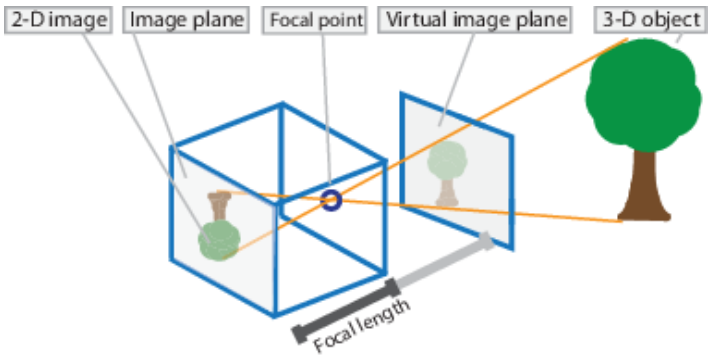
Aby sme vedeli odmerať, dĺžku a šírku chodidla z fotografie, potrebujeme vedieť zachytávať správny obraz. V súčasnosti nám na to slúži veľké množstvo zariadení, či už filmové fotoaparáty, digitálne fotoaparáty alebo aj všeobecne rozšírené kamery v každom smartfóne. Všetky tieto zariadenia fungujú podľa základného modelu fotografického zariadenia, a to dierkovej kamery. Lenže pridaním objektívu je zachytávaný obraz často skreslený a je ho treba ešte upraviť. Na odstránenie tohto skreslenia nám slúži kalibrácia kamery. V tejto časti si opíšeme aj princípy:

- dierkovej kamery
- kalibrácie kamery
- Houghovej transformácie
- homografie

1.1.1 Dierková kamera

Pre správne zachytávanie obrazu je dôležité si uvedomiť, ako takéto zachytávanie funguje. Ukážeme si to na najjednoduchšom princípe a to na princípe fungovania dierkovej kamery, alebo tiež nazývanej štrbinová kamera (pinhole camera), ako je to uvedené v [9]. Dierková kamera, alebo dierková komora (lat. camera = izba, komora), je predchodca fotoaparátov a filmovej kamery. Je to tmavá škatuľka, alebo krabica ktorá neobsahuje

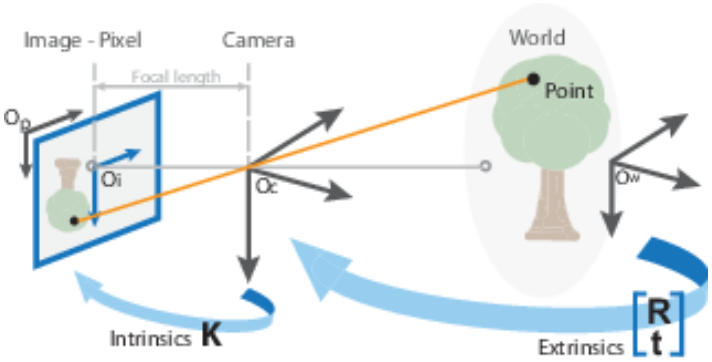
žiaden objektív len dierku. Vnútri tejto škatuľky sa na zadnej stene nachádza film. Cez malú dierku na škatuľke dopadá na film svetlo z vonkajšieho prostredia. Takto získame otočený obraz sveta, ktorý snímame.



Obr. 1.1: 3D objekt premietaný na snímaciu plochu resp. film. [9]

Ohnisková vzdialenosť (focal lenght na obr. 1.1) v dierkovej kamere je dôležitá pre veľkosť snímaného sveta. Pokiaľ je ohnisková vzdialenosť veľmi malá (plocha na ktorú sa sníma obraz je blízko pri dierke), nasnímame väčšiu časť krajiny, ako keď je ohnisková vzdialenosť veľká. Vtedy nasnímame len užšiu časť krajiny.

Pre kameru vieme určiť viacero parametrov. Rozdeľujeme ich na vnútorné parametre a vonkajšie parametre. Vnútorné parametre reprezentujú súradnice optického centra a ohniskovú vzdialenosť. Vonkajšie parametre zase reprezentujú pozíciu kamery v reálnom 3D prostredí.



Obr. 1.2: Vonkajšie a vnútorné parametre kamery. [9]

Ako sme si vyššie spomínali, vnútorné parametre sú ohnisková vzdialenosť a súradnice optického stredu (centra). Tieto informácie obsahuje matica vnútorných parametrov kamery. Ohniskovú vzdialenosť kamery budeme označovať ako f_x a f_y. Optické centrum budeme označovať ako c_x a c_y. Celú maticu vnútorných parametrov budeme označovať ako K.

Vonkajšie parametre používame na transformáciu zobrazenia z 3D svetového súradnicového systému na 3D kamerového súradnicového systému. Sú to parametre R a t. R

je rotácia a t posunutie - translácia súradnicového systému kamery oproti svetovému súradnicovému systému.

Vnútorné a vonkajšie paramtre tvoria kamerovú maticu. Tá potom vyzerá nasledovne, podľa [9]:

$$P = K \begin{bmatrix} R & t \end{bmatrix}$$

Ak rozpíšeme vnútorné parametre, kamerová matica bude potom vyzerať:

$$P = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} R & t \end{bmatrix}$$

Kamerovú maticu využívame, keď transformujeme 3D body v scéne na 2D body v obrázku. To robíme nasledovne:

$$w \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

kde w predstavuje faktor zoškálovania, x, y reprezentujú súradnice bodu v 2D obrázku, X, Y, Z reprezentujú súradnice bodu v snímanom 3D svete. P predstavuje kamerovú maticu.

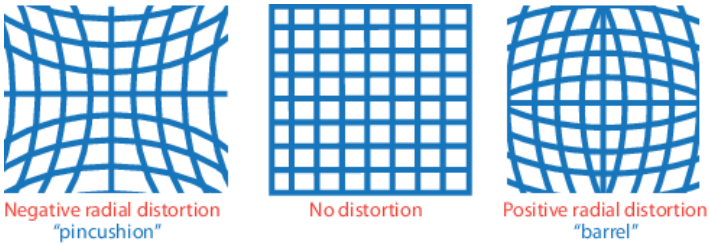
1.1.2 Kalibrácia

Pridaním objektívu sa však snímaný obraz viac či menej skresľuje, ako to môžeme vidieť napríklad na obr.1.3.

Obr. 1.3: Príklad skreslenia a využitia kalibrácie. [9]

Prečo je nám potrebné nakalibrovať kameru? Zoberme si neupravenú fotografiu s určitým skreslením. Keby sme chceli z takéhoto obrázku vyčítať veľkosti reálneho sveta, ako milimetre, centimetre atď. nepodarilo by sa nám to aj keby sme mali k dispozícií známy objekt na snímke. Nameraný počet pixelov v predmete na kraji fotografie, by bol totižto rozdielny od nameraného počtu pixelov v tom istom predmete v strede fotografie. Teda pri skreslenom obrázku nevieme jasne určiť veľkosti v reálnom svete. Z nakalibrovanej kamery môžeme tiež určiť hĺbku pri použití stereoskopie (dvoch kamier). Kalibráciou kamery taktiež dostaneme informácie o vzdialenosti objektívu od snímacích senzorov (ohnisková vzdialenosť) a informáciu o optickom strede kamery. Ďalej dostaneme takzvaný scew coeficient, čiže koeficient zošikmenia. Tento koeficient vyjadruje, či sú os x a os y v obrázku na seba kolmé. Ak tieto osi na seba kolmé sú, tak je tento koeficient nula. Pri novších senzoroch sa predpokladá, že tento koeficient je vždy nula, preto v našej matici nefiguruje.

Skreslenia poznáme rôzne. Jendým z nich je radiálne skreslenie, ktoré obrázok deformuje nasledovne:



Obr. 1.4: Modely radiálneho skreslenia kamery. [9]

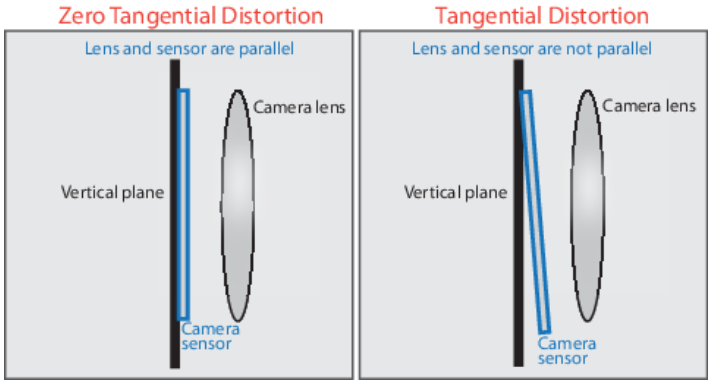
Ako je to uvedené v [15] sa pre odstránenie radiálneho skreslenia používajú vzorce:

$$x_{corrected} = x(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$

$$y_{corrected} = y(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$

Teda v upravenom obraze súradnice ($x_{corrected}$, $y_{corrected}$) zodpovedajú starým súradniciam (x, y) z pôvodného obrázku. Pri bežnej kalibrácií stačia len dva koeficienty k. Pri veľkom deformovaní, napríklad použitím širokouhlej kamery, sa používa aj tretí koeficient k_3.

Tangenciálne skreslenie je spôsobené tým, že rovina snímacej plochy nie je rovnobežná s objektívom. Tým pádom vzniká skreslenie, ako to môžeme vidieť na obr.6, ktoré sa nazýva tangenciálne.



Obr. 1.5: Tangenciálne skreslenie kamery. [9]

Na odstránenie takéhoto tangenciálneho skreslenia sa využívajú nasledujúce vzorce:

$$X_{corrected} = x + [2 * p_1 * x * y + p_2 * (r^2 + 2 * x^2)]$$

$$Y_{corrected} = y + [p_1 * (r^2 + 2 * y^2) + 2 * p_2 * x * y]$$

Teda matica parametrov skreslenia ($Distortion_{coefficients}$) má 5 členov:

$$Distortion_{coefficients} = [k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3]$$

Na prevod jednotiek medzi súradnicovými systémami, ako sme uviedli už vyššie používame:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Hodnoty v ľavej matici sú x, y, w, kde x a y reprezentujú body v obrázku. Podľa [15] sa w vysvetľuje použitím homografie (pozri 1.1.4) a teda potom $w = Z$. Stredná matica obsahuje údaje o ohniskovej vzdialenosti (fx, fy) a pozícií optického centra (cx, cy). Matica, ktorá obsahuje tieto štyri parametre spolu s maticou, ktorá obsahuje informácie o rotácií R a translácií t, tvoria kamerovú maticu. Matica napravo obsahuje hodnoty X, Y, Z, čo sú body v reálnom svete.

Na vyrátanie týchto kalibračných parametrov nám stačí pár geometrických rovníc. Použitie týchto rovníc však závisí od toho, aký kalibračný vzor si zvolíme. Poznáme viacero vzorov na kalibrovanie. Základnými vzormi pre kalibráciu však sú klasická šachovnicová sieť, symetrický vzor s krúžkami a asymetrický vzor s krúžkami. Podľa známeho rozostavenia bodov v týchto vzoroch sa určí kamerová matica.

Podľa [23] máme v súčasnosti viaceré modely kalibrovania kamery. Tri modely uvedené v tomto zdroji riešia radiálne skreslenie a sú založené na princípe dierkovej kamery. Sú to modely ktoré vymysleli Tsai (1987) [25], Heikkila & Silven (1997) [20] a Zhang (2000) [27].

Tsaiov model predpokladá, že niektoré parametre kamery nám poskytne výrobca, aby sa znížilo hádanie v prvotnom odhade parametrov. V každom obrázku je potrebné nasnímať charakteristických n bodov (n>8). Kalibračný problém rieši potom pomocou n lineárnych rovníc. Tento model radiálneho skreslenia sa používa aj keď neberieme do úvahy decentralizované skreslenie. Na kalibráciu sa využíva kalibračná mriežka, kde sú známe pozície bodov v mriežke.

