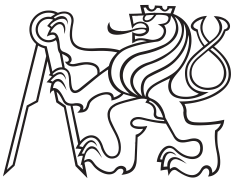
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Čtení ručkového měřicího přístroje pomocí kamery

Miroslav Běloch

Vedoucí: prof. Ing. Pavel Zahradník, CSc.
Obor: Kybernetika a robotika
Květen 2020

Poděkování

Děkuji vedoucímu práce panu prof. Ing. Pavlu Zahradníkovi, Csc., za ochotu a cenné připomínky k práci a k projektu, který předcházel. Také děkuji rodině a přátelům za vytvoření příjemného prostředí pro psaní.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.“

V Praze, 10. května 2020

Abstrakt

Tato práce navrhuje řešení pro čtení analogového měřicího přístroje pomocí kamery. Záměrem je, aby byl výsledek použitelný na různých typech měřičů. Program nejdříve detekuje region displaye, ve kterém následně hledá stupnici s ručičkou. Dále bylo naším cílem, aby byl systém do jisté míry odolný vůči vnějším vlivům jako posunutí kamery nebo snížení intenzity světla. Na závěr je návrh realizován a experimentálně ověřen pomocí platformy Raspberry Pi.

Klíčová slova: analogový měřicí přístroj, čtení, počítačové vidění, Raspberry Pi, rozpoznávání

Vedoucí: prof. Ing. Pavel Zahradník, CSc.
Katedra telekomunikační techniky FEL,
Technická 1902/2,
Praha 6

Abstract

This thesis proposes a solution for an analog measuring device reading using camera. The objective is for the system to be applicable on different types of panel meters. First it detects the region of the display, in which it then searches for the scale marks and the pointer. Eventually the reading is obtained by the comparison of scale marks and the pointer. Furthermore, the system is designed to be resistant to some extend to external influences such as device shifts, rotation or reduced illumination. The project is then implemented with Raspberry Pi and the accuracy is verified by the means of measuring of various scenarios.

Keywords: analog meter, computer vision, Raspberry Pi, reading, recognition

Title translation: Reading of an Measuring Device Using Camera

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

S nástupem digitálního věku dochází k postupnému přeměňování ručičkových přístrojů na číslicové. I přesto se však analogové přístroje nevytratily a pořád najdeme jejich uplatnění v nejrůznějších odvětvích průmyslu, v lékařství nebo v avionice. Důvodem jsou výborné čtecí vlastnosti – hodí se na místech, kde nepotřebujeme znát přesnou hodnotu, ale rychle potřebujeme oskenovat situaci. Také jsou lepším ukazatelem vývoje zkoumané veličiny – můžeme například pozorovat, jak rychle se mění. Z toho důvodu jsou analogové přístroje stále populární: příkladem je přístrojová deska v nejmodernějších automobilech, kde se na digitální obrazovce pořád replikují. Naopak nevýhodou je přesnost – lidské oči nejsou vždy srovnané s osou měřidla, případně od něho mohou být příliš daleko, a proto může docházet ke zkreslení přečtení hodnoty.

Občas máme k dispozici raritní analogový měřící přístroj, který dobře funguje, avšak by se hodilo zdigitalizovat jeho výstup. Takové hodnoty mohou posloužit ke zpracování výsledků nebo na řízení nějakého systému. Do takového staršího zařízení nemusí být vždy přístup nebo by naopak mohlo být příliš nákladné ho rozebírat a upravovat. Proto se jeví jako vhodný způsob použít technik zpracování obrazu. Měli bychom k dispozici nějakou základní kameru (postačí jakákoliv webkamera), kterou zapojíme do vestavěného mikropočítače. To pak můžeme opakovaně použít na snímání různých měřících přístrojů např. v laboratořích.

A právě zmíněná myšlenka je cílem této práce. Chceme navrhnout pokud možno robustní algoritmus, který s co nejmenšími úpravami přečte různé typy ručičkových zařízení. Zároveň chceme, aby zbylo na uživatele co nejméně práce. Proto navržené řešení myslí nejenom na detekci ukazatele, ale i na nalezení stupnice. Zároveň nám půjde i o funkci programu za nestandardních podmínek, jako je mírné natočení přístroje. Nejdříve v práci rozebereme již existující řešení a nastíníme náš přístup k problému. Ve další části podrobně popíšeme jednotlivé dílčí segmenty a metody. Dále popíšeme podrobně naši implementaci. V těchto dvou částech poskytneme názorné ukázky ze softwaru Matlab [1], v němž vznikal návrh. Nakonec návrh aplikujeme do vestavěného zařízení, experimentálně zhodnotíme a představíme výsledky měření.

1.1 Příbuzné projekty

V [2] se autoři snaží využít přečtení digitálního i analogového měřicího přístroje k jeho kalibraci. Proto je vyžadována velká přesnost. Kvůli tomu mají zkoumané zařízení v pevné předdefinované pozici. Pro analogový přístroj určí oblasti zájmu (ROI) jako dvě přímky – jedna koresponduje s nulovou výchylkou, druhá zase s maximální. Poté se najde ručka a vypočítá se měřená hodnota jako poměr úhlů mezi ručkou a oběma přímkami vynásobený celkovým rozsahem.

Zároveň uvádí, jak se vyhnout rigidnímu položení přístroje. Kamera pořídí při inicializaci dva snímky (při stejných podmínkách), jeden z nich musí být s ukazatelem v nulové pozici (slouží jako nulová reference – kamera nebo měřicí přístroj můžou být různě natočené). Následně se z nich získá obrázek, který je bez ukazatele, pomocí operátoru maximálního stupně šedi. Tento obrázek pozadí je uložen do paměti. Když je pořízen snímek s ručičkou v poloze, kterou chceme změřit, tak se využije algoritmu pro odečtení těchto dvou snímků a výsledkem je snímek pouze s ručkou. Dále se využije Houghovy transformace k nalezení parametrů přímky, ze kterých se již dopočítá úhel přímky svírající s osou x. Ten se pak porovnává s uloženou databází hodnot.

Detekcí tachometrů na palubní desce automobilu se zabývali v Pekingu [3]. Jejich úkol spočívá ve čtení současně dvou přístrojů – otáčkoměru a rychloměru. Kvůli tomu je umístěna kamera mezi tato měřidla a dochází k většímu projevu paralaxy. Jednotlivé značky symbolizující hodnotu rychlosti nebo otáček nejsou v očích kamery rozprostřeny rovnoměrně – jejich ve skutečnosti kružnicový obvod se zkresluje do elipsy. Díky tomu se musí jednotlivé dílky stupnice hledat zvlášť. K tomu je použito algoritmu K-means shlukování doplněného o statistický Z-test, aby se předešlo nežádoucímu šumu.

Na měřidla s čistě kruhovými ciferníky a důrazem na robustnost se zaměřili v tomto článku [4]. Nejprve se vyhledá ciferník pomocí metody rostoucího regionu, zároveň se tím získá jeho střed a poloměr. Poté se v nalezeném ciferníku hledá prstencová oblast se stupnicí porovnáváním stupňů šedi s využitím nalezeného poloměru a polárních souřadnic. Následně se detekují značky stupnice pomocí centrální projekce: prstencová oblast se binarizuje a každý černý pixel se spojí se středem, pak se nalezne úhel takové úsečky. Velký počet úhlů v úzkém úhlovém rozmezí (ideálně pořád ten stejný) znamená přítomnost značky s tímto úhlem. Poté se hledá ručička. Protože ručička nenabývá tvaru přímky, ale spíše zploštělého trojúhelníku, tak se nejdříve detekují hrany trojúhelníku pomocí Cannyho operátoru a poté se najde úsečka spojující střed a průsečík dvou přímek – hran ručičky. Následně se vypočítá indikovaná hodnota analogového přístroje podle úhlu ručičky a nalezených značek stupnice. Protože mají detekované značky vzestupné hodnoty a vždy se inkrementují o jedna, tak stačí přidělit hodnotu první a poslední značce. Kdyby tomu tak nebylo, tak se musí přidělit každé. Tento postup si dokáže poradit i pokud jsou značky rozprostřeny na stupnici nerovnoměrně.

V další studii [5] byla pozornost upřena ke čtení analogového multimetru. Myslelo se na to, že má multimetr přirozeně více funkcí. Nejdříve se obrázek pořízený z kamery otočí rotací okolo středu rozpoznané části multimetru podle předem daného vzoru. Zároveň se detekuje selektor funkcí a získá se jeho úhel. Následně se podobně rozpozná region obsahující stupnici a ručičku. V něm se najde počátek, ze kterého vychází ručka. Všechno toto rozpoznávání bylo

provedeno pomocí speciálních technik rozpoznávání podle vzoru založených na hledání hran a na gradientu. Poté se hledá ručka pomocí skenovací přímky vycházející z počátku. Každá taková přímka vytvoří akumulaci intenzit pixelů. Za ideálních podmínek by platilo, že úhel skenovací přímky s nejnižší akumulací přísluší ručičce, ale vzhledem k tomu, že se na číselníku objevuje mnoho další grafiky s nízkou intenzitou (např. čísla) a světelné podmínky nemusí být ideální, tak to nemusí vždy platit. Proto je vytvořen tzv. profil gradientu akumulací GA. V podstatě, kde je největší změna akumulace, tam je ručka. Nakonec se dopočítá hledaná hodnota pomocí mapovací funkce, která bere ohled na funkci selektoru a úhel ručičky. Výhoda tohoto postupu je velmi dobrá přesnost, ovšem nevýhoda může být v tom, že pro každý jiný multimetr (nebo obecně měřidlo) se musí nejprve vytvořit vhodné vzory pro rozpoznávání. Také se musí pro nelineární případ stupnice (měření odporu) naměřit většina hodnot pomocí úhloměru (a zbytek dopočítat), což není příliš robustní.

Kapitola 2

Analýza problému a řešení

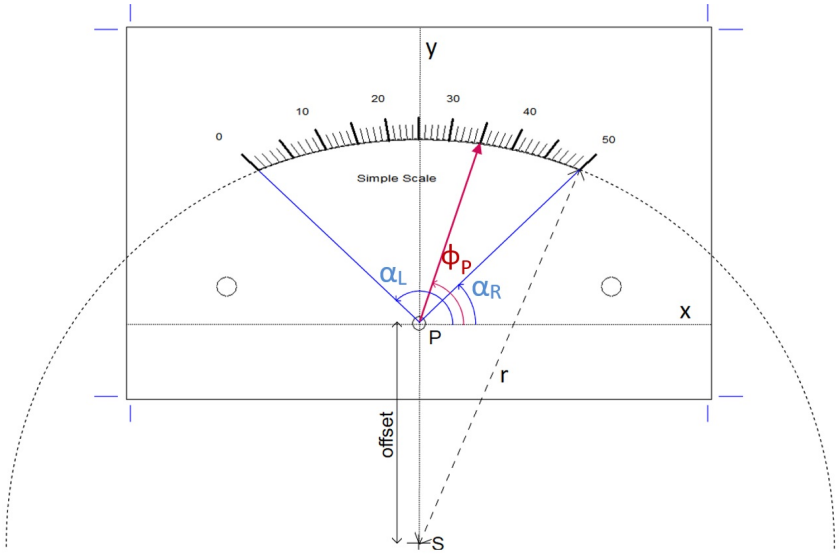
2.1 Rozbor obrazovky

Ještě předtím, než se pustíme do návrhu algoritmu, tak je potřeba si podrobněji představit strukturu displaye a vysvětlit rozdíly mezi dvěma základními typy, na které se dělí. Prvním typem jsou tzv. klasické (jednoduché) měřicí přístroje (Obrázek 2.1). Kruhová výseč, která je dána dvěma koncovými značkami na stupnici, většinou přísluší pravému úhlu, tedy $\alpha_L - \alpha_P = 90°$, při návrhu řešení budeme počítat s tím, že to tak nemusí být vždy. Ručička vychází z bodu, který se nachází pod stupnicí a je středem zmíněné kruhové výseče. Tomuto bodu budeme dále v práci říkat průsečík P, neboť se v něm pomyslně sbíhají (protínají) všechny značky na stupnici. Ručička φ_P se v analogovém přístroji pohybuje od levého konce α_L k pravému konci výseče α_R, kde většinou značce s α_L náleží nulová hodnota. Sluší se poznamenat, že všechny úhly, které používáme, jsou svírané daným objektem a kladným směrem osy x.

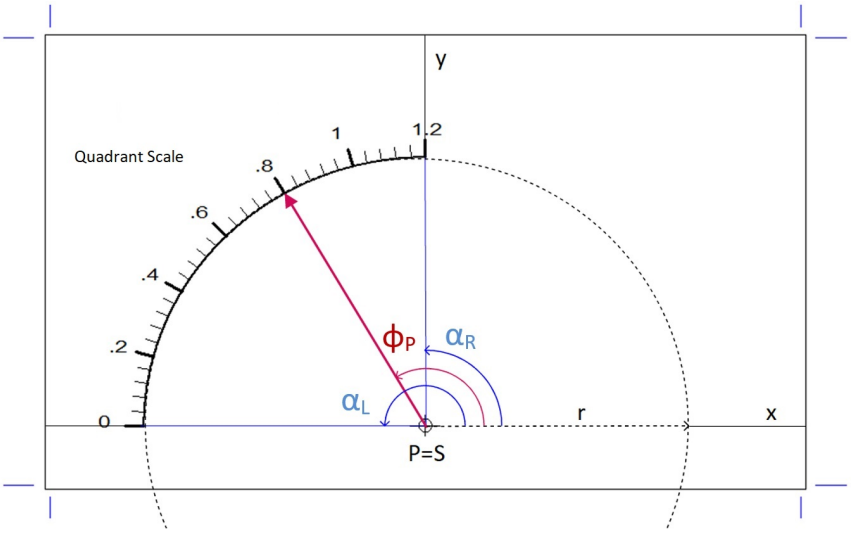
Klasické analogové přístroje se vyznačují tím, že střed oblouku S, na kterém se nachází stupnice, se neshoduje s průsečíkem P. Je posunutý o určitý offset. U moderních přístrojů se to dělá pro lepší čitelnost a přehlednost na displayi. Vždy je však posunut jen vertikálně, tedy souřadnice x je totožná u S i u P. Tyto dva body budeme oba potřebovat, střed v souvislosti s hledáním oblouku se stupnicí a hledáním všech mezikruží a průsečík v souvislosti s přiřazováním hodnot značkám na stupnici.