Model, ktorý vyvinuli Heikkila & Silven, najprv získa počiatočný odhad kamerových parametrov. Následne na ne aplikuje Levenberg-Marquardtov algoritmus, aby vyrátal parametre skreslenia. Tento model používa dva koeficienty pre obe, radiálne aj decentralizované, skreslenia. Taktiež používa kalibračnú mriežku.

Zhangov kalibračný model potrebuje, aby sa kalibračná mriežka nasnímala aspoň v dvoch rôznych orientáciách. Vyvinutý algoritmus vyberie rohy zo vzoru na vyrátanie projekčnej transformácie medzi obrázkami. Týmto sa získajú vnútorné aj vonkajšie parametre. Posledným krokom je minimalizovanie reprojekčnej chyby použitím Levenberg-Marquardtovej metódy.

1.1.3 Houghova transformácia

Houghova transformácia slúži na nájdenie priamok v obraze. Každá priamka je zaznamenávaná pomocou dvoch premenných v polárnom súradnicovom systéme v tzv. akumulátore. Čiže každá priamka je zaznamenaná pomocou jej vzdialenosti ρ od nuly (na kolmici k danej priamke) a uhla θ, ktorý zviera kolmica na danú priamku s osou x. Teda podľa [6] dostaneme zápis:

$$y = (-\frac{\cos\theta}{\sin\theta})x + (\frac{\rho}{\sin\theta})$$

Keď si z toho vyjadríme ρ, aby sme zistili hodnoty na jednej z ôs dostaneme:

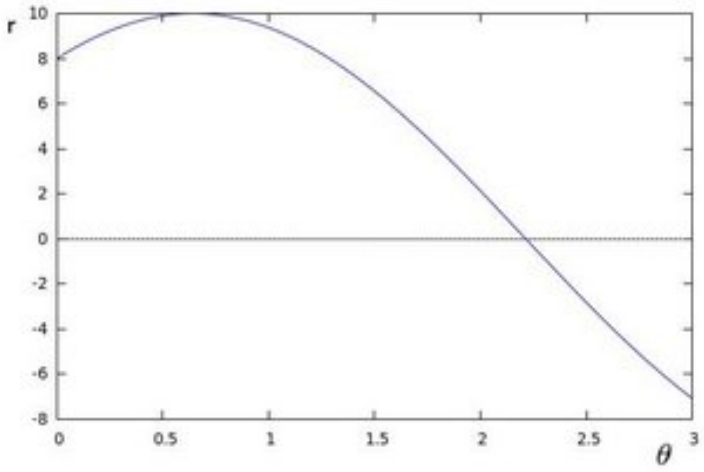
$$\rho = x * \cos\theta + y * \sin\theta$$

Každý bod (x_0, y_0) môžeme definovať ako skupinu priamok prechádzajúcich daným bodom nasledovne:

$$\rho_\theta = x_0 * \cos\theta + y_0 * \sin\theta$$

A zároveň jeden bod v polárnej súradnicovej sústave (ρ_θ, θ) reprezentuje jednu priamku v karteziánskej súradnicovej sústave, ktorá prechádza bodom (x_0, y_0). Teda keď si zakreslíme všetky možné úsečky prechádzajúce bodom (x_0, y_0), dostaneme sínusoidu, ako to môžeme vidieť na obrázku:

Ak sa pretnú sínusoidy rôznych bodov, znamená to, že tieto dva body ležia na spoločnej priamke. Je to priamka, ktorú reprezentuje daný bod v ktorom sa sínusoidy pretli. Čím viac sínusoíd sa pretne v jednom tom istom bode, tým viac bodov leží na danej priamke. Nakoniec vyberieme z akumulátora tie body, v ktorých sa pretlo viac priamok ako zadaný prah. Tieto potom reprezentujú skutočné priamky v obraze.



Obr. 1.6: Sínusoida bodu v polárnom súradnicovom systéme. [6]

1.1.4 Homografia

Ako sa uvádza v [17] homografia vyjadruje vzťah medzi bodmi v dvoch rôznych obrázkoch. Vo voľnom preklade by sme mohli povedať, že homografia zhruba znamená "podobné kresby". Ide o transformáciu, ktorá mapuje informácie o ploche z jedného pohľadu na druhý.

Podľa [24] sú dva obrázky prepojené homografiou iba ak na oboch obrázkoch pozorujeme rovnaký objekt len z iného uhlu.

Maticu homografie môžeme zapísať ako H, ako sa uvádza aj v zdroji [27]:

$$H = \begin{bmatrix} h_1 1 & h_1 2 & h_1 3 \\ h_2 1 & h_2 2 & h_2 3 \\ h_3 1 & h_3 2 & h_3 3 \end{bmatrix}$$

Homografiu ako transformáciu teda môžeme zapísať nasledovne:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 1 & h_1 2 & h_1 3 \\ h_2 1 & h_2 2 & h_2 3 \\ h_3 1 & h_3 2 & h_3 3 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homografia môže byť určená nájdením aspoň štyroch párov korešpondujúcich bodov. Každý pasujúci pár nám dáva dve obmedzenia, ktoré riešia dva stupne voľnosti.

Namapovať môžeme napríklad bod podľa [17]. Bod x z plochy vieme presunúť na jeho korešpondujúce miesto x' v druhom pohľade použitím: $x' = Hx$.

Kontúry rôznych objektov, alebo krivky, nedokážeme priamo presúvať do druhého obrázka použitím homografie. Môžeme ich však prenášať postupne ako body a čiary.

Metód odhadovania homografie existuje viacero. My si ukážeme však len jednu z mnohých a to metódu RANSAC, ako je to uvedené v [17]: Cieľ: Vypočítať homografiu medzi dvoma obrázkami s danou množinou možných korešpondujúcich párov.

Algoritmus:

- 1. Vybrať štyri body z množiny párov a vyrátať homografiu.
- 2. Vybrať všetky páry, ktoré súhlasia s vyrátanou homografiou. Dvojica $(x; x')$ súhlasí s homografiou H, ak $d(Hx; x') < t$, pre nejakú hranicu t (thershold) a $d(.)$ je euklidovská vzdialenosť medzi dvoma bodmi.
- 3. Opakovať kroky 1 a 2, kým nebude dostatočné množstvo dvojíc súhlasiť s vypočítanou homografiou.
- 4. Prepočítať homografiu s použitím všetkých vhodných korešpondencií.

1.2 Podobné práce, existujúce systémy

1.2.1 Podobné práce

Riešenia ktoré sú uvedené v tejto časti priamo neriešia rovnakú problematiku a to aplikáciu pre OS Android, ktorá meria dĺžku a šírku chodidla z fotografie. Pracujú však s obrazom z kamery, ktorý spracovávajú a rozpoznávajú v ňom rôzne objekty.

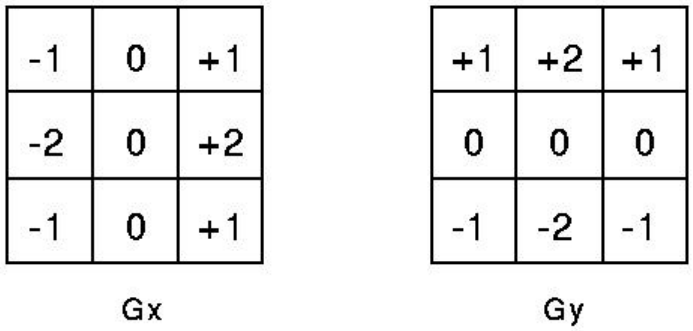
Bakalárska práca Miloša Fabiána [19] nás oboznamuje s problematikou rozpoznávania dopravných značiek. Keďže sa zvyšuje bezpečnosť na cestách, vznikajú také systémy, ktoré sa zaoberajú aj rozpoznávaním dopravných značiek. V súčasnosti stále viac a viac počuť o autonómnych vozidlách a aj preto je téma rozpoznávania dopravných značiek veľmi aktuálna. Z tejto práce môžeme čerpať informácie o spracovaní obrazu, keďže autor uviedol problematiku vhodným a jasným spôsobom. Spracovanie obrazu sa skladá zo štyroch fáz a to snímanie, predspracovanie, detekcia a rozpoznávanie.

Vo fáze snímania ide o zachytávanie aktuálneho obrazu, ktorý vidíme a teda ide o fotografovanie alebo natáčanie. Toto snímanie však ovplyvňujú rôzne faktory ako rozlíšenie kamery, šum, počasie a iné. Ďalšou fázou je predspracovanie. V nej sa snažíme čo najlepšie odstrániť poškodenia fotografie, ktoré nám zabraňujú alebo zhoršujú ďalšie spracovanie. Sú to napríklad šumy, ktoré sa snažíme odstrániť použitím rôznych filtrov na nasnímaný obraz. Alebo požívame morfologické operácie na zväčšenie alebo zmenšenie objektu. Ekvalizáciu histogramu používame na úpravu príliš jasnej alebo veľmi zosivenej snímky. V nasledujúcej fáze – detekcií sa autor snaží vyčleniť objekt, ktorý sa rozhodol skúmať, pomocou segmentácie. V poslednej fáze, rozpoznávanie, sa určuje, či vybraný objekt je skutočne ten, ktorý sme chceli. To sa deje pomocou určovania príznakov alebo strojovým učením.

Ďalšou prácou je Detekcia hrán a rohov v obraze od Stanislava Jursu [22]. V tejto práci sa dozvedáme o dôležitosti správnej detekcie hrán. Autor nás oboznamuje s rôznymi metódami detekcie hrán, ktoré kategorizuje do troch skupín: metóda prvej derivácie, metóda druhej derivácie a metódy porovnávania s ideálnymi hranami.

Hranu uvádza ako množinu pixelov ležiacich medzi dvoma regiónmi obrazu a ako kontúru, kde sa náhle mení výška jasu. Existujú rôzne príčiny nespojitosti jasových hodnôt v obraze. Napríklad nespojitosť v hĺbke, zmeny vo vlastnostiach materiálu alebo variácie v osvetlení scény a iné. Keď hrana nie je jasne daná (nie je to len čisto binárny obrázok) a jasových hodnôt je v prechode viac, vyberieme si hodnotu z prechodu, ktorá bude určovať hranu. Takýto proces sa nazýva prahovanie. Keďže v zašumenom obrázku nie je jasné či nejaký pixel je reálny bod alebo len šum, je potrebné na tento obrázok najprv použiť filter, ktorý ho vyhladí. Aj filtrov poznáme viacero druhov a to konkrétne napríklad filter strednej hodnoty, kde je hodnota jasu v pixeli nahradená hodnotou priemeru okolitých pixelov. Takýto filter ale rozmazáva hrany, preto je vhodnejšie použiť mediánový filter. Tento filter vyberie medián z jasových hodnôt okolia daného pixelu.

Metóda prvej derivácie spočíva v tom, že hrana je rozpoznaná na základe porovnania gradientu s hodnotou prahu. Gradient môže byť vygenerovaný v dvoch na seba kolmých smeroch obrazu, alebo vo viacerých smeroch. Pri generovaní v dvoch smeroch, sa dajú použiť napríklad metóda sobel alebo canny. Pri použití sobelovho operátora sa vo vyrátaní gradientu riadka a stĺpca vo štvor-okolí bodu, od ktorého rátame, zdvojnásobí hodnota, ako vidíme na obrázku obr.1.7.



Obr. 1.7: Sobelova metóda rátania hranice. [2]

V metódach druhej derivácie sa hrana pokladá za skutočnú, ak pozorujeme značnú priestorovú zmenu v druhej derivácií.