Druhým typem jsou kvadrantové přístroje. Je znázorněn na obrázku 2.2. Rozpětí stupnice bývá taktéž $\alpha_L - \alpha_P = 90°$. Střed rotace, ve kterém je upevněna ručička, se nachází v pravém dolním rohu displaye. V tomto případě střed rotace (průsečík) splývá se středem kružnice, jejíž součástí je oblouk, na kterém je položena stupnice. Tedy poloměr této kružnice je shodný s délkou ukazatele. Všimněme si, že jednotlivé značky na stupnici tvoří normály této kružnice.

Cílem práce bude, aby navržené řešení bylo funkční pro oba typy přístrojů.



Obrázek 2.1: Popis klasického displaye



Obrázek 2.2: Popis kvadrantového displaye

2.2 Navržený algoritmus

Vykonávání programu můžeme rozdělit na dvě části. První část je sekvencí na sebe navazujících kroků vedoucích k zjištění základních údajů a informací o měřicím přístroji. Zde se předpokládá, že je výchylka pointeru nulová a nehýbe se. V té druhé, aktivní části, se již pointer hýbat může. Jedná se o iteraci, která skončí, jakmile to vyžaduje uživatel. Vývoj programu je znázorněn na obr. 2.3.

Nejprve je nutné získat vzorový snímek, na kterém bude zobrazena oblast zájmu (dále

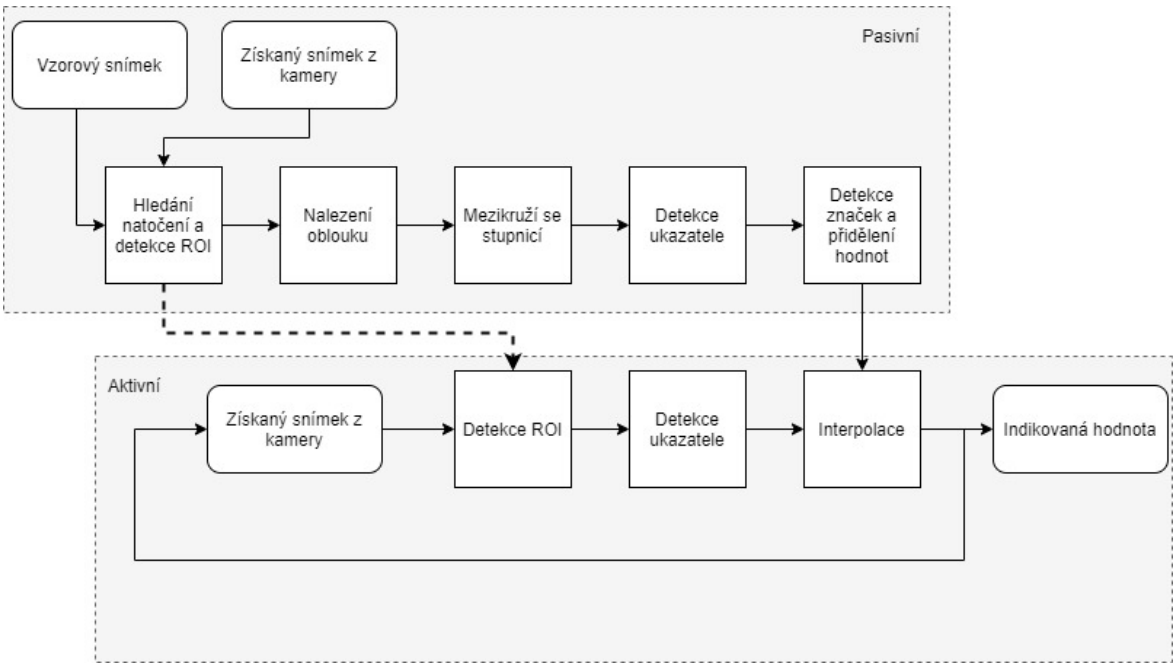
ROI), což je display měřicího přístroje. Ten obdržíme vyfocením scény s měřicím přístrojem a ručičkou nacházející se v nulové pozici. Poté ho manuálně ořízneme v grafickém editoru (např. GIMP), pokud možno převedeme na odstíny šedi a uložíme. Nyní nám tento vzorový snímek poslouží jako reference k nalezení ROI nově vyfocených scén.

V následujícím kroku již máme k dispozici vystřižený a narovnaný snímek, se kterým se bude dál pracovat. Nyní z důvodu nalezení stupnice a pointeru potřebujeme najít oblouk, na kterém se stupnice nachází. To provedeme pomocí kruhové Houghovy transformace.

Dále bude potřeba, abychom nalezli výseč mezikruží, ve kterém najdeme stupnici. Poloměr jedné z kružnic mezikruží jsme nalezli v předchozím kroku, poloměr té druhé najdeme pomocí poměrové analýzy bílých a černých pixelů v měnícím se mezikruží. Úhel výseče je z jedné strany dán ukazatelem, proto ho je potřeba detekovat ještě před hledáním stupnice.

Nakonec ve výseči mezikruží budeme zkoumat natočení jednotlivých značek a přiřadíme jim hodnoty.

Nyní již algoritmus přechází do aktivní části. Je to smyčka složená z podobných kroků zmíněných výše jen s tím rozdílem, že již víme, kde v nově pořízeném snímku hledat ROI. Také už znovu nehledáme stupnici. Na závěr iterace vypočítáme z úhlu pointeru a dříve nalezené stupnice indikovanou hodnotu.



Obrázek 2.3: Vývojový diagram navrženého řešení

2.3 Omezení algoritmu

Pro úspěšnou detekci je vyžadováno několik podmínek. Měřicí přístroj by měl být dobře viditelný, tedy měl by být dostatečně osvětlený. Implementace do jisté míry umožňuje proměnlivé osvětlení, které může nastat při denní změně osvětlení místnosti nebo náhlou přítomností jemného stínu. Okna získaná při detekci rohů jsou totiž normalizována a v dalším průběhu programu se vystřižený obrázek prahuje pomocí Otsuovy metody. Pokud je naopak obrázek přesvětlený, resp. nevhodně osvětlený, může docházet k odrazům od skla displaye, resp. se může objevit stín vržený ukazatelem.

Algoritmus předpokládá, že se jedná o klasický nebo kvadrantový typ přístroje, což je v drtivém množství dostupných měřicích přístrojů splněno. Nicméně je třeba poznamenat, že se programu musí dát dopředu vědět, o jaký typ jde. Dále se musí zadat maximální hodnota, naše implementace není vybavena rozpoznáváním čísel na stupnici. Může se stát, že některé další konstanty bude třeba v programu upravit, ty důležité jsou k dispozici s popiskem ve vstupní funkci *main*.

Dále je důležité, aby snímací zařízení mělo dostatečné rozlišení, případně bylo v přijatelné blízkosti. Pokud bude rozlišení příliš velké, může to do značné míry zpomalit běh programu či dokonce pro nedostatek paměti vestavěného zařízení program ukončit. Jako optimální se jeví být rozlišení HD 1280 x 720, které je dnes dostupné snad ve všech kamerách.

Programu svědčí, pokud je na displayi co nejméně pro měření nedůležitých grafických prvků, přesto by si s tím algoritmus měl poradit, i když třeba za cenu časové náročnosti. Je také potřeba, aby grafické prvky byly k pozadí pokud možno co nejvíce kontrastní, nejlepší je tedy černá barva stupnice a pointeru na bílém pozadí (případně naopak). Také je vhodné, aby se na pozadí snímku vyskytovalo co nejméně rušivých elementů. Přesto, pokud je kamera dobře zaostřená na analogové zařízení, tak si s tím program poradí.

Při kalibraci (pasivní fáze programu) musí být ručička v klidové nulové pozici. Pohybovat se může až v následující aktivní smyčce, kdy se začnou vypisovat vypočítané hodnoty. Nakonec se vyžaduje, aby se s přístrojem či kamerou při měření nehýbalo.

Kapitola 3

Teoretické pozadí

3.1 Derivace v obraze

Ještě před popsáním složitějších konceptů, které budou v práci potřeba, bychom měli vysvětlit pojem derivace v obraze [6]. Obraz jako takový je 2D matice, jejíž funkcí je intenzita. Definiční obor této funkce je diskrétní (souřadnice jsou celá čísla). Z tohoto důvodu je derivace obrázku spíše aproximací derivace. Říká nám, jak se mění intenzita v jednotlivých pixelech. Díky tomu je velmi užitečná při detekci hran nebo rohů. Pokud bychom dosadili do definice derivace ve spojitém případě za délku kroku 1, pak dostaneme:

$$\frac{df}{dx} = f(x+1) - f(x). \tag{3.1}$$

Ukazuje se, že přesnější je použití tzv. středové diference, která je definovaná jako

$$\frac{df}{dx} = f\left(x+1\right) - f\left(x-1\right), \tag{3.2}$$

to by znamenalo, že pokud bychom derivovali ve směru x, tak derivace v bodě by byla rozdílem intenzit pixelů napravo a nalevo. V praxi se však používá Prewittův operátor. Ten používá masku (jádro), která je pro derivace ve směrech x a y dána jako:

$$K_x = \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \qquad K_y = \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.3}$$

Obrázek je konvoluván s maskou – tedy pro každý pixel v obrázku, který odpovídá prostřednímu prvku masky, se vynásobí pixely v jeho okolí s číslem v masce a tyto součiny se sečtou. Díky tomu se redukuje nežádoucí šum, který by jinak mohl derivaci významně ovlivnit.

3.2 Harrisův detektor rohů

Při návrhu metody pro rozpoznávání obecně si musíme stanovit, jaký objekt a za jakých okolností chceme detekovat. Jelikož každý měřící přístroj (který můžeme použít, viz Omezení

algoritmu) obsahuje detaily jako stupnici, čísla a případně další popisky, tak se jako dobré řešení pro detekci jeví rozpoznání pomocí lokálních vlastností. Těmi jsou rohy – to jsou zajímavé body v tom smyslu, že mají jasně definovanou polohu v obrázku a jsou do jisté míry odolné vůči změnám osvětlení a jiným nežádoucím vlivům [7]. Rohy jsou také invariantní k natočení obrázku, což se nám hodí, protože detekce natočení je součástí implementace.

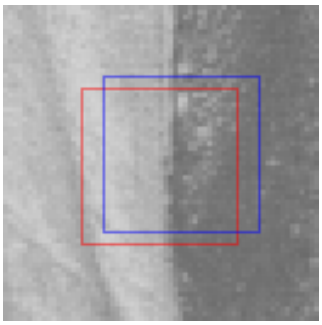
Hlavní myšlenka Harrisova detektoru rohů [8], [9] se zakládá na faktu, že v rohu dochází k prudkým změnám intenzit pixelů ve všech směrech, zatímco u hran je to pouze v jednom směru a u prázdné textury nedochází k změnám žádným. Na displayi přístroje by se jich vždy měl nacházet dostatek, protože zmíněné detaily se vyznačují rychlými změnami intenzit. V následujícím textu rozvedu, jak tuto myšlenku přeformulovat do matematické podoby.

Vycházíme z této matematické formule:

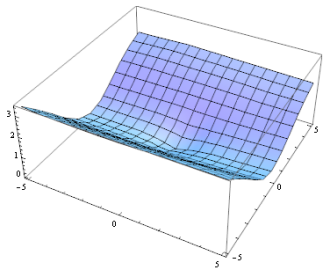
$$E(u,v) = \sum_{x,y} w(x,y) \left[I(x+u, y+v) - I(x,y) \right]^2 . \tag{3.4}$$

Funkce E vyjadřuje součet čtverců rozdílů intenzit, I je obrázek ve stupních šedi, ve kterém hledáme rohy a w je tzv. okenní funkce, která určuje, se kterou skupinou pixelů v obrázku počítáme a zároveň přiděluje pixelům váhu.

Mějme nějakou zkoumanou oblast znázorněnou červeným čtvercem na obr. 3.1a (reprezentující matici intenzit I). Nyní nás zajímá, jak se změní intenzita jednotlivých pixelů červeného okna, pokud se čtvercem posuneme o vektor (u, v). Posunuté okno je znázorněno modrou barvou. Pokud zde budeme oknem pohybovat vertikálně, tak se rozdíl intenzit moc nezmění, protože modré okno bude vypadat téměř totožně s červeným. Pokud okno posuneme horizontálně, tak se rozdíl čtverců zvětší. Jak je znázorněné na obr. 3.1b, tak funkce E má tvar údolí. To je charakteristické pro hrany.



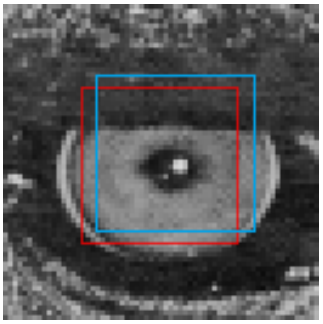
(a) : Posun okna [9]



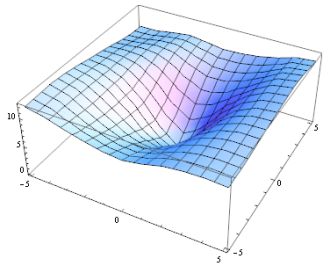
(b) : Funkce $E(u, v)$ [9]

Obrázek 3.1: Hrany

Nyní se podíváme (obr. 3.2a a obr. 3.2b), jak se funkce chová v okolí rohu. Pokud se s oknem posuneme libovolným směrem, tak se suma čtverců rozdílů intenzit mění rychle pro libovolný posun (u, v). Funkce E má tvar dolíku.



(a) : Posun okna [9](upraveno)



(b) : Funkce $E(u, v)$ [9]

Obrázek 3.2: Rohy

Pro upřesnění uvedu, že okenní funkce může být jednotková (použitá v příkladu výše)[9], tedy že v oblasti, na kterou se koukáme (červený čtverec), přidělí pixelům jednotkovou váhu a mimo tuto oblast váhu nulovou. Takové okno však není rotačně symetrické, což by nám dále působilo problémy při hledání natočení. Proto použijeme Gaussovo okno, které rotačně symetrické je a navíc klade důraz (přidělí největší váhu) na pixely okolo středu okna.