So spracovaním obrazu v operačnom systéme Android nás bližšie oboznamuje vo svojej bakalárskej práci [26] Michal Vozný. Treba si uvedomiť, že pri tvorbe aplikácií pre mobilné telefóny máme obmedzenejšiu výpočtovú kapacitu ako pri desktopových aplikáciách. Kvalita výsledného obrazu závisí aj od kvality vstupného obrazu, či je vhodne jasný, nerozmazaný atď.. Čiže aj takéto podmienky musíme brať do úvahy pri vyhodnocovaní výsledku. Čo sa týka rozdielu pri programovaní desktopových aplikácií a aplikácií pre mobilné telefóny, nevnímame ho ako veľmi výrazný. Rovnako sa dá programovať v jazyku Java, kde sú spúšťateľné skompilované knižnice naprogramované v jazyku C++. V tejto práci je pre nás zaujímavá časť o knižnici openCV pre Android,

o ktorej sa viac dočítame v nasledujúcej kapitole 1.3, Prehľad technológií.

1.2.2 Existujúce systémy

Z existujúcich riešení, aplikácií a systémov, ktoré sa zaoberajú meraním dĺžky a šírky chodidla z fotografie, nebola žiadna z nájdených aplikácií robená pre operačný systém Android. Veľmi podobná aplikácia, robená pre OS Android, bola ON 3D-CameraMeasure [4]. Táto aplikácia síce nemeria dĺžku, ani šírku chodila, ale zaoberá sa meraním dĺžok predmetov, nachádzajúcich sa v obraze. Toto meranie prebieha pomocou známeho objektu, ktorý sa nachádza v obraze. Užívateľ zadá, kde sa v obraze nachádza známy objekt, napríklad papier A4, a následne po označení predmetu, ktorého dĺžku chce užívateľ poznať, sa nad hranami zobrazia dĺžky vo zvoleých jednotkách (cm, inch). Toto druhé označenie je dynamické, a teda s ním užívateľ môže hýbať a hodnoty nad hranami sa ihneď prepočítavajú. Takto má užívateľ okamžitú spätnú väzbu.

Obr. 1.8: Ukážka merania dĺžky hrán stola na stolný tenis v aplikácií On 3D-CameraMeasure.[4]

1.3 Prehľad technológií

Podľa oficálnych zdrojov [14] je openCV open source knižnica, ktorá pracuje s počítačovým videním a strojovým učením. Obsahuje viac než 2500 algoritmov, ktoré sa dajú využiť napríklad na rozpoznávanie tvárí, identifikáciu objektov, sledovanie pohybov človeka vo videu, odstrániť červené oči z fotiek odfotených s bleskom, sledovanie pohybu očí, čiže kam sa človek pozerá, a iné. Má rozhrania pre C++, C, Python, Java a aj MATLAB. OpenCV sa dá používať na platformách Windows, Linux, Mac OS a Android. Ako sme sa dočítali v dokumentácií [13] a v [26] openCV obsahuje viaceré balíčky (packages) a toto je pár vybraných z nich:

- org.opencv.android - obsahuje metódy, ktoré slúžia napríklad na konverziu medzi bitovou mapou a dátovou štruktúrou Mat.
- org.opencv.calib3d – obsahuje metódy na kalibráciu kamery.
- org.opencv.imgproc – obsahuje algoritmy na spracovávanie obrazu, ako prahovanie obrazu, hranové detektory, konvolučné filtre a iné.

Kapitola 2

Návrh

Cieľom našej práce je zistiť dĺžku a šírku chodidla pomocou kamery a známeho predmetu. Teda objektom nášho skúmania boli snímky obsahujúce obrys chodidla užívateľa. Naším známym predmetom, podľa ktorého určujeme rozmery chodidla, je papier veľkosti A4. Kvôli perspektíve budeme tento papier transformovať na celú obrazovku (teda ako keby ležal rovnobežne s priemetňou), čím si zabezpečíme, že určitý počet pixlov bude zodpovedať určitej veľkosti v centimetroch.

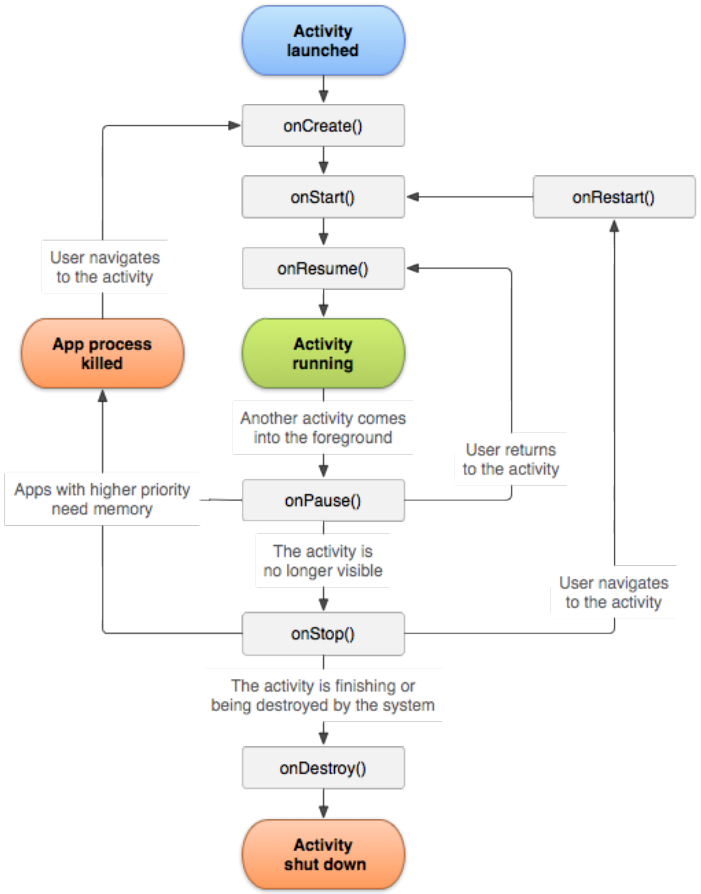
Spracovanie, ktorým dospejeme až k želanému výsledku môžeme zhrnúť do ôsmych krokov:

- 1. Kalibrácia
- 2. Prahovanie
- 3. Hľadanie hrán
- 4. Houghova transformácia
- 5. Nájdenie priesečníkov priamok
- 6. Homografia
- 7. Nájdenie opísaného štvorca chodidla
- 8. Meranie veľkosti

2.1 Android

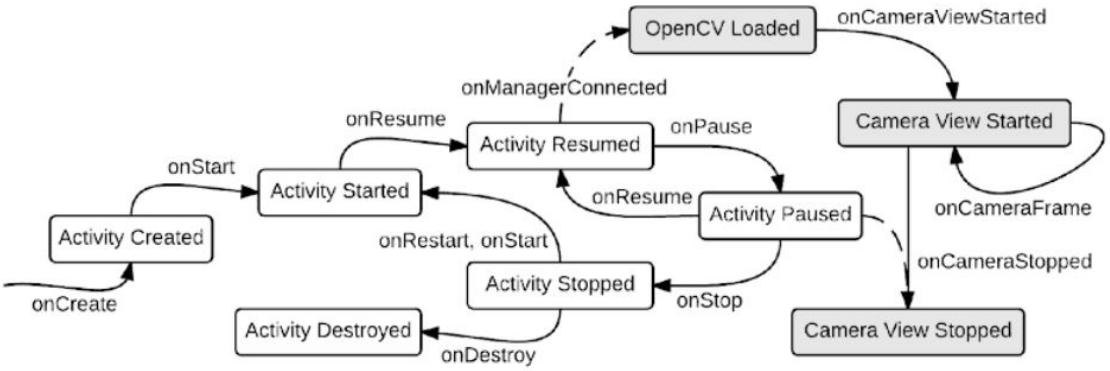
Nevyhnutými základnými stavebnými časťami aplikácie sú komponenty aplikácie. Ako sa píše na [18] poznáme štyri druhy komponentov: aktivity, služby, vysielacie prijímače (Broadcast Recievers) a poskytovatelia obsahu (Content Providers). Pre vysvetlenie

fungovania behu aplikácie je pre nás zaujímavý komponent aktivita. Aktivita je vstupom interakcie užívateľa. Predstavuje iba jednu obrazovku s UI (User Interface). Trieda Activity poskytuje niekoľko callback funkcií, pomocou ktorých aktivita vie, v akom je štádiu životného cyklu. Životný cyklus android aplikácie prebieha vo viacerých krokoch. Zdroj [1] uvádza, že jadro pozostáva zo šiestich callback funkcií, ako môžeme vidieť na nasledujúcom obr.:



Obr. 2.1: Životný cyklus aktivity v Android aplikácií. [1]

Po pridaní knižnice OpenCV, sa počet stavov v ktoých môže byť aplikácia zvýši, ako je to uvedené v [21]. Sivé boxy na obrázku 2.2 reprezentujú nové stavy, kotré patria životnému cyklu OpenCV knižnice. Prerušovanými čiarami je zobrazený vzťah medzi životným cyklom aktivity a životným cyklom OpenCV.



Obr. 2.2: Životný cyklus OpenCV aktivity v Android aplikácií. [21]

Prichádzajúce snímky budeme spracovávať vo funkcií *public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame)*. Táto funkcia dostane na vstupe premennú *inputFrame*, ktorá obsahuje aktuálny vstup z kamery. Na výstupe tejto funkcie je obraz – matica, ktorá sa zobrazuje na obrazovke užívateľa.

2.2 Kalibrácia

Obraz z kamery môže byť mierne skreslený kvôli šošovke. Ak by sme používali takýto obraz, mohlo by sa stať, že nejaké dôležité informácie budú skreslené. Dôležitá je napríklad rovná hrana papiera, alebo veľkosť obrysu. Ak by boli tieto informácie skreslené, čiže stena papiera alebo obrys by boli zakrivené, mohlo by to spôsobiť nepresnosť merania, alebo by sme nemuseli byť schopní nájsť papier v obraze. Preto je dôležité, aby bola naša kamera nakalibrovaná. (viac v 1.1.2). Informácia o vnútorných parametroch našej kamery, získaných z kalibrácie, nám umožní kompenzovať toto skreslenie a vytvoriť nový neskreslený obraz.

2.3 Hľadanie hrán

Aby sme dokázali zistiť, kde sa nachádza papier A4 s obrysom chodidla, môžeme obraz z kamery vyprahovať s určitým prahom tak, aby sme dostali binárny obraz. V tomto binárnom obraze bude papier biely a jeho okolie čierne.

Z takéhoto binárneho obrazu budeme môcť získať hrany použitím Cannyho algoritmu. Keďže Cannyho algoritmus použijeme až po prahovaní, počet hrán bude zredukovaný, lebo všetky šedé tóny pod prahom budú nastavené na čiernu a teda sa stratia určité tmavé prechody.

2.4 Houghova transformácia

Ak na obraz, na ktorom bol použitý Cannyho algoritmus, aplikujeme Houghovu transformáciu, dostaneme všetky priamky, ktoré sa v obraze nachádzajú (viac v 1.1.3). Nasledovne môžeme vyrátať, kde sa tieto priamky pretínajú a tým získame presné polohy rohov hľadaného papiera A4.

2.5 Homografia

Výpočet veľkosti chodidla nebudeme môcť realizovať, ak bude obraz snímaný z uhlu a nie priamo, pretože určitá veľkosť v pixeloch na jednej strane chodidla by nezodpovedalo rovnakej reálnej veľkosti ako rovnaká veľkosť v pixeloch na druhej strane.

Preto chceme transformovať obrázok tak, aby bol papier s chodidlom zobrazený na celej obrazovke, ako keby rovnobežne s kamerou. Vďaka už známej polohe rohov papiera budeme môcť zistiť maticu homografie a pomocou nej dokážeme obrázok želane transformovať.

2.6 Meranie veľkosti

Ak už budeme mať papier transformovaný homografiou a zobrazený na celú obrazovku, môžeme začať rátanie rozmerov chodidla. Najprv budeme musieť získať všetky body, ktoré patria chodidlu. Následne nájdeme opísaný obdĺžnik okolo týchto bodov. Keď teda získame opísaný obdĺžnik okolo chodila, tak jeho dĺžka je dĺžka chodidla a jeho šírka je šírka chodidla. Tieto veľkosti už budeme ľahko vedieť prepočítať z pixelov na centimetre vďaka známym rozmerom nájdeného papiera A4.