Nyní tedy máme funkci E. Mohli bychom teoreticky tuto funkci spočítat pro každý pixel a následně zkoumat jejich tvar, to je ale pro počítač zbytečně náročné. Ve skutečnosti nám stačí porovnávat tvar této funkce pouze v okolí sledovaného bodu a tady si pomůžeme Taylorovou aproximací funkce.

$$I\left(x+u, y+v\right) = I\left(x, y\right) + uI_x\left(x, y\right) + vI_y\left(x, y\right) + O\left(u^2, v^2\right), \tag{3.5}$$

kde I_x je obrázková diskrétní derivace podle x a podobně I_y podle y. $O\left(u^2, v^2\right)$ je notace pro chybu způsobenou zanedbáním druhého a vyšších řádů.

Dosadíme-li do rovnice 3.4, dostaneme v maticové formě

$$E\left(\boldsymbol{u}\right) \approx \sum_{\boldsymbol{x_0}} w\left(\boldsymbol{x_0}\right) \left[\left(I\left(\boldsymbol{x_0}\right) + \frac{\partial I\left(\boldsymbol{x_0}\right)}{\partial \boldsymbol{x}}^T \boldsymbol{u}\right) - I\left(\boldsymbol{x_0}\right)\right]^2. \tag{3.6}$$

Tedy

$$E\left(\boldsymbol{u}\right) \approx \sum_{\boldsymbol{x_0}} w\left(\boldsymbol{x_0}\right) \left[\frac{\partial I\left(\boldsymbol{x_0}\right)}{\partial \boldsymbol{x}}^T \boldsymbol{u}\right]^2 \tag{3.7}$$

a

$$E\left(\boldsymbol{u}\right) \approx \sum_{\boldsymbol{x_0}} w\left(\boldsymbol{x_0}\right) \boldsymbol{u}^T \frac{\partial I\left(\boldsymbol{x_0}\right)}{\partial \boldsymbol{x}} \frac{\partial I\left(\boldsymbol{x_0}\right)}{\partial \boldsymbol{x}}^T \boldsymbol{u}. \tag{3.8}$$

Nechť

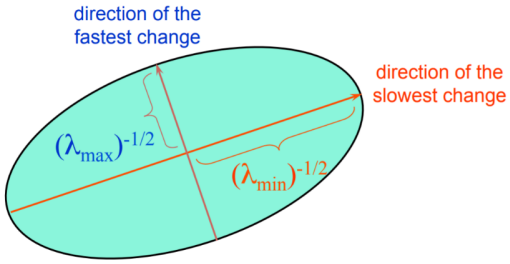
$$M = \sum_{\boldsymbol{x_0}} w\left(\boldsymbol{x_0}\right) \frac{\partial I\left(\boldsymbol{x_0}\right)}{\partial \boldsymbol{x}} \frac{\partial I\left(\boldsymbol{x_0}\right)}{\partial \boldsymbol{x}}^T$$

$$= \sum_{x_0, y_0} w\left(x_0, y_0\right) \begin{bmatrix} I_x\left(x_0, y_0\right)^2 & I_x\left(x_0, y_0\right) I_y\left(x_0, y_0\right) \\ I_x\left(x_0, y_0\right) I_y\left(x_0, y_0\right) & I_y\left(x_0, y_0\right)^2 \end{bmatrix}, \tag{3.9}$$

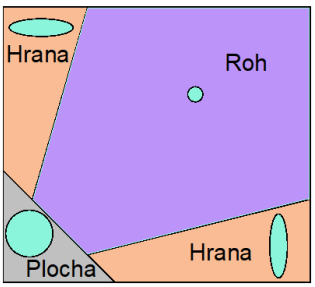
díky tomu můžeme napsat

$$E(\boldsymbol{u}) \approx \boldsymbol{u}^T M \boldsymbol{u}. \tag{3.10}$$

Tedy jedná se o kvadratickou formu, díky které můžeme dobře definovat výše zmíněné tvary dolíku nebo údolí. Provedeme-li řez této formy, tak dostaneme elipsu (jednou z vlastností kvadratické formy je, že nabývá extrému v bodě $\mathbf{0}$ – v našem případě jde o minimum), jejíž tvar je určen vlastními čísly matice M (znázorněno na obr. 3.3). Pokud jsou obě vlastní čísla malá, tak se nejedná o roh ani hranu. Pokud je jedno vlastní číslo velké a druhé malé, tak to značí přítomnost hrany. Pokud jsou obě velká, tak jde o roh (obr. 3.4).



Obrázek 3.3: Řez kvadratickou formou [8] (upraveno)



Obrázek 3.4: Vliv vlastních čísel na detekci [8] (upraveno)

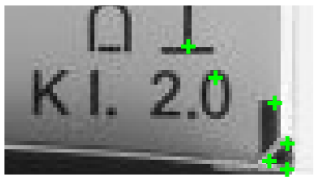
Nyní potřebujeme matematicky vyjádřit jedním číslem, jak moc se zajímavý bod zdá být rohem. Tím je Harrisova funkce odezvy na roh:

$$R = \det(M) - k \cdot tr(M)^2, \tag{3.11}$$

kde $\det(M)$ je determinant M, $tr(M)$ je stopa M a k je konstanta v intervalu $< 0.04, 0.06 >$.

Jako lepší volba (dokazuje obrázek 3.5) se ukazuje použít odezvu od Shi a Tomasiho:

$$R = min\{\lambda_1, \lambda_2\}. \tag{3.12}$$



(a) : Harris

(b) : Shi-Tomasi

Obrázek 3.5: Porovnání detektorů

3.3 Nalezení odlehlých hodnot pomocí mediánové absolutní odchylky

Pokud pracujeme s větším množstvím dat, tak se často potřebujeme zbavit hodnot, které se velmi liší od ostatních (v našem případě kvůli špatným detekcím). Protože se mění podmínky

měření (tedy natočení měřicího přístroje, světelnost nebo jiný měřicí přístroj), tak si nemůžeme empiricky zvolit interval, ve kterém, pokud se nenachází zkoumaná veličina, tak se jedná o odlehlou hodnotu. Každý takový ‚interval' je potřeba pro každý případ (měření) určit zvlášť.

Toho docílíme pomocí tzv. mediánové absolutní odchylky od mediánu (dále MAD). MAD je mírou statistického rozptylu užívaná k zjištění směrodatné odchylky [10]. Tuto techniku používáme z toho důvodu, protože není tolik citlivá na přítomnost extrémních hodnot (na rozdíl od střední hodnoty). Je to tedy medián množiny obsahující rozdíly jednotlivých dat od jejich mediánu v absolutní hodnotě

$$MAD = b \cdot med\left(|x - med\left(x\right)|\right), \tag{3.13}$$

kde $\boldsymbol{x}$ jsou data a b je škálovací konstanta, jejíž hodnota závisí na rozdělení. Pro normální rozdělení je to $b = 1.4826$.

Za odlehlé hodnoty (anglicky outlier) považujeme data, pro která platí, že jsou od mediánu vzdáleny více než tři MAD. Tedy

$$\frac{|\boldsymbol{x} - med\left(\boldsymbol{x}\right)|}{MAD} = \begin{cases} \leq 3 & x \text{ není outlier} \\ > 3 & x \text{ je outlier} \end{cases} \tag{3.14}$$

3.4 Kruhová Houghova transformace

Houghova transformace obecně je technika k nalezení matematicky popsatelných objektů pomocí tzv. hlasování v prostoru parametrů. Naším předmětem bude detekce kružnic, proto se zde budeme zabývat speciální kruhovou Houghovou transformací (dále KHT).

Kružnici lze definovat pomocí rovnice:

$$(x - a)^2 + (y - b)^2 = r^2, \tag{3.15}$$

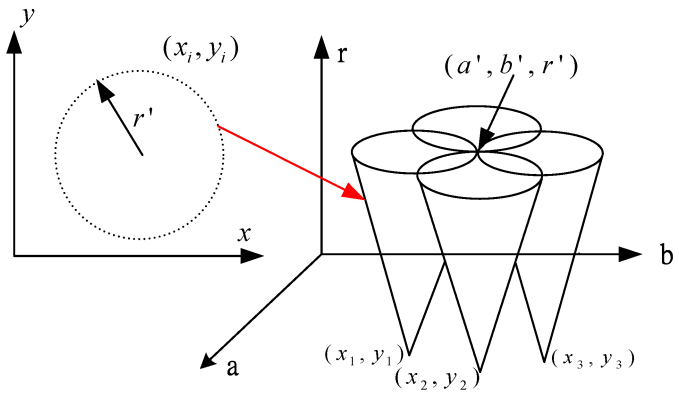
kde (a, b) je střed kružnice a r je její poloměr. Proměnné (x, y) značí body na obvodu kružnice. Její parametrické vyjádření pro $\alpha \in < 0, 2\pi >$ (tedy pokrytí celého obvodu) je

$$\begin{cases} x = a + r \cdot \cos\left(\alpha\right) \\ y = b + r \cdot \sin\left(\alpha\right) \end{cases} \tag{3.16}$$

KHT funguje na následujícím principu [11], [12]. Vstupními daty jsou binární body na potenciálním obvodu kružnice – tedy (x, y). Nyní záleží, jestli je znám poloměr hledané kružnice. Pokud ano (pojmenujme si ho R), tak se pracuje s dvourozměrným prostorem parametrů (a, b). Pro každý bod (x_0, y_0) se vytvoří kružnice o známém poloměru R. V bodě, kde se protnou zkonstruované kružnice, leží v geometrickém prostoru střed kružnice (a_0, b_0). Počet takových průsečíků si ukládáme do tzv. akumulátoru, který má rozměry prostoru parametrů. Lokální maxima s vysokými hodnotami poukazují na přítomnost kružnice.

V reálných aplikacích ovšem většinou poloměr neznáme přesně – známe ho jen přibližně. V tom případě KHT funguje nad 3D prostorem parametrů (a, b, r) a velmi se zvyšuje výpočetní i

paměťová náročnost. Pro každý bod (x, y) v geometrickém prostoru vnikají kužele v prostoru parametrů (znázorněno na obrázku 3.6). Analogicky k 2D případu: v bodě (a, b, r), kde se protne nejvíce povrchů kuželů, se nachází parametry (souřadnice středu a poloměr) hledané kružnice.



Obrázek 3.6: Každý bod z geometrického prostoru (vlevo) vytvoří kužel v parametrickém prostoru (vpravo) [13]

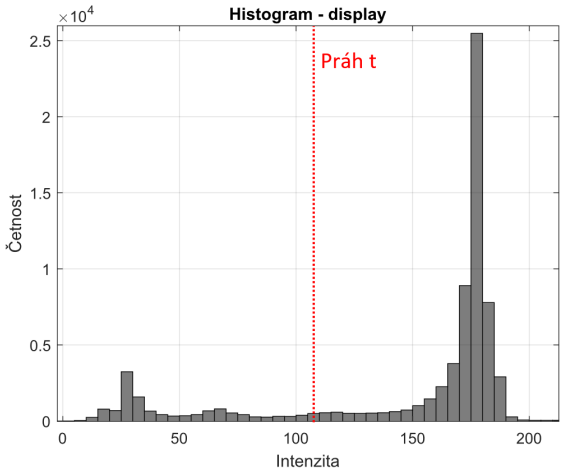
3.5 Otsuovo prahování

V některých krocích našeho algoritmu bude potřeba pracovat s binárním obrázkem. Ten se skládá ze dvou hodnot – nuly reprezentují černé pixely (jinak také ‚pozadí') a jedničky představují pixely bílé (tzv. ‚objekt' nebo ‚popředí'). Prahování funguje na tom principu, že je zvolena prahová hodnota a v závislosti na tom, jestli jsou intenzity obrázku (v odstínech šedi) pod ní nebo nad ní, tak se jim přidělí třída – binární hodnota. Jak ovšem zvolit správně mezní hodnotu? Jejím automatickým nastavením se zabývá Nobuyuki Otsu [14].

Otsuova metoda zkoumá rozdělení intenzit pixelů ve dvouvrcholovém histogramu. Za ideálních podmínek by měl oba vrcholy výborně rozeznatelné (s co nejmenší směrodatnou odchylkou) s hlubokým údolím mezi nimi a bez šumu. Pak bychom jako práh zvolili intenzitu s nejmenší četností mezi nimi. Ve skutečnosti je však běžným jevem, že v histogramu oba vrcholy najdeme jen s notnou dávkou představivosti (viz obrázek 3.7).

Myšlenka spočívá v tom, že se zvolí taková mezní hodnota, která rozdělí obrázek na dvě třídy, aby hodnoty pixelů v obou třídách měly co nejmenší odchylku od příslušných středních hodnot. Přesněji řečeno, metoda prochází přes všechny možné prahy, a počítá rozptyl obou částí histogramu, které se sečtou. Tento součet chceme minimalizovat.

Počet pixelů (hodnot intenzit) v první třídě je $N_0 = n_0 + n_1 + \cdots + n_t$ a ve druhé $N_1 = n_{t+1} + n_{t+2} + \cdots + n_{L-1}$. Prahová hodnota je značena písmenem t a L značí počet sloupečků histogramu (úrovně intenzit – např. pro 8bitový obrázek je $L = 256$). Četnost intenzity ve sloupečku i je značeno n_i. Celkový počet pixelů je $N = N_0 + N_1$.



Obrázek 3.7: Histogram – vrchol vlevo není tak rozeznatelný jako vrchol vpravo

Střední hodnoty obou tříd μ_0 a μ_1 získáme takto:

$$\mu_0 = \frac{1}{N_0} \sum_{i=0}^{t} n_i i, \tag{3.17}$$

$$\mu_1 = \frac{1}{N_1} \sum_{i=t+1}^{L-1} n_i i. \tag{3.18}$$

Nyní můžeme vypočítat rozptyly:

$$\sigma_0^2 = \frac{1}{N_0} \sum_{i=0}^{t} (i - \mu_0)^2 , \tag{3.19}$$

$$\sigma_1^2 = \frac{1}{N_1} \sum_{i=t+1}^{L-1} (i - \mu_1)^2 . \tag{3.20}$$

Váhy pro obě třídy pak tímto způsobem:

$$\sigma_w^2 (t) = \omega_0 (t) \sigma_0^2 (t) + \omega_1 (t) \sigma_1^2 (t) , \tag{3.21}$$

tedy

$$t^* = argmin\ \sigma_w^2 (t) . \tag{3.22}$$

Pro úplnost je třeba uvést, že v praxi se tato úloha převádí na úlohu k ní ekvivalentní, kde se maximalizuje tzv. rozptyl vnější třídy (anglicky ‚between-class variance‘)

$$\sigma_B^2 (t) = \omega_0 (\mu_0 - \mu)^2 + \omega_1 (\mu_1 - \mu)^2 , \tag{3.23}$$

kde μ je střední hodnota intenzit všech pixelů.

3.6 Binární morfologie

Jedním z důležitých principů ve zpracování obrazu je tzv. binární morfologie. Jedná se o sadu algoritmů zpracovávajících obraz, hodících se např. pro odstranění šumu, nebo naopak pro zvýraznění objektů. Morfologickou transformací rozumíme pohybování se tzv. strukturním elementem SE (také jako ‚jádro') přes celý obraz (podobně jako při konvoluci). Ten je většinou malou maticí o rozměrech 3 x 3, může být i větší. Existují různé tvary SE, nejčastěji jsou elementy izotropické, tedy mají ve všech směrech stejnou hodnotu [15]. Na pozici definovaného počátku (většinou uprostřed SE) je pak hodnota získána podle následujících kritérií [16].

V morfologické erozi dostaneme na místě počátku logickou 1, pokud všechny pixely pod jádrem jsou logické hodnoty 1. Matematicky to lze vyjádřit jako minimum z hodnot pod jádrem.

Naopak v morfologické dilataci bude na místě počátku logická 1, pokud alespoň jeden pixel pod jádrem je 1. Tedy jinak řečeno se jedná o maximum z hodnot pod jádrem.

Obě tyto základní operace jsou k sobě duální. Eroze ztenčuje popředí a zvětšuje pozadí a naopak dilatace objekt zvětšuje a zmenšuje pozadí. Z těchto dvou operací se dají vytvořit složitější, které jsou jednoduše jejich kombinací. I ty zde uvedu, neboť budou dále v práci použity.

Při erozi se sice zbavíme šumu, ale zároveň se zmenší i hledaný objekt. Proto se přišlo s binárním otevřením, které spojuje erozi s následnou dilatací. Ta dokáže oddělit objekty spojené úzkým objektem. Také odstraní malé výběžky.

Naproti tomu binární otevření je dilatace následovaná erozí. Její vlastností je schopnost zacelit štěrbiny a spojit objekty nacházející se blízko u sebe. Morfologické uzavření i otevření jsou idempotentní, tedy použijeme-li uzavření (nebo otevření) vícekrát za sebou, bude to mít stejný efekt, jako bychom je použili jen jednou.

Kapitola 4

Implementace

4.1 Detekce displaye a jeho natočení

Proto, abychom mohli hledat objekty na obrazovce přístroje, tak je dobrý nápad nejdříve detekovat samotnou obrazovku. Tím omezíme rušivé elementy v pozadí přístroje (případně na samotném přístroji mimo obrazovku), díky kterým by program s největší pravděpodobností selhal.

Detekci displaye tedy provedeme následujícím způsobem. Identifikujeme, kde se nachází rohy ve vzorovém obrázku a v pořízeném snímku. Z těchto nalezených zajímavých bodů vybereme 50 nejlepších podle rohové odezvy (mají ji nejvyšší). Poté kolem těchto rohů vytvoříme kruhová okna. Najdeme orientaci každého z nich pomocí gradientu. Okna ze vzoru porovnáme s okny z obrazu a vybereme ta, která se navzájem nejvíce podobají. Následně provedeme test nejbližších sousedů pro potlačení falešných detekcí.

Pro další pracování s obrazem je potřeba ho nejdříve převést z barevného modelu RGB do odstínů šedi. To je provedeno váženým součtem jednotlivých složek barevného obrázku

$$I_g = 0.2989R \ + \ 0.5870G \ + \ 0.1140B. \tag{4.1}$$

Tím získáme 2D matici intenzit I_g.

4.1.1 Detekce rohů

Nyní pro výpočet rohů potřebujeme získat matici M (z rovnice 3.9). Ta je složena ze součtů produktů parciálních derivací I_g v Gaussově okně. Horizontální derivaci obrázku obdržíme konvolucí funkce I_g s jádrem K_x (3.3) a podobně vertikální derivaci získáme konvolucí s jádrem K_y (3.3):

$$I_x = I_g * K_x = \ I_g * \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \tag{4.2}$$

$$I_y = I_g * K_y = I_g * \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}. \tag{4.3}$$

Nyní vypočítáme prvkové součiny jednotlivých parciálních derivací a tyto konvolujeme s Gaussovým jádrem G, ve kterém je každá hodnota přibližně (kvůli diskretizaci) dána formulí

$$f(x,y) \approx \frac{1}{\sigma^2 2\pi} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{4.4}$$

kde σ je směrodatná odchylka nastavená na třetinu velikosti konvolučního okna, x a y jsou pozice pixelů jádra. Máme tedy čtvercovou symetrickou matici M[1] , označme si její vypočítané prvky takto:

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}, \tag{4.5}$$

kde[2]

$$A = (I_x \circ I_x) * G, \quad B = (I_y \circ I_y) * G, \quad C = (I_x \circ I_y) * G. \tag{4.6}$$

Teď můžeme najít vlastní čísla matice M (a tedy rohové odezvy vybráním menších vlastních čísel – viz 3.12). Ta jsou řešením jejího charakteristického polynomu:

$$\lambda_{min} = R = \frac{A+B}{2} - \frac{\sqrt{(A+B)\circ(A+B) - 4(A\circ B - C\circ C)}}{2}. \tag{4.7}$$

V dalším kroku vybereme 50 rohů s největší rohovou odezvou R. Toto číslo bylo zvoleno tak, abychom detekovali dostatečné množství rohů, ale na druhou stranu se vyvarovali falešných shod (odpovídajících si rohů), protože s rostoucím počtem klesá jejich kvalita. Při výběru rohů je myšleno na to, aby se dva rohy nenacházely téměř u sebe – může se jednat o tentýž roh. Proto je u každého rohu ověřováno, že jeho odezva je lokální maximum z okolních (8-konektivita) potenciálních rohů (znázorněno na obr. 4.1 – v červeném rámečku jsou vybrané rohy).

0	0	0	0	0
0	6	0	3	0
0	4	0	0	0
0	0	7	0	0
0	0	0	0	0

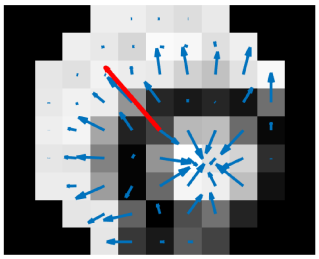
Obrázek 4.1: Ilustrace výběru lokálního maxima rohové odezvy při 8-konektivitě

[1]Prvky A,B,C mají rozměry zkoumaného obrázku a obecně nemusí být čtvercové, proto ani M nemusí být čtvercová. Ovšem dále budeme provádět pouze prvkové operace mezi A,B,C, takže na ně nahlížíme jako na samotné prvky 1 x 1.

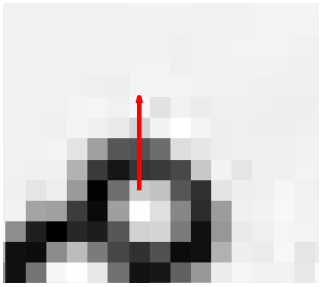
[2]Operace $\circ$ značí prvkové násobení.

4.1.2 Deskriptor nalezených rohů

K tomu, abychom mohli rohy ze vzoru porovnat s rohy v pořízeném snímku, tak musíme kolem každého rohu vyříznout okno. Protože pořízený snímek může být vůči vzoru natočen, tak potřebujeme zjistit orientaci každého takového okna. Tu zjistíme tak, že kolem každého rohu vyřízneme kruhové okno a v něm najdeme gradienty jednotlivých pixelů (obr. 4.2). Tyto gradienty sečteme a dostaneme výsledný vektor, kterým definujeme orientaci rohu. Poté vyřízneme čtvercové okno (obr. 4.3), které je vhodnější na následné porovnávání, protože v diskretizovaném kruhu by mohly vznikat nepřesnosti v podobě chybějících pixelů.



Obrázek 4.2: Kruhové okno s gradienty



Obrázek 4.3: Narovnané čtvercové okno

Nakonec narovnané čtvercové okno normalizujeme pro částečnou invarianci k osvětlení, tedy nová hodnota pixelů bude standardizované z-skóre se střední hodnotou 0 a směrodatnou odchylkou 1:

$$z = \frac{x - \mu}{\sigma}, \tag{4.8}$$

kde μ je aritmetický průměr intenzit pixelů v okně a σ je jejich směrodatná odchylka daná vzorcem

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(I_i - \mu)^2}. \tag{4.9}$$

Zde je N počet pixelů v okně a I_i jsou intenzity pixelů v okně.

4.1.3 Hledání odpovídajících si rohů

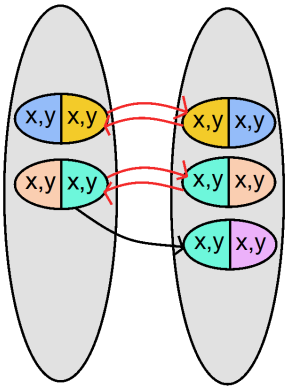
V předchozích krocích jsme vypočítali souřadnice rohů a nalezli jsme jejich deskriptory – normalizovaná orientovaná okna intenzit, v jejichž středu se roh nachází. Máme tedy vše potřebné pro jejich porovnávání.

Postupujeme tak, že procházíme deskriptory z pořízeného snímku a porovnáváme je s deskriptory ze vzoru. Uložíme si dvojice mající nejmenší sumu absolutních rozdílů intenzit. Tedy každému oknu z pořízeného snímku přiřadíme okno ze vzoru: *argmin

$$w^* = \underset{w \in W}{argmin} \sum_{x,y} |w_0(x,y) - w(x,y)|, \tag{4.10}$$

kde w_0 je okno ze snímku, w je právě porovnávané okno ze vzoru, w^* je nejpodobnější okno ze vzoru s w_0 a x,y jsou souřadnice pixelů. Podobně každému oknu ze vzoru přiřadíme okno ze

snímku. I když teď máme dvojice, tak tyto dvojice nemusí odpovídat skutečnosti. Pokusíme se tedy vyfiltrovat rohy, které si neodpovídají. To uděláme pomocí dopředno-zpětného testu znázorněného na obr. 4.4. Máme k dispozici dvě množiny dvojic a koukáme se, zdali dvojice v první množině odpovídá dvojici ze druhé množiny.



Obrázek 4.4: Dopředno-zpětný test na výběr rohů

V následujícím kroku budeme pokračovat v potlačování falešných detekcí (špatně navzájem přiřazených rohů). V testu nejbližších sousedů iterujeme přes všechny rohy a vždy najdeme tři rohy, které mají k danému rohu nejmenší euklidovskou vzdálenost (platí rotační symetrie):

$$m_e = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2} \quad i = 1, \ldots, (N-1)\,. \tag{4.11}$$

Souřadnice vybraného rohu jsou $(x_0,\ y_0)$, souřadnice ostatních rohů (ze kterých vybíráme tři nejbližší) jsou $(x_i,\ y_i)$ a počet rohů je N. Nyní se podíváme, jestli alespoň jeden z nejbližších rohů k rohu ze vzoru odpovídá jednomu z nejbližších rohů korespondujícího rohu ze snímku. Pokud ne, tak tyto rohy dále neuvažujeme.

4.1.4 Úhel natočení

Pro zjištění úhlu natočení snímku se jako první varianta nabízí použít orientaci deskriptorů rohů (viz kapitola 4.1.2). Úhel bychom vypočítali jako rozdíl orientací dvou oken korespondujících rohů, statisticky bychom odstranili úhly, které jsou odlehlé hodnoty. Pak bychom ty zbylé zprůměrovali. Taková metoda sice poslouží dostatečným způsobem k nalezení odpovídajících si rohů (protože jich detekujeme hodně), pro odhadnutí natočení snímku však budeme potřebovat přesnější metodu.

Ta spočívá v tom, že vždy vytvoříme vektor $\boldsymbol{v_1}$ určený dvojicí rohů ve snímku a druhý vektor $\boldsymbol{v_2}$ korespondujících rohů ve vzoru. Poté najdeme úhel, které tyto dva vektory svírají, přičemž musíme zohlednit jeho znaménko. Tímto způsobem najdeme úhly všech možných dvojic rohů:

$$\alpha = sgn\left(det\left(\begin{bmatrix}\boldsymbol{v_1} & \boldsymbol{v_2}\end{bmatrix}\right)\right) \cdot acos\left(\frac{\boldsymbol{v_1}^T\boldsymbol{v_2}}{||\boldsymbol{v_1}||||\boldsymbol{v_2}||}\right). \tag{4.12}$$

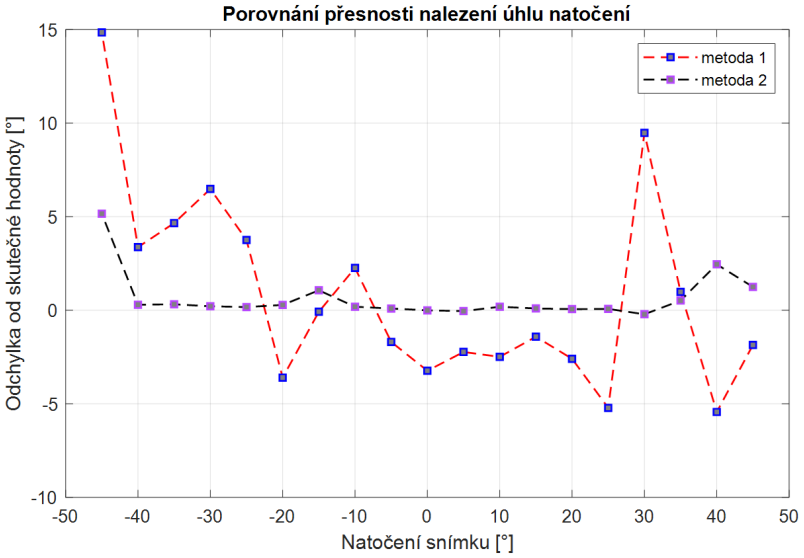
Počet všech těchto kombinací je:

$$\binom{N}{2} = \frac{N \cdot (N-1)}{2}, \tag{4.13}$$

kde N je počet korespondujících rohů.