Kapitola 3

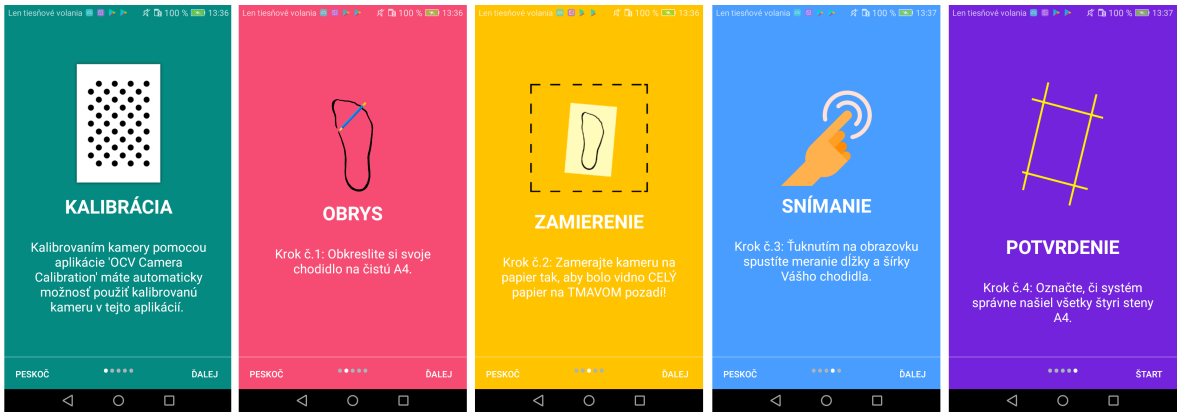
Implementácia

V tejto kapitole je opísaný postupný vývoj aplikácie a postupy, ktoré sme pri tvorbe použili.

3.1 Android

Naša aplikácia je programovaná pre platformu Android od verzie systému KitKat po Marshmallow. Ako vývojové prostredie sme zvolili AndroidStudio.

V našej aplikácií sme vytvorili dve aktivity. Sú podtriedami triedy AppCompatActivity. Prvá aktivita obsahuje informáciu o možnosti kalibrácie kamery a úvodný návod na používanie aplikácie a teda ako treba postupovať pri meraní chodidla. Tento postup je zhrnutý do štyroch bodov, ako to môžeme vidieť na obrázku:



Obr. 3.1: Slajdy z úvdného návodu aplikácie.

Všekty obrázky použité v aplikácií sú vytvorené vlastnoručne, okrem ruky, ktorá je zo zdroja [3]. Druhá aktivita obsahuje hlavnú časť našej práce. Nachádza sa v nej použite kamery, spracovanie jednotlivých snímok a aj zisťovanie rozmerov chodidla. Tento proces je bližšie opísaný nižšie.

3.2 Importovanie knižnice OpenCV4Android

Po vytvorení projektu v programe Android Studio 2.3 bolo naším prvým krokom importovať knižnicu OpenCV4Android, aby sme mohli zobraziť pohľad kamery a následne ho spracovať. Túto knižnicu sme stiahli z oficiálneho zdroja [12], verziu 3.3.0, kde je voľne dostupná. V čase sťahovania to bola najnovšia verzia knižnice OpenCV. Mobilnú verziu tejto knižnice sme vložili do projektu, aby bola zahrnutá v aplikácií a teda užívateľ si nemusel inštalovať knižnicu samostatne.

Zahrnutie tejto knižnice do aplikácie síce zväčšilo veľkosť aplikácie, ale uľahčilo to prácu užívateľovi. Keby nebola zahrnutá, bolo by potrebné túto knižnicu OpenCV4Android stiahnuť z tretej strany. Zariadenie po inštalácií našej aplikácie by automaticky ponúklo možnosť stiahnuť aplikáciu OpenCV Manager z obchodu Google Play. Daná aplikácia OpenCV Manager však bola v čase písania tejto bakalárskej práce naposledy aktualizovaná 21.9.2015, ako je to uvedené v [5]. Aplikácia preto nevyhovovala našim potrebám, pretože keď sme ju skúšali použiť, naša aplikácia nefungovala. Knižnicu OpenCV sme si do mobilu preto importovali pomocou aplikácie, ktorá bola súčasťou súborov v OpenCV4Android, ktoré sme si stiahli do počítača. Vhodnú aplikáciu z ponúknutého zoznamu knižníc sme vybrali podľa typu architektúry mobilného zariadenia na ktorom sme testovali.

Knižnicu OpenCV4Android sme zahrnuli do aplikácie aj po aktualizovaní prostredia Android Studio na verziu 3.0.1.

3.3 Povolenie

Nazačiatku bolo potrebné získať povolenie od používateľa aplikácie na používanie kamery. Toto povolenie sme zadefinovali v súbore AndroidManifest.xml. Následne sme mohli v hlavnej triede MainActivity vyžiadať dané povolenie. Toto získavanie sa nachádza vnútri callback-ovej funkcie onCreate().

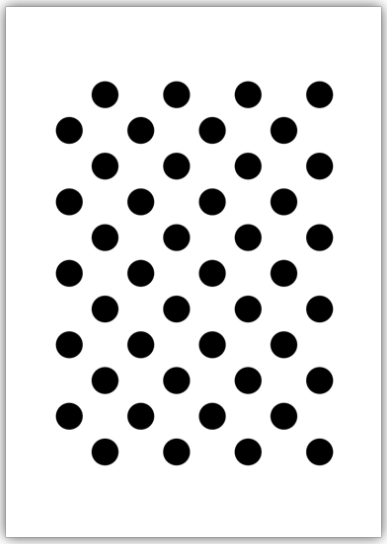
3.4 Kalibrácia

Na zistenie kamerovej matice a koeficientu skreslenia pre vyrátanie kalibrácie kamery zariadenia sme sa rozhodli používať oddelenú aplikáciu od našej aplikácie. Použili sme aplikáciu *camera-calibration*, ktorá bola súčasťou OpenCV knižnice ako ukážka.

Do tejto aplikácie sme doplnili vyžiadanie povolenia na používanie kamery. Aby sme mohli z našej aplikácie získať údaje o kalibrácií kamery, nastavili sme $SharedPreferences$ (rozhranie - interface, pomocou ktorého dokážeme ukladať informácie do zariadenia vo forme kľúč - hodnota a neskôr ich môžeme čítať) z MODE_PRIVATE na

MODE_WORLD_READABLE, čím sme umožnili prístup k ukladaným údajom aj iným aplikáciám.

Na kalibráciu sme požili kruhový vzor, viď. obr zo zdroja [10].



Obr. 3.2: Kalibračný kruhový vzor.

Pokiaľ si užívateľ kalibroval kameru pomocou danej aplikácie, v našej aplikácií na meranie chodidla sa mu zobrazí slajder. Pomocou tohoto slajdra môže určiť, či chce používať nekalibrovanú alebo kalibrovanú kameru. Na nasledujúcich obrázkoch môžme vidieť, že rozdiel medzi kalibrovanou a nekalibrovanou kamerou je takmer nebadateľný. To (vzhľadom na dôslednú kalibráciu) naznačuje, že šošovka nemá výrazné skreslenie.

Obr. 3.3: Zobrazenie pohľadu kamery s vypnutou kalibráciou.

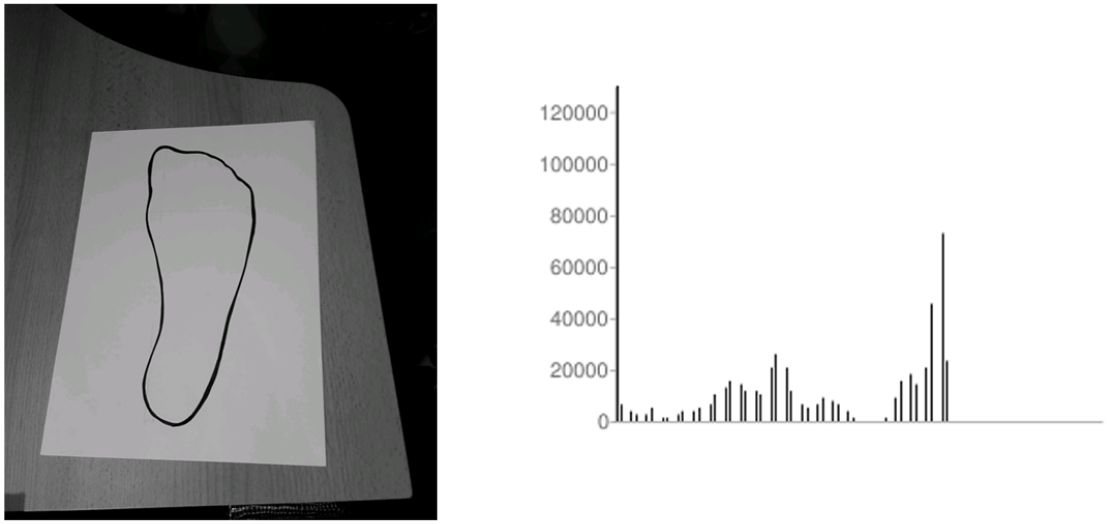
Obr. 3.4: Zobrazenie pohľadu kamery so zapnutou kalibráciou.

3.5 Prahovanie

Na zistenie papiera v obraze sme predpokladali nasledujúce tvrdenia:

- Snímaný papier je svetlejší, ako jeho okolie
- Snímaný papier má viditeľné všetky štyri steny

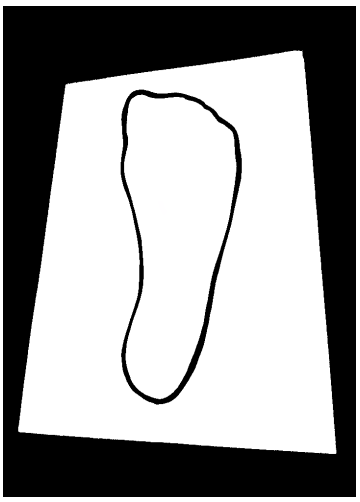
Vstupný farebný obraz z kamery, sme najprv transformovali na šedotónový obraz a podľa vyššie uvedených predpokladov vieme tento obraz zmeniť na binárny tak, aby ostal papier biely a jeho okolie čierne. Na toto spracovanie sme spravili funkciu *public Mat myThreshold(Mat image, int thresh_value)*. Má dva vstupné parametre a to *Mat image* a *int thresh_value*, kde image je obraz, ktorý sa spracováva a thresh_value je prah, podľa ktorého sa obraz bude prahovať. Hodnotu prahu sme po viacerých testovaniach nastavili na 125. Táto hodnota je približne v strede rozsahu histogramu šedotónového obrázku (0 - 255). Táto hodnota nie je vyššia, ako by sa dalo predpokladať že bude, keďže je papier biely. Nachádza sa v strede kvôli tomu, že papier nemusí byť rovnomerne osvetlený a teda tieň môže spôsobiť, že aj nižšie jasové hodnoty ešte patria papieru. Takýto prah bol vyhovujúci v 9 z 10 prípadov.



Obr. 3.5: Šedoúrovňové zobrazenie vstupného obrazu a jeho histogram.