Z těchto úhlů vybereme ty, které nejsou odlehlé hodnoty pomocí mediánové absolutní odchylky (kapitola 3.3). Vybereme hodnoty, které nejsou větší než tři MAD. Na závěr najdeme výsledný úhel natočení jako aritmetický průměr.

Na grafu níže (obrázek 4.5) porovnáváme dvě zmíněné metody ve smyslu přesnosti nalezených úhlů. Na snímcích byl měřicí přístroj natočený od $-45°$ do $45°$ po pěti stupních. Je vidět, že naše druhá metoda (černo-fialový graf) je mnohem přesnější a pro naši aplikaci uspokojivá. Odchylka od skutečného úhlu natočení je minimální (v řádu minut), začíná se trochu zvětšovat až při velkém natočení přístroje. První metoda (červeno-modrý graf) se sice k reálnému natočení přibližuje, ale k ideálu má daleko.



Obrázek 4.5: Graf zobrazující přesnost obou metod pro nalezení natočení

4.1.5 Nalezení středu ve snímku pro ořez a otočení

Abychom mohli vyříznout ROI ze snímku (display měřicího přístroje), tak potřebujeme znát jeho střed. Ten bude také potřeba pro jeho otočení. Nejdříve vypočítáme souřadnice těžiště $\boldsymbol{t_1}$ ve vzoru a podobně těžiště $\boldsymbol{t_2}$ ve snímku z již vyfiltrovaných rohů pomocí aritmetického průměru:

$$\boldsymbol{t_1} = \frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix} x_{1i} \\ y_{1i} \end{bmatrix}, \qquad \boldsymbol{t_2} = \frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix} x_{2i} \\ y_{2i} \end{bmatrix}. \tag{4.14}$$

Kde (x_{1i}, y_{1i}) resp. (x_{2i}, y_{2i}) jsou souřadnice korespondujících rohů ve vzoru, resp. ve snímku a N je jejich počet (viz rovnice 4.13).

Nyní si vytvoříme vektor $\boldsymbol{u}$ znázorňující posun těžiště od středu $\boldsymbol{s_1}$ ve vzoru:

$$\boldsymbol{u} = \boldsymbol{s_1} - \boldsymbol{t_1}. \tag{4.15}$$

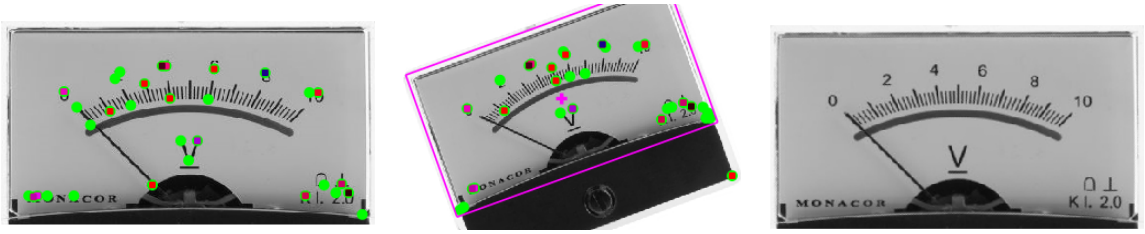
Tento vektor je potřeba otočit o nalezený úhel natočení snímku θ z předchozí kapitoly. Tedy pomocí rotační matice získáváme vektor

$$\boldsymbol{u'} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \boldsymbol{u}. \tag{4.16}$$

Střed ROI ve snímku $\boldsymbol{s_2}$ pak najdeme jako

$$\boldsymbol{s_2} = \boldsymbol{u'} + \boldsymbol{t_2}. \tag{4.17}$$

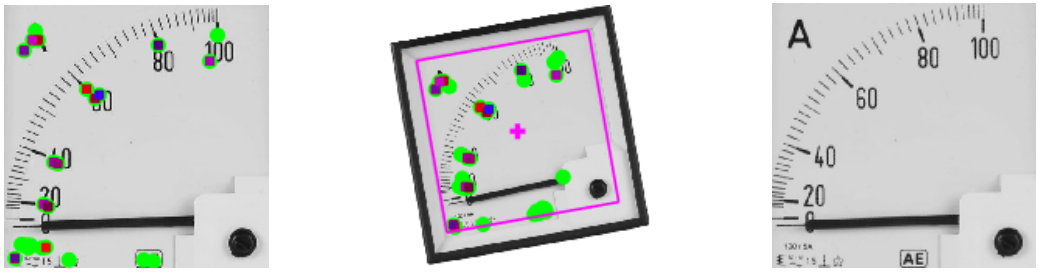
Nyní již máme k dispozici vše potřebné (úhel natočení, střed ROI, rozměry – stejné jako u vzoru) pro vyříznutí displaye ze snímku a k jeho natočení.



(a) : Vzor **(b)** : Obraz **(c)** : Výsledek

Obrázek 4.6: Nalezení vzoru ve snímku a jeho natočení – klasický měřicí přístroj

Na obrázku 4.6 vidíme výsledek kapitoly 4.1. Vlevo se nachází vzor sloužící k hledání ROI ve snímku, který je uprostřed (nalezený ROI je vyznačen purpurovým obdélníkem). Zelenými body jsou znázorněny detekované rohy. Červená barva představuje korespondující rohy, které prošly dopředno-zpětným testem (kap. 4.1.3). Ostatní body (odstíny fialové) jsou konečnou množinou rohů, které prošly testem nejbližších sousedů (kap. 4.1.3) . Z této konečné množiny najdeme úhel, o který je snímek natočený (kap. 4.1.4). Vystřižený a narovnaný ROI (tedy display) se nachází napravo. Na obrázku 15 vidíme tu stejnou situaci, ale na kvadrantovém měřicím přístroji.



(a) : Vzor **(b)** : Obraz **(c)** : Výsledek

Obrázek 4.7: Nalezení vzoru ve snímku a jeho natočení – kvadrantový měřicí přístroj

4.2 Hledání oblouku

Proto, abychom mohli detekovat stupnici, tak potřebujeme najít kružnici (přesněji oblouk), na které se stupnice nachází v drtivé většině případů. Takovou kružnici najdeme pomocí cirkulární Houghovy transformace[3] podrobněji popsané v kapitole 3.4.

Protože kruhová Houghova transformace funguje na základě hlasování v parametrovém prostoru podle počtu bodů (bílých pixelů) ležících na hledané kružnici, potřebujeme, aby těchto bodů bylo co nejvíce. Jelikož jednotlivé čárky na stupnici se vyznačují tím, že jsou úzké a na malém prostoru mění rychle intenzitu, jsou dobrými kandidáty pro přítomnost rohů. Nyní tedy podruhé v této práci použijeme detektor rohů (teorie viz kapitola 3.2, implementace viz kapitola 4.1.1), nyní však už v menším obrázku – v nalezeném ROI. A protože jich potřebujeme co nejvíce, tak nyní počet rohů nebudeme omezovat číselně, ale budeme se dívat pouze na kvalitu jejich rohové odezvy. Potřebujeme tedy mezní hodnotu. Když bude vyšší než 5 % maximální hodnoty z matice hodnot rohové odezvy R, tak se bude jednat o roh. Tedy pro souřadnice rohu x,y platí

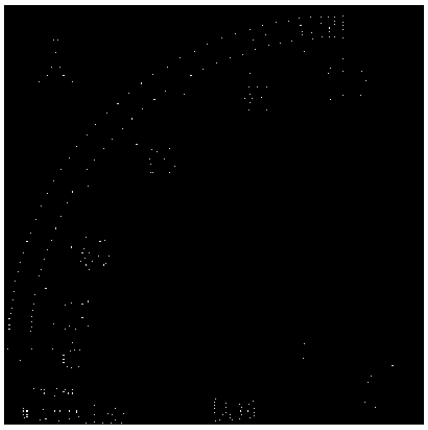
$$R(x, y) > 0.05 \cdot \max(R). \tag{4.18}$$

To by mělo pokaždé zaručovat dostatečný počet rohů. Houghova transformace je funkční nad binárním obrázkem, proto si vytvoříme matici BW o rozměrech ROI a na získané souřadnice uložíme logickou jednotku (obrázek 4.8). Protože nemáme k dispozici poloměr hledané kružnice, tak je potřeba si určit interval, ve kterém se bude hledaný poloměr nacházet. Jeho dolní mez zvolíme jako polovinu výšky obrázku. Poloměr kružnice, na které se nachází stupnice, by nikdy neměl být menší, protože stupnice se vždy pro přehlednost a pro dostatečné rozpětí nachází v horní polovině měřicího přístroje (případně nad uhlopříčkou z levého dolního do pravého horního rohu u kvadrantového čtvercového přístroje). Zároveň tato kružnice nikdy nemá menší poloměr, než je délka ručičky, která vychází téměř z hrany displaye (u kvadrantového přístroje z pravého dolního rohu). Horní mez je nastavena na výšku a půl obrázku. Střed kružnice se totiž může nacházet mimo display (více v kapitole 2.1).

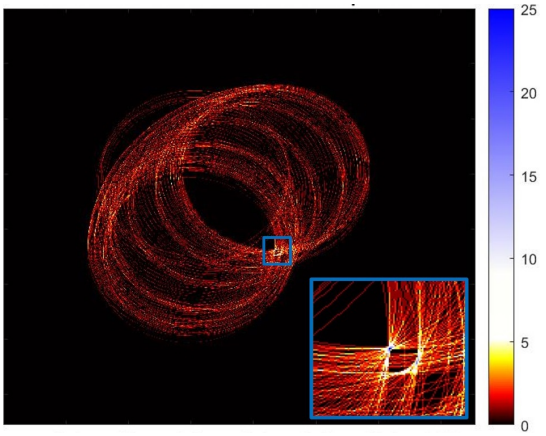
Algoritmus postupuje následovně. Nejdříve založí akumulátor (3D matice), jehož rozměry jsou určené intervalem poloměrů a rozměry obrázku. Protože se střed hledané kružnice může nacházet i mimo ROI, tak musíme adekvátně akumulátor zvětšit, jeho výška (resp. šířka) bude třikrát větší než výška (resp. šířka) ROI. Poté se podívá na matici BW. Pokud se pracuje s klasickým měřicím přístrojem, tak pro ušetření výpočetní náročnosti se nastaví nula na pozice jedniček, které jsou v dolní polovině. Pokud se jedná o kvadrantový přístroj, tak se nastaví nuly v pravém dolním rohu. V těchto místech totiž stupnice rozhodně není. Poté se iteruje přes interval poloměrů a na základě metody popsané v kapitole 3.4 se inkrementuje akumulátor s příslušným poloměrem. Následně se vybere kružnice s nejvyšší hodnotou v akumulátoru.

Na obrázku 4.9 pozorujeme jednu rovinu z Houghova prostoru. Tedy jedná se o vrstvu akumulátoru představující námi hledanou kružnici s jejím poloměrem – její střed (v obrázku přiblíženo, s modrou barvou) dostal v hlasování nejvíce bodů.

[3] Jádro kódu je převzato od [17], pro naše účely byl program pozměněn.



Obrázek 4.8: Binární obrázek

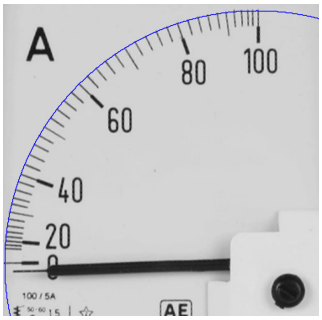


Obrázek 4.9: Vrstva akumulátoru

4.3 Detekce prstencového regionu se stupnicí

Nyní, když jsme našli oblouk, na kterém je stupnice položena, tak můžeme detekovat prstencovou oblast, kde se stupnice nachází. Navržená metoda funguje na základě porovnávání počtu bílých a tmavých pixelů. Proto je nejdříve potřeba obrázek binarizovat, to provedeme pomocí Otsuovy metody podrobněji popsané v kapitole 3.5.

(a) : Oblouk je na vnitřní straně stupnice



(b) : Oblouk je na vnější straně stupnice

Obrázek 4.10: Porovnání nalezených oblouků.

Vstupem do algoritmu jsou parametry kružnice (a_0, b_0, R) získané v předchozí kapitole. Nejdříve je potřeba ověřit, na které straně kružnice se oblouk nachází. Buď se nachází na vnější straně stupnice, nebo na vnitřní, jak je znázorněno na obrázku 4.10. Všimněme si, že kružnice se nachází na té straně, na které nejsou čísla – tedy všechny stupně vycházejí z oblouku a nepřesahují ho. Díky tomu bylo na tomto oblouku nalezeno nejvíce rohů. Vytvoříme si dvě velmi tenká mezikruží M_1a M_2. M_1 se skládá z kružnic o poloměrech R a $R-5$. Naproti tomu M_2 je složeno z kružnic o poloměrech R a $R+5$. Nyní porovnáme počty černých pixelů k bílým každého mezikruží. Podle toho, ve kterém z mezikruží bude větší zastoupení černých pixelů, tak určíme, na jaké straně kružnice (a_0, b_0, R) najdeme stupnici a jakým směrem d se budeme v dalším kroku pohybovat.

Algoritmus nyní postupuje následujícím způsobem. Kromě kružnice s poloměrem R mějme index i znázorňující iteraci algoritmu, konstantu s představující délku kroku, s jakou se bude v každé iteraci zvětšovat poloměr a směr d nabývající hodnot $\{-1, 1\}$. Každý kruh bude mít střed v (a_0, b_0). Na začátku vyřízne kruh k_1 o poloměru R. Poté vyřízne kruh k_2o poloměru $R + d \cdot i \cdot s$. Tyto dva kruhy od sebe odečte a získá mezikruží. Nyní vypočítá poměr k_i počtu světlých N_W a tmavých pixelů N_B:

$$k_i = \frac{N_B}{N_W}. \tag{4.19}$$

Poté se vypočítá poměr k_{i+1} pro mezikruží s poloměry R a $R + d \cdot (i + 1) \cdot s$. Pokud je tento poměr mnohem menší než průměrný poměr předchozích mezikruží k_m (s indexy $1 \dots i$)

$$k_m = \frac{\sum_{j=1}^{i} k_j}{i}, \tag{4.20}$$

tak je konec a je nalezeno hledané mezikruží. Pokud se tento poměr moc neliší, tak se iteruje dále.