Keďže boli metódy get() a put(), ktoré sme využívali v našej funkcií, časovo náročné, rozhodla som sa použiť metódu *public static double threshold(Mat src, Mat dst, double thresh, double maxval, int type)* triedy *org.opencv.imgproc.Imgproc*. Na vstupe bola matica obsahujúca obrázok, ktorý sa bude spracovávať a matica do ktorej sa uloží obrázok. Hodnotu prahu som nastavila na zistených 125, ako sme spomínali vyššie. Premenná *maxval* predstavuje hodnotu, ktorá bude nastavená namiesto hodnôt, ktoré

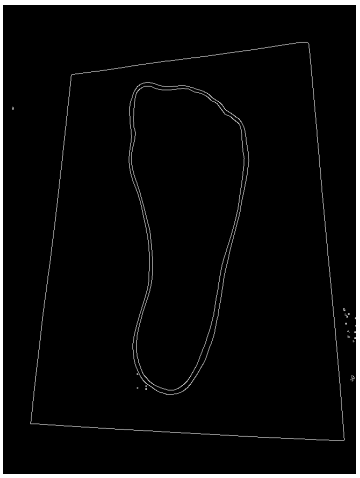
sa nachádzajú nad prahom. Ako premennú type sme zvolili THRESH_BINARY, lebo chceme dosiahnuť binárny obrázok. Prahovaný obrázok vyzerá nasledovne:



Obr. 3.6: Výsledný prahovaný binárny obrázok.

3.6 Získavanie hrán

Na získanie hrán v obrázku sme použili Cannyho algoritmus implementovaný v metóde *public static void Canny(Mat image, Mat edges, double threshold1, double threshold2, int apertureSize, boolean L2gradient)* triedy *org.opencv.imgproc.Imgproc*. Vstupným obrázkom bol náš binárny obrázok. Na výstupe sme dostali hrany v premennej edges. *Threshold1* a *threshold2*, prahy, sme nastavili na hodnotu 50, 200 ako horný a dolný prah. Tieto prahy sa ukázali ako najúčinnejšie pri testovaní na desiatich snímkoch. Premennú apertureSize, ktorá vyjadruje veľkosť okolia, s ktorým sa pracuje, sme nastavili na 3, lebo pri získavaní výrazných hrán je presnejší výsledok pri práci s malým okolím. Výsledok môžeme vidieť na obr.canny

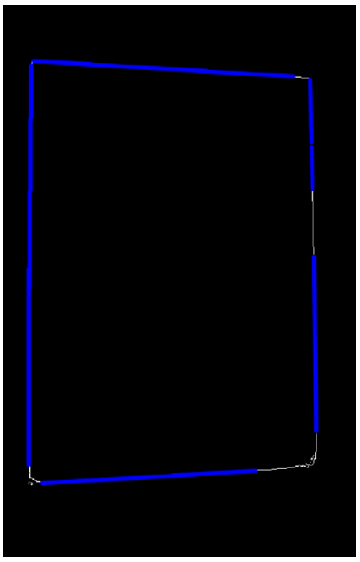


Obr. 3.7: Cannyho transformácia použitá na náš obrázok chodidla.

Na obrázku 3.7 môžeme vidieť dve kontúry chodidla. Je to spôsobené hrúbkou čiary akou sa obkreslovalo chodidlo. Keby bola táto čiara príliš tenká, alebo svetlá (ako sa to stáva pri používaní ostrej ceruzky), mohla by sa kontúra chodidla po homografií rozpadnúť na neúplný obrys s veľkými dierami medzi jednotlivými časťami kontúry.

Pomocou zobrazených hrán sme dokázali nájsť priamky, ktoré sa na obrázku nachádzajú. Spravili sme na to vlastnú funkciu *public double[][] myHough(Mat edges)*. Oproti obyčajnej Houghovej transformácií je naša funkcia obohatená o to, že vráti len štyri najvýraznejšie priamky z obrazu, ktoré sú na seba približne kolmé (45°- 135°kvôli tomu, že pravý uhol sa zdá byť pri pohľade z perspektívy väčší alebo menší).

Prvým krokom bolo získanie všetkých priamok, ktoré sa nachádzajú v obraze. Najprv sme používali takú implementáciu Houghovej transformácie, ktorá nám vrátila úsečky vyskytujúce sa v obraze. Na identifikáciu hľadaného papiera sme však potrebovali poznať, kde sa nachádzajú jeho rohy. Pri používaní úsečiek sa ale stávalo, že neoznačilo celú hranu papiera ako jednu úsečku (viď 3.8). A teda nebolo možné určiť, kde je roh papiera.



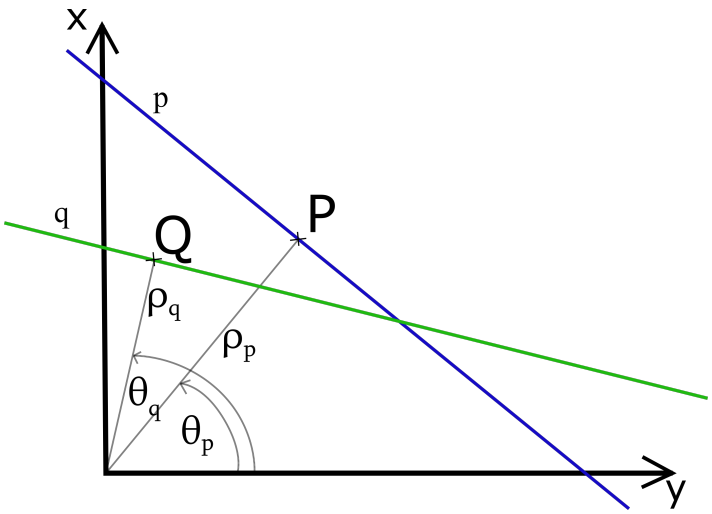
Obr. 3.8: Výsledný obraz metódy, ktorá vracia úsečky z Houghovej transformácie.

Použili sme teda metódu *public static void HoughLines(Mat image, Mat lines, double rho, double theta, int threshold)* triedy *org.opencv.imgproc.Imgproc*, ktorá vracia priamky. Priamky dostaneme ako maticu, kde každý prvok matice je dvojrozmerné pole. Toto pole obsahuje informácie z akumulátora Houghovej transformácie a teda hodnoty θ (uhol danej priamky s osou x) a ρ (vzdialenosť priamky od bodu [0, 0]) (viac v 1.1.3).

Nasledujúcim krokom bolo vyselektovať zo všetkých priamok, práve štyri také, ktoré sú na seba kolmé a sú od seba dostatočne vzdialené. Čiže presne štyri priamky lemujúce papier. V prvom selektovaní sme vyradili z matice tie priamky, ktoré sú približne

totožné. Za približne totožné priamky sme pokladali také, ktoré sa nachádzajú blízko seba a sú otočené len o určitý uhol, dostatočne malý alebo veľký na to, aby to, s veľkou pravdepodobnosťou, nebol roh papiera ani keď sa pozeráme na papier z uhlu.

obr:houghRozdiely



Obr. 3.9: Vzdialenosť a uhol dvoch priamok p, q.

Ak uhol θ vyjadruje pre jednu priamku uhol medzi osou x a danou priamkou, tak uhol medzi dvoma priamkami p, q dokážeme vyjadriť ako $\theta_p - \theta_q$ (pozri obr. 3.9). Toleranciu na uznanie uhla medzi dvoma priamkami ako roh papiera sme si zvolili rozmedzie 45° až 135°. Aby sme mohli tvrdiť, že tieto dve priamky sa nachádzajú pri sebe, okrem uhla musia mať aj podobnú vzdialenosť od bodu $[0, 0]$. Túto vzdialenosť vyjadruje ρ. Vzdialenosť medzi dvoma priamkami p, q teda približne určuje absolútna hodnota ich rozdielu hodnoty ρ: $|\rho_p - \rho_q|$. Vzdialenosť pre označenie ako približne totožnej priamky by nemala prekročiť 150, pri obraze veľkosti 960x720.

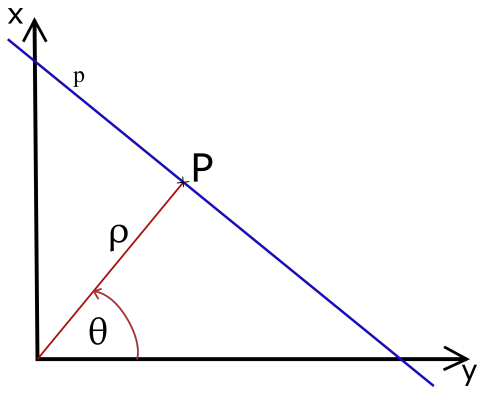
V druhom kole selektovania sme z vyselektovaných priamok vybrali prvú najvýraznejšiu priamku. K nej sme vybrali priamky, ktoré sú na ňu kolmé. Potom sme pokračovali analogicky pre ďalšie priamky, až kým sme nezískali štyri priamky.

V určitých prípadoch, napríklad tmavá alebo presvetlená snímka, nedostaneme správne všetky štyri priamky. Kvôli zjednodušeniu sme si preto vytvorili dialógové okno (viď. 4.3), v ktorom užívateľ odklikne, či našlo tieto hľadané štyri priamky správne.

Takto sme dostali štyri lemujúce priamky hľadaného papiera A4. Aby sme mohli použiť homografiu na zobrazenie papiera na celú obrazovku, musíme nájsť významné body, ktoré spárujeme s pripravenou maticou, ktorá predstavuje čistý papier A4. Významné body sú štyri rohy papiera, ktoré budeme párovať s pripravenou maticou na zistenie matice homografie.

3.7 Hľadanie priesečníkov

Rohy papiera sa nachádzajú na priesečníkoch našich nájdených štyroch priamok. Priesečník sme zistili pomocou parametrických rovníc priamok. Zoberme si dve priamky, p a q. Parametrickú rovnicu pre priamku p zostrojíme pomocou smerového vektora priamky p. Lenže na to musíme poznať aspoň dva body, ktoré ležia na danej priamke. Keďže vieme, že smerový vektor je kolmý na normálový vektor danej priamky, vieme ľahko určiť smerový vektor pomocou normálového vektora. Normálový vektor priamky p je smerový vektor kolmice na priamku p, ktorá ide z bodu $[0,\ 0]$ do bodu P, ako to môžeme vidieť na obr. 3.10:



Obr. 3.10: Vzdialenosť ρ a uhol θ priamky p.

Pozíciu bodu P sme vyrátali pomocou Pytagorovej vety a údajov, ktoré máme o priamke z Houghovej transformácie. Veľkosť úsečky $|[0,0],P|$ vyjadruje hodnota ρ a uhol medzi osou x a ρ vyjadruje θ. Súradnice P_x a P_y sme teda vypočítali nasledovne:

$$P_x = \rho * cos(\theta)$$

$$P_y = \rho * sin(\theta)$$

V tomto bode sme už vedeli zostrojiť smerový vektor úsečky ρ:

$$\vec{\rho} = (P_x - 0, P_y - 0)$$

To sme upravili ako:

$$\vec{\rho} = (P_x, P_y)$$

Normálový vektor ku priamke ρ a zároveň smerový vektor priamky p sme teda zapísali nasledovne:

$$\vec{np} = (P_y, -P_x)$$

Parametrické vyjadrenie priamky p sme preto mohli zapísať ako:

$$p : x = P_x + n\vec{p}_1$$

$$y = P_y + n\vec{p}_2$$

Pre výpočet parametrického vyjadrenia priamky q sme postupovali analogicky, ako pre priamku p:

$$q : x = Q_x + n\vec{q}_1$$

$$y = Q_y + n\vec{q}_2$$

Aby sme dostali priesečník týchto priamok, zostavili sme si sústavu rovníc o dvoch neznámych:

$$P_x + n\vec{p}_1 = Q_x + n\vec{q}_1$$

$$P_y + n\vec{p}_2 = Q_y + n\vec{q}_2$$

Na vyriešenie týchto rovníc sme použili knižnicu JAMA.

JAMA je balíček (package) pre jazyk java, pomocou ktorého vieme počítať lineárne algebraické úlohy. Vyvinuli ho MathWorks a NIST (National Institution of Standards and Technology). Slúži hlavne na prácu s maticami. Je zameraný hlavne na operácie, ktoré sú najbežnejšie používané. Dokáže teda vykonávať základné operácie ako sčítanie, odčítanie, násobenie, transponovať, vyrátať skalárny súčin, determinant a aj rátať sústavu rovníc [8].

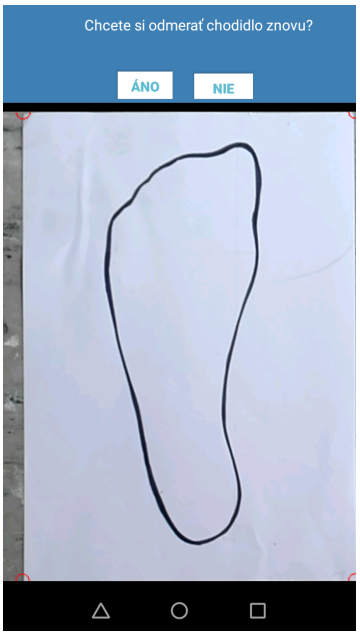
Takto sme dostali súradnice priesečníka dvoch priamok z Houghovej transformácie. Rovnaký postup sme aplikovali na všetky dvojice pretínajúcich sa priamok. Tým sme získali štyri body - vrcholy hľadaného papiera A4.

3.8 Aplikovanie homografie

Po nájdení rohov hľadaného papiera A4 sme mohli použiť homografiu na zobrazenie A4 na celom displeji. Párovali sme teda rohy A4 s našou predpripravenou maticou, ktorá má jednu stranu veľkú ako hrana displeja a druhú stranu relatívne veľkú, aby sedel pomer strán $1 : \sqrt{2}$ ako pri pomere strán na papieri A4.

Použili sme metódu *public static Mat findHomography(MatOfPoint2f srcPoints, MatOfPoint2f dstPoints, int method, double ransacReprojThreshold)* triedy *org.opencv.calib3d.Calib3d*. Vstupnými parametrami *srcPoints* a *dstPoints* boli pozície rohov hľadaného papiera A4 z kamery a pozície rohov nášho pripraveného papiera na celej obrazovke. Pomocou tejto metódy sme našli maticu homografie. Na aplikovanie homografie na náš vstupný obrázok a teda na zobrazenie nájdeného papiera na celú

obrazovku sme použili metódu *public static void warpPerspective(Mat src, Mat dst, Mat M, Size dsize)* triedy *org.opencv.imgproc.Imgproc*. Na vstupe bol obraz z kamery *src*, matica homografie M a na výstupe sme dostali transformovaný obraz *dsc*, ako to môžeme vidieť na nasledujúcom obrázku:

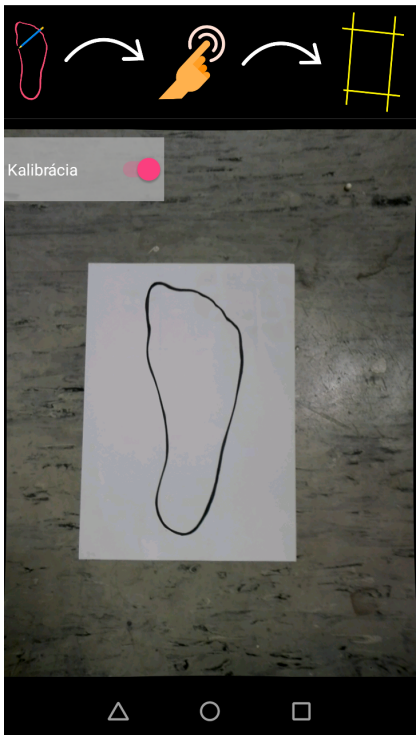


Obr. 3.11: Obraz po aplikovaní homgrafie.

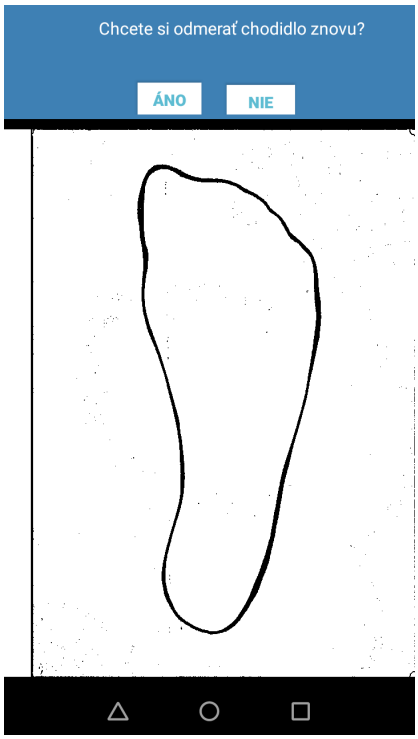
3.9 Adaptívne prahovanie

Rozhodli sme sa získaný obraz znovu prahovať, kvôli lepším výsledkom. Pomocou homografie sme si nazačiatku zobrazovali na celú obrazovku už prahovaný obraz. Tento obraz však neobsahoval všetky potrebné informácie, pretože prah, ktorý sme použili spôsobil, že obrysy kreslené ceruzkou alebo bledým perom nebolo vidieť.

Preto sme pomocou homografie zobrazovali pôvodný neupravený obraz, ktorý prišiel z kamery. Po homografií sme aplikovali adaptívne prahovanie. Použili sme funkciu *public static void adaptiveThreshold(Mat src, Mat dst, double maxValue, int adaptiveMethod, int thresholdType, int blockSize, double C)* triedy *org.opencv.imgproc.Imgproc*. Ako vstupnú maticu sme zadávali neupravený obraz z kamery. Prah sa počíta z okolia blockSize, v našom prípade to bolo 11, metódou $ADAPTIVE\_THRESH\_GAUSSIAN\_C$, typom $THRESH\_BINARY$, čiže sa tvoril binárny obrázok kde sa prah počítal z priemeru okolia. Za hodnoty nad prahom sa nastavovala $maxValue$, v našom prípade 255.



Obr. 3.12: Kalibrovaný vstup z kamery.



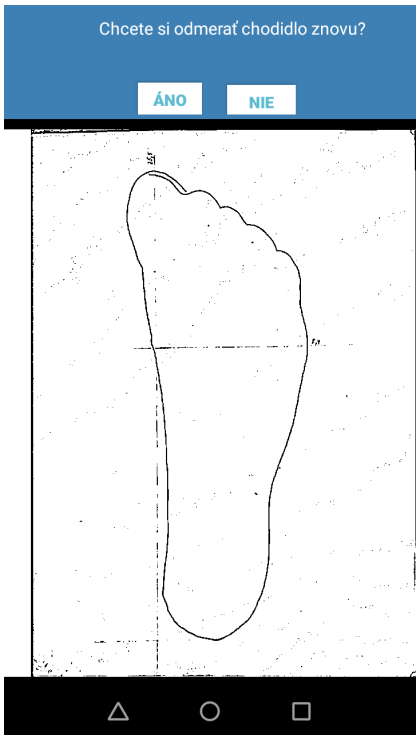
Obr. 3.13: Zobrazenie A4 na celej obrazovke po adaptívnom prahovaní.

3.10 Morfologické operácie

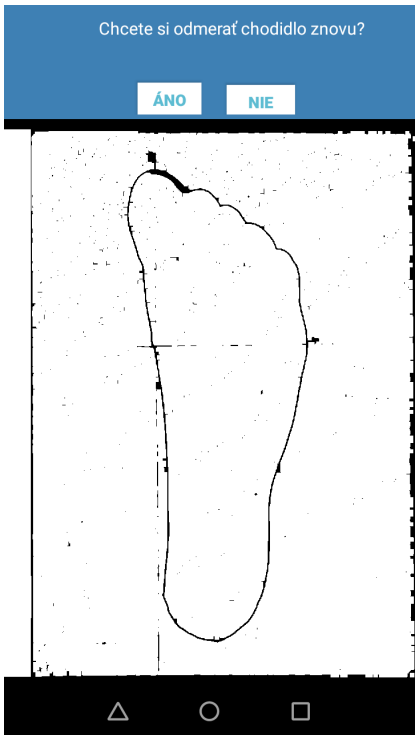
Kvôli prahovaniu sa môže stať, že obrys chodidla nebude úplne na všetkých miestach spojitý. Preto sme sa rozhodli použiť morfologické operácie, eróziu a dilatáciu. Postupným prechádzaním obrazu s určím okolím sa celé okolie zmení na čiernu alebo bielu. Ak sú v okolí pixela samé biele pixely, tento pixel ostáva biely, inak sa zmení na čiernu - erózia. Opakom erózie je dilatácia, čiže ak sa v okolí pixela nachádza nejaký biely pixel, mení sa na biely, ako je to uvedené v [16].

Chceli sme dosiahnuť uzatvorenie (uzavretie malých dier a nespojitých oblastí objektu). Tieto morfologické operácie sú však postavené na tom, že farba popredia (objektu) je biela a farba pozadia čierna. Naše farby sú však opačné, preto budeme na uzatvorenie používať našu funkciu *public Mat myInvClosing(Mat src)*, čo je v skutočnosti otvorenie.

Otvorenie je použitie najprv erózie a po nej dilatácie, teda sme najprv kontúry chodidla zhrubli a následne späť stenčili. Použili sme funkciu *public static void morphologyEx(Mat src, Mat dst, int op, Mat kernel)* triedy *org.opencv.imgproc.Imgproc*. Na vstupe *src* bol náš spracovaný obraz. Použili sme otvorenie, teda parameter *op* bol *MORPH_OPEN*. Ako *kernel* (okolie) sme použili sedem-okolie. Tým sme dosiahli spojenie obrysu na miestach, kde nebol spojitý.



Obr. 3.14: Obrázok pred použitím morfologických operácií.



Obr. 3.15: Obrázok po použitím morfologických operácií.

3.11 Hľadanie kontúry

Na hľadanie kontúr chodidla sme mali viaceré možnosti. Uvažovali sme nad metódami *connectedComponents* a *findContours* triedy *org.opencv.imgproc.Imgproc*.

Ako prvú sme skúšali metódu *connectedComponents*, ktorá postupne prechádza celý obraz a každému pixlu priradí číslo oblasti. Rovnaké číslo oblasti majú také pixely, ktoré spolu susedia a majú rovnakú farbu. Táto metóda vracia maticu s označenými oblasťami. Po získaní oblastí sme si usporiadali všetky body do asociatívneho poľa typu Map. Číslo oblasti boli kľúče a body danej oblasti boli hodnoty. Následne sme vybrali druhú najväčšiu oblasť, pretože najväčšia spojitá oblasť je biele pozadie papiera. Ak máme neprerušovaný obrys chodila, tak druhou najväčšou oblasťou v obraze je biele vnútro chodila.

Druhou testovanou metódou bola metóda *findContours*. Táto metóda vracia zoznam matíc. Každá matica zo zoznamu obsahuje body jednotlivých kontúr. Podobne ako pri predchádzajúcej metóde sme vyberali druhú najväčšiu maticu.

Po testovaní oboch metód sme sa rozhodli použiť *findContours*, pretože v metóde *connectedComponents* sa stávalo, že aj po morfologických trasformáciách neoznačilo celé chodidlo ako jednu oblasť, alebo ako druhú najväčšiu oblasť nenašlo vnútro chodidla.

3.12 Opísaný obdĺžnik a veľkosť chodidla

Vďaka získaným bodom, ktoré sa nachádzajú vnútri obrysu chodidla, sme mohli použiť metódy na ohraničenie oblastí. Použili sme metódu *public static Rect boundingRect(MatOfPoint points)* triedy *org.opencv.imgproc.Imgproc*, ktorá okolo zadanej oblasti vytvorí obdĺžnik. Keďže sme zadávali ako vstupný parameter *points* druhú najväčšiu nájdenú oblasť a to vnútro obrysu chodidla, môžeme povedať, že sme získavali opísaný obdĺžnik okolo chodila. Tým, že sme nepoužili tretiu najväčšiu oblasť (obrys) sme eliminovali nepresnosti merania rozmerov chodidla, spôsobené hrúbkou ceruzky, pera či fixky.