Na závěr k poloměru kruhu k_2 ještě přičte nebo odečte (v závislosti na směru d) konstantu, která mezikruží ještě malinko rozšíří: na stupnici jsou v menšině zastoupeny větší stupně, ke kterým je většinou na měřicím přístroji napsaná jejich číselná hodnota. Tyto větší stupně budeme detekovat k pozdější kapitole.

Je třeba poznamenat, že ve skutečnosti porovnáváme poměry k_i a k_m až od určitého indexu dál (tedy i není na začátku nastaveno na jedničku), protože se může stát, že nalezená kružnice (a_0, b_0, R) nepřiléhá dokonale ke stupnici (to je dáno nedokonalostí vytištěné stupnice).

(a) : Odstíny šedi **(b) :** Binarizováno

Obrázek 4.11: Detekované mezikruží se stupnicí.

4.4 Detekce pointeru

4.4.1 Zjednodušení snímku

Ještě předtím, než přiřadíme hodnotu jednotlivým značkám na stupnici, tak je výhodné najít ručičku. Tu v další kapitole použijeme pro vymezení výseče mezikruží, abychom kromě stupnice nedetekovali jiné objekty.

Přestože už máme ROI displaye, na kterém se ručička nachází, tak to je pořád až moc velká plocha. Na té stále najdeme mnoho struktur (jako popisky nebo jinou grafiku), které

mají podobně jako pointer tvar přímky. To vede k eventuálním falešným detekcím. Proto využijeme kružnice z předchozích kapitol a morfologických operací (blíže popsaných v kapitole 3.6) ke zmenšení regionu a odstranění nežádoucích objektů.

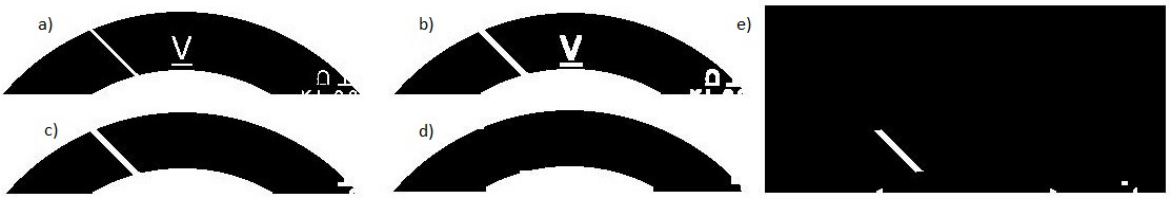
Pokud byla pomocí KHT rozpoznána kružnice na vnější straně stupnice, tak je její poloměr náležitě upraven, aby se vždy nacházela na té vnitřní. Také je myšleno na případné zrcátko, které může být na displayi měřicího přístroje přítomné – poloměr kružnice by byl zmenšen o zvolenou konstantu. Poté vytvoříme mezikruží, které se nyní bude nacházet pod stupnicí (blíže ke středu). V něm se nachází pointer – jeho část. Šířka mezikruží je totiž zvolena tak, aby ho ukazatel přesahoval z obou stran. Mezikruží převedeme na černobílý obrázek pomocí Otsuovy metody.

Nyní je myšlenka následující. Najdeme největší souvislou oblast (8-konektivita). Tou bude oblast mimo mezikruží z obou stran (včetně ukazatele), neboť tato oblast je spojena ukazatelem. Přesněji můžeme říct, že ukazatel vytváří most mezi komponentami oddělenými mezikružím. Důležité však je, aby byl pointer spojitý (také ve smyslu 8-konektivity). Může se totiž stát, že bude příliš tenký a při binarizaci vzniknout díry. Proto ještě předtím použijeme binární dilataci, která se vyznačuje vyplňováním děr a zmohutněním objektů. Jako strukturní element použijeme

$$SE = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{4.21}$$

Ten zaplní mezery o tloušťce dvou pixelů a zároveň z každé strany objekt zvětší o jeden pixel.

Teď tedy najdeme největší komponentu. Tím se efektivně zbavíme malých objektů uvnitř mezikruží. Hledání komponent je založeno na algoritmu dvojího průchodu (anglicky two-pass connected component labeling) s ekvivalenční tabulkou [18]. Nakonec použijeme morfologické otevření. To způsobí oddělení větších komponent (tj. oblast nad a pod mezikružím) spojených úzkým mostem. My však potřebujeme pouze most – ručičku, a proto odečteme otevřený obrázek od neotevřeného. Všechny operace jsou znázorněny na obrázku 24.



Obrázek 4.12: a) binarizované mezikruží; b) dilatované; c) největší region; d) otevření; e) odečtení (Jednotlivá mezikruží jsou vždy obdélníkové obrázky)

4.4.2 Hledání přímky

Naším cílem je proložit ručičku přímkou, protože chceme zjistit náklon ručičky. Také ji budeme potřebovat pro nalezení průsečíku, ve kterém se pomyslně zbíhají všechny stupně na stupnici. Pro hledání přímky byl zvolen RanSaC (Random Sample Consensus) [19],[20], který je velmi

rychlý a je jednoduchý na implementaci. Jeho velkou výhodou je odolnost proti odlehlým hodnotám. Postupujeme následujícím způsobem:

a) Z množiny dat, kterým je výsledek předchozí podkapitoly, vybereme náhodně dva body.
b) Vytvoříme si model přímky (reprezentované normálovým vektorem).
c) Vypočítáme vzdálenost všech datových bodů od přímky (rovnice 4.22).
d) Zjistíme počet bodů, které leží v těsné blízkosti přímky (dopředu je zvolena mezní hodnota). Pokud ano, tak je model přímky spolu s blízkými body uložen.
e) Opakuj kroky a) – d) po N iterací.

Vzdálenost bodu $\boldsymbol{z}$ od přímky $\{\boldsymbol{x} \in \mathbb{R}^2 \mid \boldsymbol{u}^{\mathbf{T}}\boldsymbol{x} = \boldsymbol{u}^{\mathbf{T}}\boldsymbol{x_0}\}$, kde $\boldsymbol{u}$ je její normálový normalizovaný vektor spočítáme jako

$$dist\,(\boldsymbol{z}) = \left|\boldsymbol{u}^{\mathbf{T}}\,(\boldsymbol{z} - \boldsymbol{x_0})\right|. \tag{4.22}$$

Počet iterací N je napevno zvolen v programu. Jeho vhodnou hodnotu jsme odhadli takto. Pravděpodobnost, že vybereme dva body ležící v sousedství přímky, je zhruba w^2, kde w označuje poměr blízkých bodů ke všem datům. Pak pravděpodobnost, že se nevyberou správné body Nkrát je $(1-w^2)^N$. Tedy $P = 1-(1-w^2)^N$ značí pravděpodobnost, se kterou po N pokusech najdeme alespoň jednou správnou dvojici bodů. Za použití logaritmů vyjádříme N:

$$N = \frac{\log\,(1-P)}{\log\,(1-w^2)}. \tag{4.23}$$

Chceme-li, abychom po N iteracích našli téměř se 100% pravděpodobností správné body, tak si nastavíme $P = 0.99$. S konstantou w již musíme být opatrnější. Např. pro výsledný obrázek z předchozí podkapitoly je to asi 58 %. Musíme však počítat s tím, že se v obrázku kromě ručičky může objevovat i více odlehlých hodnot (např. pokud bude ručička nad nějakým grafickým prvkem displaye). Proto zvolíme w jako 10 %. Po dosazení pak dostaneme $N = 458$.

Po použití metody RanSaC je ještě vhodné pozici přímky drobně upravit. Může se totiž stát, že je přímka mírně nakloněná, neboť binární komponenta pointeru může být širší. Korekci provedeme pomocí lineární regrese ve smyslu nejmenších čtverců [21], tedy minimalizujeme součet čtverců reziduí:

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^2}{min}\sum_{i=1}^{m}\left(y_i - f\,(x_i, \boldsymbol{\theta})\right)^2 = \underset{\boldsymbol{\theta}\in\mathbb{R}^2}{min}||\boldsymbol{y} - A\boldsymbol{\theta}||^2, \tag{4.24}$$

kde (x_i, y_i) jsou souřadnice pixelů blízkých k přímce, m je jejich počet, $\boldsymbol{\theta}$ jsou hledané parametry přímky, $\boldsymbol{y}$ je vektor druhých složek souřadnic pixelů. Matice $A^{m\times 2}$ vypadá takto:

$$A = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix} \tag{4.25}$$

V Matlabu pak vektor parametrů přímky získáme jako $\boldsymbol{\theta} = A\backslash\boldsymbol{y}$.

4.5 Přiřazení hodnot ke značkám na stupnici

V případě, že se jedná o klasický typ měřicího přístroje, tak se pomyslný průsečík P značek na stupnici nemusí shodovat se středem kružnice nalezeným pomocí KHT (viz kapitola 4.2). Přesněji řečeno, totožná bude x-ová souřadnice, mezi ypsilonovými souřadnicemi bude jistý posun. A protože již máme parametry přímky procházející ručičkou, která je vždy stejně nakloněná jako značka, na kterou ukazuje, tak můžeme vypočítat P dosazením již známé x-ové souřadnice a_0 do rovnice přímky:

$$y = \theta_1 + \theta_2 a_0. \tag{4.26}$$

Pak tedy dostaneme $P = (a_0, y)$.

Dále si z jedné strany vymezíme výseč mezikruží pomocí nalezené přímky. Pořád se totiž nacházíme v pasivní fázi algoritmu (viz obrázek 1), kdy se ručička nachází v klidové nulové pozici – tedy najdeme ji na kraji stupnice. Úhel směrového vektoru $\boldsymbol{u'} = (u'_1, u'_2)$ přímky s osou x je

$$\varphi = atan2\left(-u'_2, u'_1\right). \tag{4.27}$$

Všimněme si záporného znaménka u prvního argumentu. Je to kvůli tomu, že souřadnicový systém obrázku má otočenou osu y. Z druhé strany je výseč omezena podle typu měřicího přístroje: u klasického je to 0° a u kvadrantového je to 45° (chceme, aby tam byla určitá rezerva).

Nyní si upravíme souřadnicový systém, jehož počátek posuneme do průsečíku P. Využijeme detekovaného mezikruží se stupnicí z kapitoly 4.3. Každý pixel, jehož hodnota je 1, spojíme s počátkem. Podobně jako výše, vytvoříme směrový vektor této spojnice a vypočítáme úhel s osou x. Z těchto úhlů si vytvoříme histogram, který nám popisuje četnosti jednotlivých úhlů. Naším cílem je najít ve stupnici velké značky, ke kterým je většinou přidružené číslo na displayi. Počet sloupců je zvolen následující úvahou. Většina měřicích přístrojů má úhlové rozpětí stupnice 90°. Dejme tomu, že v nejhorším případě by značky od sebe byly vzdáleny $0,5°$ (menší rozestup mít určitě nebudou, protože by byla stupnice velice špatně čitelná), pak by se nacházelo 90 značek a 90 mezer na zmíněném rozpětí stupnice. Tedy bychom potřebovali alespoň 180 vzorků.

Nejdříve najdeme místa (sloupce histogramu) bez značek jako lokální minima. Tím se oddělí od sebe jednotlivé sloupce (nebo skupiny sloupců) odpovídajících značek. Pokud se jedná o skupinu sloupců, tak náležitě upravíme úhel korespondující značky jako vážený průměr všech úhlů φ_i (sloupců ze skupiny). Váhy n_i jsou v tomto případě četnosti jednotlivých úhlů. Tedy nový úhel je

$$\bar{\varphi} = \frac{n_1\varphi_1 + \cdots + n_k\varphi_k}{n_1 + \cdots + n_k}, \tag{4.28}$$

kde k značí počet sloupců ve skupině. Zároveň každé značce přidělíme celkovou četnost (tedy suma četností jednotlivých sloupců ze skupiny). Ta odpovídá počtu pixelů ve značce.

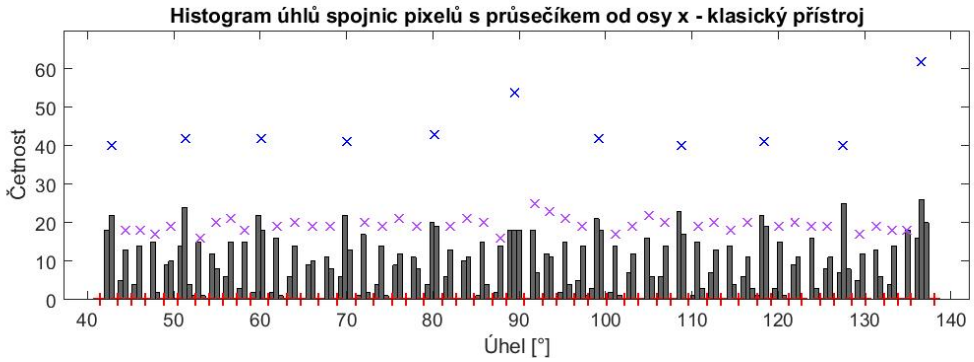
Protože chceme chceme najít velké značky, tak se na ně můžeme dívat jako na odlehlé hodnoty s větším množstvím pixelů (tj. větší četností). Proto si zvolíme značku s největším počtem pixelů N_{max}. Značky, které mají více pixelů než $\frac{2}{3}N_{max}$ jsou považovány za odlehlé

hodnoty. Tím jsme detekovali velké značky.Pro doplnění uvedu, že ve skutečnosti jako N_{max} bereme druhou největší značku. Ta první je totiž překryta ukazatelem a ten může její plochu velmi zvětšit.

Nakonec přiřadíme detekovaným značkám hodnotu. Dopředu musí být zadaná maximální hodnota M, kterou na displayi měřicího přístroje najdeme. Poté je značce s největším identifikovaným úhlem od osy x přidělena 0. Ostatním značkám se jejich hodnota v vypočítá takto:

$$v_i = i\frac{M}{N-1}, \qquad i = 1, \ldots, \ N-1, \tag{4.29}$$

kde N je počet nalezených velkých značek. Pokud jsou čísla na stupnici např. logaritmicky nebo jinak uspořádaná, tak je možnost zadat všechny hodnoty (velkých značek).