Táto metóda vracia obdĺžnik, ktorý je zarovnaný so súradnicovými osami (dve strany sú paralelné s osou x a dve s osou y). Pri testovaní aplikácie užívateľmi, si niektorí užívatelia obkreslovali chodidlo došikma. Preto sme sa rozhodli použiť metódu *public static RotatedRect minAreaRect(MatOfPoint2f points)* z rovnkej triedy ako predchádzajúca metóda. Táto metóda vracia najmenší obdĺžnik okolo zadanej oblasti. Preto ak si užívateľ obkreslí chodidlo došikma, aj opísaný obdĺžnik chodidla bude došikma.

Získaním dĺžky a šírky opísaného obdĺžnika získame aj dĺžku a šírku chodidla v pixeloch. Tieto rozmery si vieme jednoducho trojčlenkou prepočítať na centimetre, pretože rozmery A4 sú známe (29,7cm : 21cm) a teda vieme, že počet pixelov vo väčšej strane obrazovky (v šírke obrazovky, pretože pracujeme v *landscape* móde) zodpovedá dĺžke 29,7cm. Prepočet z pixelov na centimetre môžeme preto zapísať nasledovne:

Ak je v nájdenom obdĺžniku šírka väčšia ako dĺžka:

$$dlzka\_chodidla = \frac{sirka\_stvorca * 29,7}{sirka\_obrazovky}$$

$$sirka\_chodidla = \frac{dlzka\_stvorca * 29,7}{sirka\_obrazovky}$$

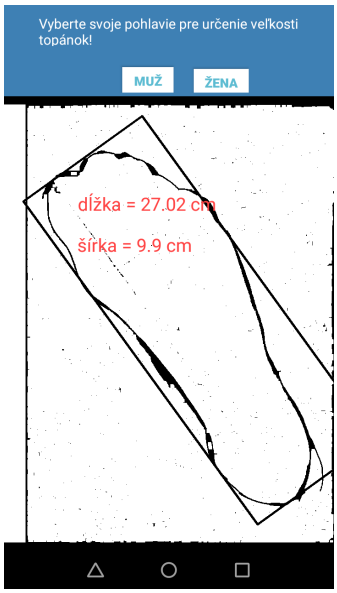
Ak je v nájdenom obdĺžniku dĺžka väčšia ako šírka:

$$dlzka\_chodidla = \frac{dlzka\_stvorca * 29,7}{sirka\_obrazovky}$$

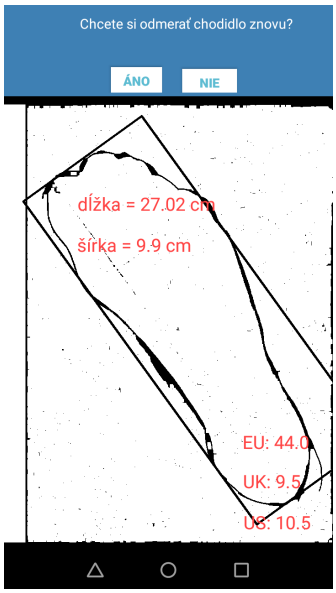
$$sirka\_chodidla = \frac{sirka\_stvorca * 29,7}{sirka\_obrazovky}$$

Podľa zistenej hodnoty sme pomocou tabuliek zo zdroja [7] a [11] užívateľovi vypísali informáciu o čísle topánky pre jeho veľkosť chodidla v európskom, anglickom a americkom číslovaní. Keďže americké číslovanie sa delí na ženské a mužské, vytvorili sme dialógové okno, kde užívateľ zvolí, aké má pohlavie. Po zistení dĺžky chodidla v centimetroch sa táto veľkosť vypíše na obrazovku. Keď si užívateľ vyberie svoje pohlavie, zobrazia sa mu aj veľkosti topánok zodpovedajúcich danej veľkosti.

Výsledný obraz môžeme vidieť na nasledujúcich obrázkoch:



Obr. 3.16: Výsledný obraz s veľkosťou chodidla v cm.



Obr. 3.17: Výsledný obraz s veľkosťami topánok pre dané rozmery chodidla.

Kapitola 4

Používateľské rozhranie

Naše rozhranie sa skladá z dvoch častí a to úvodný návod a hlavná obrazovka. Úvodný návod je pár obrazoviek, kde je popísané ako treba postupovať pri používaní aplikácie. Tieto obrazovky môžete vidieť vyššie viď. 3.1. Rozloženia komponentov (layout) sú definované v súbore activity_tutorial.xml. Každý slajd návodu je definovaný v .xml súbore s názvom slide + číslo slajdu. Zostrojili sme ich pomocou Android Studia, kde sme nastavili farbu pozadia každému slajdu, ďalej sme pridali komponenty TextView pre texty a ImageView pre obrázky. Obrázky sme zostrojili pomocou programu Photoshop.

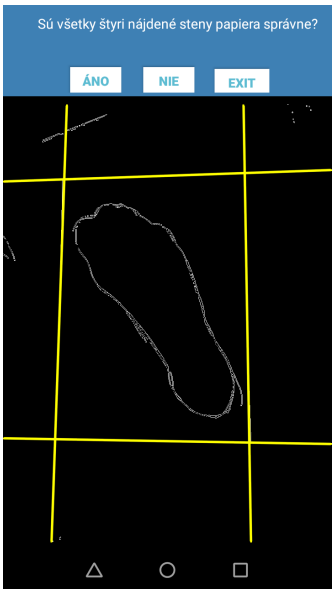
Na definovanie rozloženia elementov hlavnej aktivity na obrazovke slúži súbor show_camera.xml. Doň sme pridali komponent JavaCameraView, ktorý predstavuje pohľad z kamery. Kvôli používaniu kamery sme sa rozhodli použiť napevno orientáciu na šírku (Landscape). Keby bolo povolené otáčanie obrazovky podľa orientácie zariadenia, otáčal by sa aj kamerový pohľad. To by ho však deformovalo, pretože by bol otočený o 90°. Toto rozhodnutie spôsobilo len to, že sa hlavička aplikácie neotáčala podľa orientácie zariadenia, ale ostala stále v rozložení na šírku. Túto hlavičku sme sa neskôr aj tak rozhodli odstrániť, kvôli designu aplikácie. Namiesto hlavičky sme do hlavnej obrazovky umiestnili komponent ImageView, kde je zobrazený stručný návod len pomocou ikon, ako môžete vidieť na obr. 4.1

Obr. 4.1: Panel so skratkami návodu na používanie aplikácie.

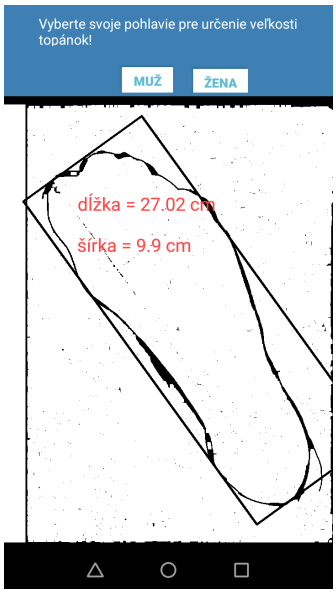
Do hlavnej obrazovky sme tiež pridali okrúhly komponent ProgressBar, ktorý sme použili ako informáciu pre užívateľa o tom, že aplikácia pracuje – prebiehajú výpočty.

Obr. 4.2: Krútiaci krúžok zobrazovaný počas načítavania.

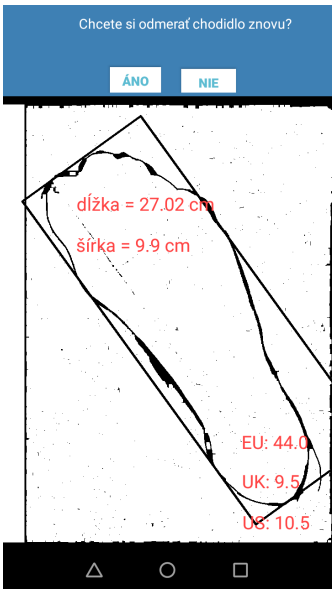
Cez návod v hornej časti obrazovky sa postupne zobrazujú dialógové okná na potvrdenie správnosti nájdeného papiera a tiež dialógové okno na opakovanie merania. Pre dané dialógové okná sme nepoužili triedu *public class Dialog*, pretože sme nechceli, aby bolo okno zobrazené v strede obrazovky a ani aby stmavlo pozadie pod ním, keďže sme potrebovali, aby užívateľ videl, čo je práve na obrazovke. Vytvorili sme si preto LinearLayout, v ktorom sme si vytvorili celý výzor dialógových okien. Tieto dialógové okná môžete vidieť nižšie.



Obr. 4.3: Dialógové okno po Houghovej transformácií.



Obr. 4.4: Výsledný obraz s veľkosťou chodidla v cm.



Obr. 4.5: Výsledný obraz s veľkosťami topánok pre dané rozmery chodidla.

Výsledky merania sú zobrazené v komponente TextView. Tento text sa nachádza približne v strede obrazovky. Informácia o veľkosti topánky sa nachádza v pravom dolnom rohu (viď. obr. 4.5).

Kapitola 5

Vyhodnotenie

V tejto kapitole sme zhrnuli výsledky a určli priemernú odchylku merania rozmerov chodila.

V nasledujúcej tabuľke je zaznamenaný čas trvania jednotlivých metód, ktoré sme použili v aplikácií:

Rýchlosť jednotlivých metód	
metóda	čas v milisek.
Prvé prahovanie	2
Cannyho algoritmus	22
Houghova transformácia a vykreslenie nájdených priamok	234
Rátanie priesečníkov	10
Získanie matice homografie	7728
Druhé prahovanie	578
Uzatvorenie	42
Hľadanie chodidla	58
Nájdenie a vykreslenie opísaného obdĺžnika	3

Ako vidíme vo vyššie uvedenej tabuľke, metóda na získavanie matice homografie trvá najdlhšie. Pre zrýchlenie by sme v budúcnosti mohli skúsiť použiť namiesto metódy RANSAC metódu, ktorá pracuje so všetkými bodmi.

Aplikáciu sme skúšali na vzorke dvadsiatich obrysoch chodidla. Tieto obrysy boli odmerané pravítkom a porovnané s výsledkami našej aplikácie. V tabuľke *Odmerané a zistené hodnnoty v cm* sú za poradovým číslom v prvých dvoch stĺpcoch zaznamenané rozmery chodidla zistené pomocou pravítka a druhých dvoch stĺpcoch sú zaznamenané hodnoty namerané pomocou našej aplikácie.

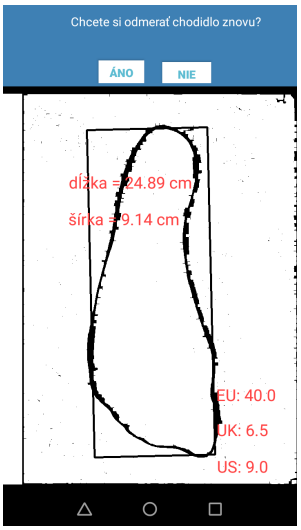
Odmerané a zistené hodnnoty v cm				
č. chodidla	odmeraná dĺžka	odmeraná šírka	zistená dĺžka	zistená šírka
1.	24,9	9,3	24,89	9,14
2.	27,1	10,2	27,02	9,9
3.	26,1	9,5	26,41	9,14
4.	24,6	9,6	24,6	9,56
5.	26,4	8,8	25,92	8,57
6.	25,5	8,4	25,49	9,03
7.	24,9	9,1	25,06	8,98
8.	25,7	8,5	25,6	8,51
9.	26	10,2	25,85	10,17
10.	25,8	10,2	25,54	10,19
11.	25,8	8,8	25,73	8,66
12.	27,8	10,5	27,74	10,2
13.	21,5	7,8	21,38	7,66
14.	27,7	10	27,49	9,74
15.	19,8	7,2	19,6	6,25
16.	25,3	10	25,12	9,85
17.	29,4	10,6	29,37	10,5
18.	23	7,9	22,85	7,98
19.	29,9	11,8	29,41	11,93
20.	25,5	9	25,2	8,69

Z nameraných hodnôt sme zistili, že priemerná odchýlka v dĺžke chodidla je 0,1685 cm a v priemerná odchýlka v šírke je 0,2225 cm. Najpresnejšie naša aplikácia odmerala chodilo s rozdielom 0 cm a najväčšia odchýlka našich meraní bola 0,95 cm.

Odchýlky medzi týmito typmi merania mohli byť spôsobené nepresným meraním pravítkom, nepresným zistením matice homografie alebo sa mohlo stať, že obrys chodidla ostal po morfologických operáciách širší. Nepresnosti mohlo spôsobiť aj deformovanie obrazu kvôli šošovke, ak užívateľ nepoužil kalibráciu. Tiež však mohlo spôsobiť nepresnosti zlé kalibrovanie kamery a tým sa nesprávne deformoval obraz.

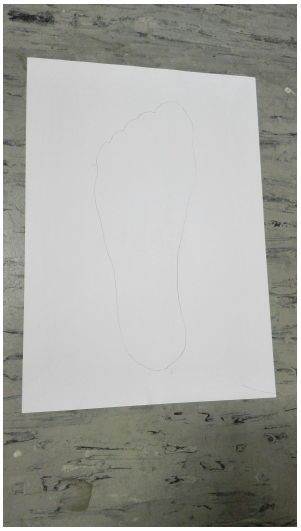
Niekedy sa stane, že páry bodov, z ktorých sa ráta matica homografie sú zle priradené k sebe čo deformuje obraz napríklad otočením o 180°alebo zrkadlovým zobrazením obrazu ako to môžeme vidieť na obr. 5.2.

Obr. 5.1: Zobrazenie pohľadu kamery ako správny zdroj papiera pre porovnanie.

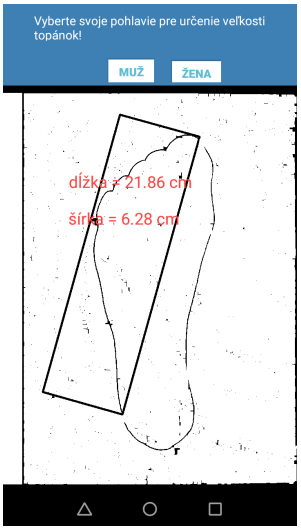


Obr. 5.2: Papier po homografií s nesprávnym otočením.

Taktiež ak je chodidlo obkreslené príliš jemnou čiarou, aplikácia ho nedokáže správne nájsť, ako to vidíme na obr. 5.4, pretože v obryse vznikajú príliš veľké medzery, ktoré sa už nedajú spojiť morfologickými operáciami.



Obr. 5.3: Fotka príliš slabého obrysu chodidla.



Obr. 5.4: Nesprávne nájdenie chodila spôsobené slabým obrysom.

Kapitola 6

Ďalšia práca

Počas práce na tejto aplikácií nám napadli viaceré podnety na zlepšenie merania. Pre nedostatok času sme sa nestihli dostať ku všetkým bodom, preto ich spomeniem nižšie ako inšpiráciu do budúcej práce.

Pre urýchlenie behu aplikácie by sme mohli zredukovať počet pixelov v obrázku. S menej obrazovými bodmi by výpočty pre spracovanie obrazu prebehli rýchlejšie. Mohlo by sa však stať, že užívateľ si obkreslí chodidlo príliš tenkou čiarou a na kamere by to bolo v šírke len jedného pixela. Zredukovaním počtu pixelov by sa teda mohlo stať, že sa táto informácia o čiare stratí a vznikla by tak medzera v obryse chodidla. To by mohlo viesť k nepresnostiam v metóde na hľadanie chodila a teda by sme nedospeli k správnemu výsledku.

Aby sme vedeli presnejšie určiť odchýlku merania našej aplikácie, mohli by sme použiť špeciálne meradlo chodila. V súčasnosti obrys na kontrolu správnosti merania aplikácie fyzicky meriame pomocou pravítka a tak môže dôjsť k rozdielu medzi nameranými hodnotami nie len výchylkami merania aplikácie, ale aj kvôli nepresnému meraniu kontrolnej hodnoty.

Keďže majú podľa amerického číslovania ženy iné veľkosti ako muži pri rovnakých centimetroch, bolo by zaujímavé zistiť, či sa dá rozlíšiť ženská noha od mužskej len vďaka obrysu chodidla, napríklad podľa pomeru dĺžky chodidla ku šírke.

Ľudia, ktorí skúšali našu aplikáciu, mali relevantnú pripomienku a to, že či je nutné chodidlo obkreslovať. Čas vynaložený na nájdenie čistého papiera, fixky a obkreslenia chodidla môže byť odradzujúci od používania takejto aplikácie. Pri zisťovaní rozmerov chodidla z fotografie priamo chodidla užívateľa však nastáva problém, že lýtko zakrýva vždy určitú časť obrazu preto by bolo komplikované hľadať tieto rozmery. Určitým riešením by však bolo, ak by sa papier nachádzal pri stene, užívateľ by priložil pätu k stene a zvyšok chodila by mal na papieri. Pri odfotografovaní chodila zvrchu by takto bolo vidieť aj najširšiu časť chodidla a keďže by sme vedeli, že chodidlo začína hneď na kraji papiera, vedeli by sme vypočítať aj dĺžku chodidla z daných informácií.

Záver

Podarilo sa nám vytvoriť aplikáciu, ktorá skutočne dokáže z jednej fotografie zistiť dĺžku a šírku chodidla. Táto fotografia však musí obsahovať známy predmet, ktorý leží v rovnakej rovine ako skúmaný objekt, podľa ktorého relatívne určíme veľkosť chodidla.

Ako známy objekt sme použili papier A4, ktorý má rozmery $21cm * 29,7cm$. Na tento papier užívateľ obkreslí svoje chodidlo, čím zároveň zaručí, že bude ležať v rovnakej rovine. Následne ho zosníma našou aplikáciou, ktorá z poskytnutých informácií vyráta dĺžku a šírku chodidla. Na dosiahnutie presnejších výsledkov má užívateľ možnosť využiť kalibráciu kamery. Pre použitie kalibrácie v našej aplikácií sme potrebovali získať kamerovú maticu a koeficient skreslenia. Rozhodli sme sa využiť aplikáciu, ktorá bola súčasťou knižnice OpenCV4Android využívanej pri tvorbe našej aplikácie. Preto okrem našej aplikácie na meranie rozmerov chodila prikladáme v prílohe aj aplikáciu na kalibrovanie kamery. Tieto aplikácie dokážu medzi sebou komunikovať a preto po kalibrácií kamery má užívateľ našej aplikácie automaticky možnosť použiť kalibrovanú kameru na meranie chodidla.

Dôležitým faktom je tiež, že papier je biely a predpokladáme teda aj to, že je svetlejší ako jeho okolie. Vedeli sme preto správnym prahom získať binárny obraz, kde je hľadaný papier biely a jeho okolie čierne. Pomocou Houghovej transformácie sme získali štyri priamky lemujúce papier. Priesečníky týchto priamok sme použili na získanie matice homografie, aby sme vedeli transformovať papier na celú obrazovku, čím sme zabezpečili aj upravenie skreslenia chodidla spôsobené perspektívou. Takto zobrazenému chodidlu sme už vedeli určiť veľkosť, pretože už dokážeme previesť veľkosť z pixelov na centimetre vďaka známym rozmerom papiera.

Takto sme dokázali vytvoriť Android aplikáciu, ktorá dokáže zmerať veľkosť obrysu chodidla s presnosťou na 0,2 cm. Aplikáciu sme testovali na vzorke dvadsiatich rôznych chodidiel (viac v kapitole 5). Fotografie týchto obrysov a snímky obrazovky po odmeraní veľkosti chodidiel sú priložené v prílohe. Samozrejme sme počas tvorby prišli k mnohým nápadom na zlepšenie, no nestihli sme všetky zakomponovať do súčasnej verzie aplikácie. Tieto podnety sú bližšie opísané v kapitole 6.

Počas hľadania podobných prác a riešení sme nenarazili na podobnú aplikáciu, ktorá by automaticky merala veľkosť chodidla z fotografie. Preto si myslíme, že používame

zaujímavé a, dalo by sa možno povedať, aj jedinečné riešenie takéhoto problému.

Literatúra

[1] Developers: Understand the activity lifecycle. https://developer.android.com/guide/components/activities/activity-lifecycle.html. [cit. 2018-05-15].

[2] Edge detection tutorial. http://www.doc.gold.ac.uk/~mas02fl/MSC101/ImageProcess/edge.html. [cit. 2018-01-20].

[3] Flaticon: hand-gesture. https://www.flaticon.com/free-icon/hand-gesture_252038. [cit. 2018-05-05].

[4] Google play: On 3d-camerameasure. https://play.google.com/store/apps/details?id=com.potatotree.on3dcamerameasure&hl=sk. [cit. 2018-01-12].

[5] Google play: Opencv manager. https://play.google.com/store/apps/details?id=org.opencv.engine&hl=sk. [cit. 2018-04-15].

[6] Hough line transform. https://docs.opencv.org/3.3.1/d9/db0/tutorial_hough_lines.html. [cit. 2018-02-12].

[7] inov: Footwear. http://weightliftingshoeguide.com/wp-content/uploads/2016/01/inov-8-shoes-sizing-table.png. [cit. 2018-05-21].

[8] Jama: A java matrix package. https://math.nist.gov/javanumerics/jama/. [cit. 2018-04-15].

[9] Mathworks: Camera calibration. https://www.mathworks.com/help/vision/ug/camera-calibration.html. [cit. 2018-01-06].

[10] nerian: Circle grid camera calibration patterns. https://nerian.com/support/resources/patterns/. [cit. 2018-05-15].

[11] next: Size conversion. http://www.next.co.uk/shoes/childrens-size-guide/conversion-chart. [cit. 2018-05-21].

[12] Opencv. https://opencv.org/releases.html. [cit. 2018-02-15].

[13] Opencv 3.0.0. https://docs.opencv.org/java/3.0.0/. [cit. 2018-01-12].

[14] Opencv: About. `https://opencv.org/about.html`. [cit. 2018-01-12].

[15] Opencv: Camera calibration with opencv. `https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html`. [cit. 2018-01-08].

[16] Opencv: Morphological transformations. `https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html`. [cit. 2018-05-15].

[17] Anubhav Agarwal, CV Jawahar, and PJ Narayanan. A survey of planar homography estimation techniques. *Centre for Visual Information Technology, Tech. Rep. IIIT/TR/2005/12*, 2005.

[18] Erika Chin, Adrienne Porter Felt, Kate Greenwood, and David Wagner. Analyzing inter-application communication in android. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 239–252. ACM, 2011.

[19] Miloš Fabian. *Rozpoznávanie dopravných značiek*. 2012. Bakalárska práca.

[20] Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1106–1112. IEEE, 1997.

[21] Joseph Howse. *Android application programming with OpenCV 3*. Packt Publishing Ltd, 2015.

[22] Stanislav Jursa. *Detekcia hrán a rohov v obraze*. 2007. Bakalárska práca.

[23] Fabio Remondino and Clive Fraser. Digital camera calibration methods: considerations and comparisons. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5):266–272, 2006.

[24] Dr. Gerhard Roth. Homography. `http://people.scs.carleton.ca/~c_shu/Courses/comp4900d/notes/homography.pdf`. [cit. 2018-02-11].

[25] Roger Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987.

[26] Michal Vozný. *Detekcia ŠPZ v reálnom čase – aplikácia pre OS Android*. 2012. Bakalárska práca.

[27] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.