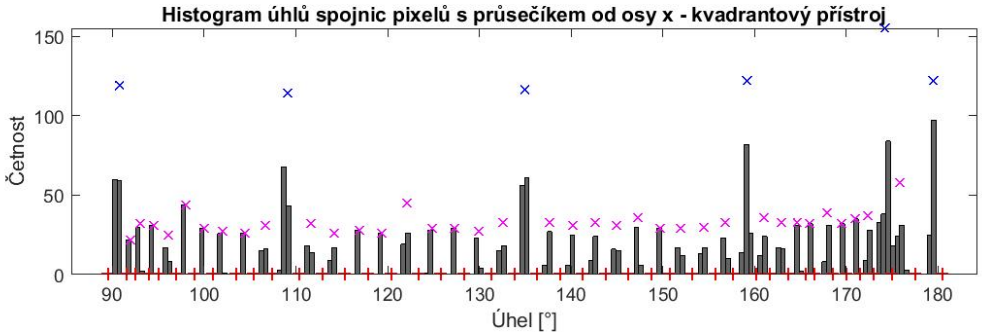
Obrázek 4.13: Histogram (klasický přístroj)

Na obrázku 4.13 je znázorněn histogram, se kterým jsme pracovali. Křížky ‚x' značí přítomnost jakékoliv značky, čím výše jsou položené, tím více mají pixelů. Modré křížky jsou odlehlé hodnoty (tedy velké značky), které hledáme. Červené křížky ve tvaru ‚+' jsou lokální minima, která oddělují jednotlivé skupiny sloupců (tedy jednotlivé značky). Přesná čísla úhlů velkých značek a jejich přidělené hodnoty jsou uvedeny v tabulce 4.1. Histogram i tabulka odpovídají měřicímu přístroji na obrázku 4.6.

Úhel [°]	42.7	51.3	60.1	70.0	80.1	89.5	99.2	108.7	118.2	127.5	136.5
Hodnota	10	9	8	7	6	5	4	3	2	1	0

Tabulka 4.1: Nalezené úhly velkých značek a k nim přiřazené hodnoty (klasický přístroj)

Na obrázku 26 vidíme podobnou situaci, jen s tím rozdílem, že se jedná o kvadrantový přístroj (obrázek 4.7). Také je vidět, že jsou na něm velké značky rozestaveny nerovnoměrně. Přiřazení úhlů a hodnot je uvedeno v tabulce 2.



Obrázek 4.14: Histogram (kvadrantový přístroj)

Úhel [°]	90.7	109.0	135.0	159.2	174.2	179.4
Hodnota	100	80	60	40	20	0

Tabulka 4.2: Nalezené úhly velkých značek a k nim přiřazené hodnoty (kvadrantový přístroj)

4.6 Aktivní část

Nyní již máme vše potřebné a přesouváme se do smyčkové části algoritmu (viz obrázek 2.3). V této části se již může pohybovat ručička. Využíváme údajů získaných z pasivní části. Tedy pro vystřižení displaye z pořízeného snímku kamery použijeme údaj o středu displaye (ROI) a jeho natočení získaného v kapitole 4.1.4. Také využijeme detekované kružnice, na které je položena stupnice (tu již nyní nebudeme znovu detekovat), pro vytvoření mezikruží, ve kterém najdeme ručičku.

Po nalezení ručičky uděláme ještě malý test, který kontroluje, zdali jsme pomocí metody RanSaC opravdu našli ukazatel, nebo jiný objekt. Podíváme se, jestli nově vypočítané souřadnice průsečíku odpovídají tomu z pasivní části – tedy jestli jsou od sebe vzdáleny méně, než je stanovená mez (ta je nastavena na 30 % výšky vystřiženého ROI).

Nyní se znalostí natočení ručičky a jednotlivých velkých značek, kterým byly přiřazeny hodnoty, můžeme vypočítat indikovanou hodnotu. Tu zjistíme pomocí lineární interpolace. Známe-li úhel směrového vektoru ručičky φ, úhly nejbližších dvou značek α_0 a α_1 a jejich hodnoty T_0 a T_1 a platí, že $\varphi \in \langle \alpha_0, \alpha_1 \rangle$, pak pro indikovanou hodnotu platí:

$$V(\varphi) = T_0 + \frac{T_1 - T_0}{\alpha_1 - \alpha_0} (\varphi - \alpha_0) . \tag{4.30}$$

Zároveň stanovíme, že pokud bude ručička přesahovat poslední značku, tak jí nastavíme hodnotu poslední značky. A naopak pokud bude ručička před první značkou, tak jí nastavíme hodnotu první značky.

Kapitola 5

Experimentální ověření na embedded platformě

5.1 Zapojení

Jako vestavěný systém jsme si zvolili jednodeskový mikropočítač Raspberry Pi model 3B+. Jeho součástí je 64-bit čtyřjádrové CPU ARM Cortex-A53 taktovaný na frekvenci 1.4 GHz. Je vybaven 1 GB operační pamětí. Zároveň obsahuje moduly Wi-Fi a Bluetooth a koknektory USB a CSI. Jako operační systém byl zvolen Raspbian. Byl stažen jeho obraz a nainstalován na paměťovou kartu.

Raspberry Pi byl zvolen z toho důvodu, protože umožňuje uložení obrázku do paměti (a jeho čtení), dále Cortexový procesor zaručuje vysokou rychlost, která je v našem projektu důležitá a konečně umožňuje připojení různých webkamer pomocí USB, případně kamerových modulů. Jako výhodu lze vnímat i připojení výstupního displaye, na kterém lze snadno zobrazovat zpracovávaný obraz.

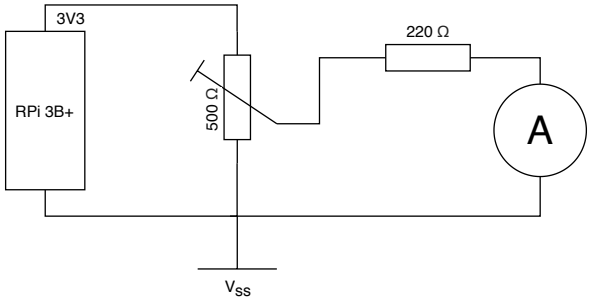
Pro naši práci byl zvolen oficiální kamerový modul, který dokáže pořizovat foto v rozlišení až 3280 x 2464 px. Zároveň, a to je pro náš projekt důležité, natáčí video 1080p při 30 fps nebo 720p při 60 fps. Je umístěn do speciálního plastového krytu, který kromě ochrany nabízí uchycení na ministativu. Pomocí plochého kabelu je připojen k RPi.

Jako analogový měřič byl zvolen ampérmetr měřící protékající proud. Zvolený model má tu výhodu, že nemá zabudovaný vnitřní rezistor (ve skutečnosti má jen malý vnitřní odpor). Maximální proud, při kterém bude ručička ukazovat na poslední dílek, je 15 mA – to odpovídá dílku s hodnotou 25 na stupnici. Protože budeme napájet z pinu RPi o napětí $3V3$, tak potřebujeme ochranný rezistor R_A o velikosti 220 Ω, který bude suplovat chybějící vnitřní rezistor. Za normálních okolností bychom nyní použili paralelně zapojený bočník R_B pro změnu rozsahu ampérmetru:

$$R_B = R_A \frac{1}{(n-1)}, \tag{5.1}$$

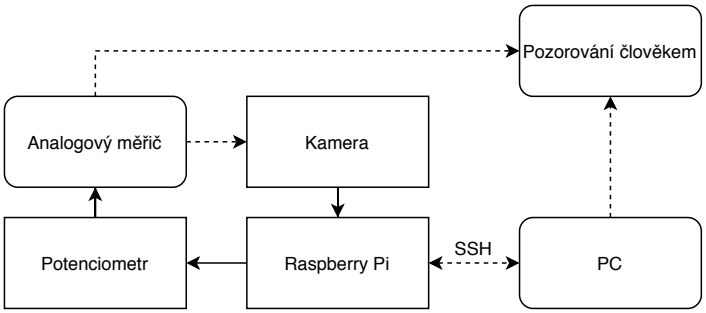
kde n říká, kolikrát se zvětší rozsah ampérmetru. Tedy v našem případě by bylo $n = 25/15$. Pak by $R_B = 330\ \Omega$. Naším cílem ovšem není, aby se proud procházející obvodem shodoval s čísly na stupnici. Pro ověření funkčnosti potřebujeme pouze hýbat s ručičkou. Proto místo

bočníku použijeme trimr – tedy nastavitelný rezistor fungující na principu napěťového děliče. Pro pohodlné ovládání byl vybrán trimr o velikost 500 Ω. Kdybychom zvolili větší, tak malé otočení by znamenalo velkou změnu pozice ukazatele. Zapojení je znázorněno na obrázku 5.1.

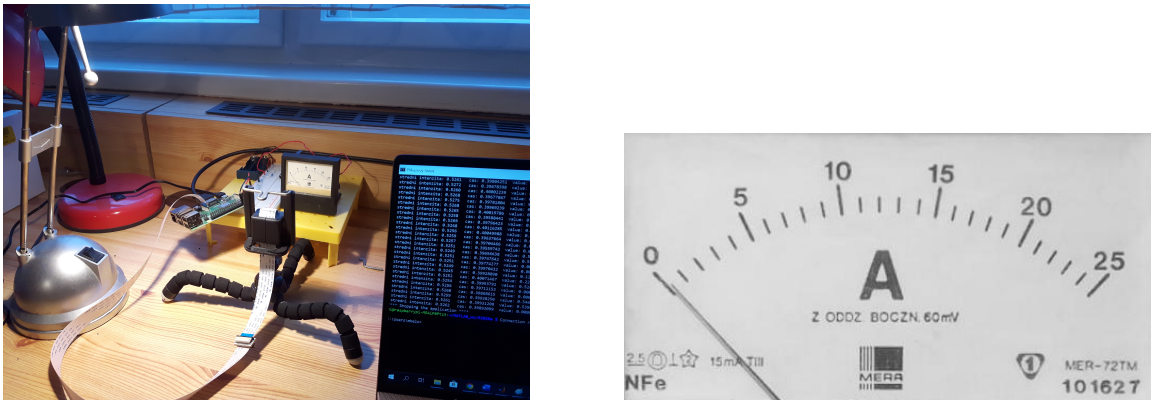


Obrázek 5.1: Elektrické zapojení ampérmetru a Raspberry Pi

Na Raspberry Pi jsme napojení z jiného počítače pomocí Secure Shellu. Díky tomu lze např. pozorovat hodnoty na měřiči z jiné místnosti. Podmínkou je připojení na stejné síti. Na obrázku 5.2 je schéma představující náš experiment. Na obrázku 5.3 je vyfocená scéna se zapojením. Je na něm vidět i přítomnost světelného zdroje, kterým budeme měnit osvětlení měřicího přístroje.



Obrázek 5.2: Experiment – schéma



(a) : Zapojení **(b) :** Display

Obrázek 5.3: Experiment – foto

5.2 Měření

Ověření přesnosti je v případě analogových měřicích přístrojů složitější z toho důvodu, protože čtení z nich je částečně subjektivní a je ovlivněno mnoha vnějšími faktory. Proto skutečná hodnota neexistuje. Mohli bychom teoreticky měřit pomocí digitálního multimetru, jenže zde bychom neověřovali výchylku ukazatele na displayi, ale proud samotný. A to není předmětem této práce. A navíc bychom tím zanedbávali chybu způsobenou nedokonalostí vytištěné stupnice nebo mechanickými vlastnostmi přístroje. My budeme za skutečnou hodnotu V_s považovat přečtení od jedince se zdravým zrakem dívajícího se pokud možno pod stejným úhlem (tedy za normálních okolností kolmo) k displayi jako kamera.

V tabulkách 5.1, 5.2, 5.3 a 5.4 vidíme vypočítané (naměřené) hodnoty od nejvyšší intenzity osvětlení po nejnižší. Nahoře jsou zobrazeny vybrané skutečné hodnoty V_s a v jejich sloupečcích jsou naměřené hodnoty, které by se jim měly podobat. Číslo n udává číslo měření.

V_s / n	**5**	**7.5**	**10**	**12.5**	**15**	**17.5**	**20**
1.	4.9933	7.4205	9.8126	12.3887	15.2681	17.5745	20.3208
2.	5.0372	7.4122	9.8153	12.4574	15.2669	17.5643	20.2508
3.	5.0874	6.9825	10.0832	12.4584	15.1327	18.1744	20.2735
4.	4.9684	7.2257	10.0826	12.4600	15.1974	17.5643	20.3557
5.	5.1006	7.4372	9.8327	12.4599	15.3491	17.5643	20.2543
Průměr:	5.0374	7.2956	9.9253	12.4449	15.2428	17.6884	20.2910

Tabulka 5.1: Naměřené hodnoty: Střední intenzita = 0.6

V_s / n	**5**	**7.5**	**10**	**12.5**	**15**	**17.5**	**20**
1.	5.0287	7.6922	10.2671	12.4687	15.4149	17.5760	20.4773
2.	4.9933	7.7301	10.1509	12.4679	15.2359	17.5781	20.3834
3.	5.0369	7.6113	10.2713	12.4691	15.4870	17.5818	20.3461
4.	5.1077	7.4828	10.0101	12.4691	15.4867	17.7805	20.3322
5.	5.0881	7.7090	10.1336	12.4691	15.4225	17.5818	20.4922
Průměr:	5.0509	7.6451	10.1666	12.4688	15.4094	17.6196	20.4062

Tabulka 5.2: Naměřené hodnoty: Střední intenzita = 0.4

V_s / n	**5**	**7.5**	**10**	**12.5**	**15**	**17.5**	**20**
1.	5.1344	7.5084	10.0791	-	15.4910	17.7546	20.4928
2.	5.0697	7.5529	10.2752	12.4707	15.5087	17.9526	-
3.	4.9647	7.8252	10.2753	-	15.5153	17.8522	20.5210
4.	5.0872	7.7741	10.1206	12.5324	15.5071	17.9457	-
5.	5.0508	7.5324	10.0807	-	15.5003	-	20.3924
Průměr:	5.0614	7.6386	10.1662	(12.50)	15.5045	(17.88)	(20.47)

Tabulka 5.3: Naměřené hodnoty: Střední intenzita = 0.17

n \ V_s	5	7.5	10	12.5	15	17.5	20
1.	-	-	9.8036	-	-	-	-
2.	-	-	10.4351	-	15.4375	17.5996	-
3.	5.6546	-	10.0117	-	-	-	22.1237
4.	4.3795	7.4923	10.0035	-	-	-	-
5.	4.6979	6.9021	-	-	-	-	21.2970
Průměr:	(4.91)	(7.20)	(10.06)	-	(15.44)	(17.60)	(21.71)

Tabulka 5.4: Naměřené hodnoty: Střední intenzita = 0.02

Tato měření představují situaci, kdy bylo ROI displaye správně detekováno a natočeno. V prvních dvou tabulkách vidíme, že ve všech případech byl ukazatel nalezen a hodnota byla vypočítaná. Lze si všimnout, že např. pro skutečnou hodnotu $V_s = 5$ je měření velice přesné i precizní. U $V_s = 10$ je vidět, že jsme sice přesní, ale už ne tolik precizní, protože se hodnoty dost mění. Koukneme-li se na $V_s = 15$ nebo $V_s = 20$, tak pozorujeme určitou systematickou chybu, která je nejspíše dána lidským faktorem při přečtení skutečné hodnoty. Také se mohlo stát, že kamera místo ručičky přečetla její stín. To se může zhoršovat při horším osvětlení – snížení intenzity světla bylo docíleno překrýváním světelného zdroje, čímž dojde k rozptýlení světla a to může mít vliv na stín.

V tabulkách 5.3 a 5.4 vidíme[1], že se občas ručička nedetekovala – osvětlení bylo už moc nízké. Přesto, pokud k nalezení došlo, tak jsou hodnoty relativně přesné (až na výjimky).

Nyní jsou představeny různé scénáře, jak by mohl být měřicí přístroj posunut nebo natočen (nebo naopak kamera). Podíváme se na první fázi programu – na detekci ROI displaye. Tato část programu je citlivá na změny, a to hlavně na snížení osvětlení. Proto je detekce u všech scénářů zkoumána za normálních[2] a zhoršených[3] světelných podmínek.

Transpozice	Dobré o.	Špatné o.
0	100%	50%
-2°	80%	40%
-10°	60%	20%
-15°	70%	20%
+1 cm →	90%	50%
+2 cm →	90%	60%

(a) : První část

Transpozice	Dobré o.	Špatné o.
-10°, +2 cm →	80%	30%
1 cm ↑	100%	80%
0.5 cm dál	90%	60%
1 cm dál	40%	20%
0.5 cm blíž	90%	60%
1 cm blíž	60%	20%

(b) : Druhá část

Tabulka 5.5: Úspěšnost detekce ROI při změně pozice měřiče vůči kameře při dobrém a horším osvětlení

Jak vidíme v tabulce 5.5[4], tak za dobrých světelných podmínek je detekce displaye poměrně přesná a v nejhorším případě (při transpozici v jedné rovině), tedy při větším natočení, byla

[1] Průměrné hodnoty jsou v závorkách, protože průměr nebere zřetel na nedetekované případy
[2] Střední intenzita pořízeného snímku je 0.3 (střední intenzita ROI je 0.5)
[3] Střední intenzita pořízeného snímku je 0.18 (střední intenzita ROI je 0.25)
[4] V každé situaci bylo provedeno 10 měření

úspěšnost 60%. Je tedy vidět, že rotace má určitý vliv na úspěšnost detekce. Lze si všimnout, že při posouvání v rovině (doprava nebo nahoru) není detekce skoro vůbec ovlivněná a za dobrých světelných podmínek se pohybuje kolem 100 %. V prvním řádku 5.5b je situace, kdy je měřicí přístroj posunut i natočen. 80 % lze považovat za dobrý výsledek. Za horších světelných podmínek lze pozorovat, že úspěšnost detekce klesá. Nakonec bylo také zkoumáno, jaký vliv má posouvání přístroje blíž nebo dál od kamery. Jak je vidět, tak při posunutí o 0.5 cm dostáváme dobré výsledky. Jakmile je přiblížení či oddálení větší, tak se úspěšnost detekce ROI zmenšuje. To může být dáno horší detekcí rohů (přiblížený deskriptor rohu se bude drobně lišit od deskriptoru odpovídajícího rohu ve vzoru).

5.3 Časová a paměťová náročnost

Zpracovávání obrazu může být časově i paměťově poměrně náročné, protože se pracuje s velkým množstvím dat. Čím větší je rozlišení snímku, tím více paměti potřebujeme. Proto byl zvolen kompromis mezi velkými daty a dostatečným rozlišením a bylo vybráno rozlišení 1280x720 px. Takové rozlišení je navíc podporováno u většiny webových kamer. Zároveň je třeba dbát na to, že operační paměť mikropočítače Raspberry Pi je 1 GB. Ve skutečnosti máme k dispozici 875 MB, protože 125 MB je sdíleno s GPU). Z této hodnoty musíme ještě odečíst zhruba 170 MB, které využívá operační systém Raspbian. Pro náš program tedy můžeme využít cca 700 MB.

	Hledání natočení a detekce ROI	Nalezení oblouku	Mezikruží se stupnicí	Detekce ukazatele	Detekce značek a přidělení hodnot	Iterace aktivní části
Čas [s]	11.1873	2.4822	0.5936	0.3212	0.0073	0.3829
Využívaná pamět [MB]	147	374	151	152	151	146

Tabulka 5.6: Nároky na čas a paměť podle jednotlivých fází programu

V tabulce 5.6 je znázorněna doba trvání jednotlivých částí algoritmu a zároveň množství paměti, které je zrovna používáno. Nejdelší je samotná detekce displaye. Většinu tohoto času tvoří detekce rohů v pořízeném snímku. Nalezení oblouku pomocí kruhové Houghovy transformace má potenciál být extrémně časově náročná. V našem případě jsme se snažili omezit počet pixelů tím, že uvažujeme jen rohy. A ty pak vybíráme jen z určité části ROI (podrobněji viz 4.2). Díky tomu jsme tento čas stáhli na 2.5 s. Ostatní fáze programu jsou v řádu desetin vteřiny. Důležitá je doba jednoho přečtení z displaye (poslední sloupeček). Díky tomu se dostáváme na necelé 3 fps. Díky tomu je možné zaznamenat pohyb ručičky, pokud se tedy hodnoty mění dostatečně pomalu. Je třeba si uvědomit, že část s nejdelším trváním se děje pouze při spuštění programu. Co se využívané paměti týče, tak nejnáročnější je zmíněná transformace. Jako datový typ akumulátoru jsme zvolili 8bitové bezznaménkové číslo (uint8). Akumulátor by se přes hodnotu 255 neměl nikdy dostat, to by muselo být na displayi cca 255 dílků (rohy se detekují na koncích dílku). V paměťově nejnáročnější části má tedy Raspberry Pi pořád dostatek místa. Využití paměti je sledováno pomocí linuxového programu htop [22].

Kapitola 6

Závěr

Tato práce se zabývala návrhem a implementací algoritmu pro čtení hodnot na analogovém měřicím přístroji. Naším záměrem bylo, aby byl program použitelný na různých typech měřičů, tedy přesněji na klasických a kvadrantových typech, které jsou nejpoužívanější.

Návrh jsme provedli v softwaru Matlab, který je mimořádně vhodný, protože poskytuje příjemnou práci s maticemi, se kterými v práci hodně počítáme. Kromě detekce ukazatele bylo myšleno i na hledání stupnice, které je velmi důležité v případě, kdy je stupnice nelineárně uspořádaná. Takový případ nastává např. u ohmmetrů. Nalezené dílky a ukazatel jsou pak použity při interpolaci k výpočtu indikované hodnoty.

Poté byl návrh převeden do podoby kódu v jazyce C, který je vyhovující pro embedded platformu. Tou byl zvolen jednodeskový mikropočítač Raspberry Pi. Pomocí něho jsme tedy projekt realizovali a navržené řešení otestovali na skutečném ručkovém měřiči. Experimenty potvrdily funkčnost návrhu. Ta ale závisí na vnějších faktorech.

V praxi se totiž stává, že se kamera vůči měřiči posune nebo natočí (případně naopak). Proto jsme program vybavili invariantností k natočení, ke změně pozice či jasu (osvětlení) snímku. Detekce je úspěšná za dobrých světelných podmínek, za špatných už může mít problémy.

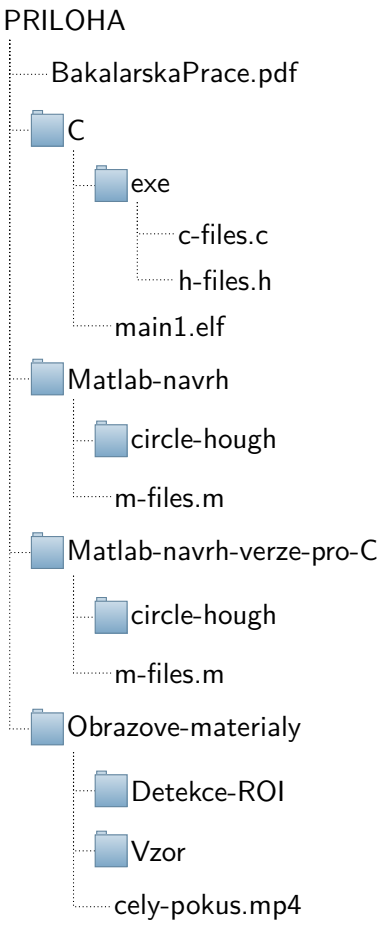
K Raspberry Pi se lze napojit pomocí síťového protokolu Secure Shell z příkazové řádky, což umožňuje sledování hodnot na přístroji např. z jiné místnosti, pokud jsme připojeni ke stejné síti. Proto je uplatnění vhodné např. v laboratořích nebo na místech, kde může být nepraktické chodit sledovat hodnoty.

Možné navázání na práci do budoucna vidím v podobě vyřešení přítomnosti stínu, který může ručička vrhat. Ten může při špatném nasvícení způsobovat špatné výsledky. Zároveň některé měřicí přístroje mají klidovou pozici ručičky jinde, než je předpokládáno v této práci.

Příloha A

Obsah DVD

Na DVD je přiložena elektronická verze bakalářské práce ve formátu PDF a dále zdrojové kódy (oboje Matlab i C). Dále je zde spustitelný soubor ELF. Přítomné jsou i obrazové materiály, se kterými jsme prováděli experimentální část, a to včetně vzoru, který je nahrán v paměti Raspberry Pi. Přítomné jsou i videoukázky.



Příloha B

Literatura a Reference

[1] *MATLAB version 9.8.0.1323502 (R2020a)*. The Mathworks, Inc. Natick, Massachusetts, 2020.

[2] Leo SL Pang a WL Chan. "Computer vision application in automatic meter calibration". In: *Fourtieth IAS Annual Meeting. Conference Record of the 2005 Industry Applications Conference, 2005.* Sv. 3. IEEE. 2005, s. 1731–1735.

[3] Ming Yi et al. "A clustering-based algorithm for automatic detection of automobile dashboard". In: *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2017, s. 3259–3264.

[4] Jiannan Chi et al. "Machine vision based automatic detection method of indicating values of a pointer gauge". In: *Mathematical Problems in Engineering* 2015 (2015).

[5] Yung-Sheng Chen a Jeng-Yau Wang. "Computer vision-based approach for reading analog multimeter". In: *Applied Sciences* 8.8 (2018), s. 1268.

[6] Chris McCormick. *Image Derivative*. Ún. 2013. URL: `https://mccormickml.com/2013/02/26/image-derivative/`.

[7] Deepanshu Tyagi. *Introduction To Feature Detection And Matching*. Led. 2019. URL: `https://medium.com/data-breach/introduction-to-feature-detection-and-matching-65e27179885d`.

[8] Yung-Yu Chuang. *Features*. Univerzitní přednáška. 2020. URL: `https://www.csie.ntu.edu.tw/~cyy/courses/vfx/20spring/lectures/`.

[9] Niki Estner. *Mathematics of Harris corner point detection*. Zář. 2012. URL: `https://dsp.stackexchange.com/a/3339`.

[10] Matej Kollár. "Aplikácia klasterizačných algoritmov na namerané dáta kalibrácií antén". Dis. FEI STU, 2012.

[11] Harvey Rhody. "Lecture 10: Hough circle transform". In: *Chester F. Carlson Center for Imaging Science, Rochester Institute of Technology* (2005).

[12] A Oualid Djekoune, Khadija Messaoudi a Mahmoud Belhocine. "A New Modified Hough Transform Method for Circle Detection." In: *IJCCI*. 2013, s. 5–12.

[13] Chih-Jer Lin et al. “Design of an Image-Servo Mask Alignment System Using Dual CCDs with an XXY Stage”. In: *Applied Sciences* 6.2 (2016). ISSN: 2076-3417. DOI: 10.3390/app6020042. URL: https://www.mdpi.com/2076-3417/6/2/42.

[14] Nobuyuki Otsu. “A threshold selection method from gray-level histograms”. In: *IEEE transactions on systems, man, and cybernetics* 9.1 (1979), s. 62–66.

[15] Michaela KOLOUCHOVÁ. *Morfologické operace ve zpracování obrazu*. 2008.

[16] Thalles Silva. *An Introduction to Morphological Operations for Digital Image Text Classification*. Červ. 2019. URL: https://medium.com/hackernoon/an-introduction-to-morphological-operations-for-digital-image-text-classification-79cb14bab2d7.

[17] David Young. *Hough transform for circles*. Ver. 1.2.0.0. URL: https://www.mathworks.com/matlabcentral/fileexchange/26978-hough-transform-for-circles.

[18] Robert M Haralick a Linda G Shapiro. *Computer and robot vision*. Sv. 1. Addison-wesley Reading, 1992, s. 28–48.

[19] Martin A Fischler a Robert C Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6 (1981), s. 381–395.

[20] Tomáš Svoboda. “Ransac random sample consensus”. In: (lis. 2008), s. 2–18. URL: https://cw.fel.cvut.cz/b172/_media/courses/bxaro/2009svobodaransac.pdf.

[21] Tomáš Werner. “Optimalizace”. Pros. 2019.

[22] Hisham Muhammad. *htop*. Ver. 2.2.0. 10. dub. 2018. URL: http://hisham.hm/htop/index.php?page=downloads.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Běloch** Jméno: **Miroslav** Osobní číslo: **465919**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Čtení ručkového měřicího přístroje pomocí kamery

Název bakalářské práce anglicky:

Reading of an Measuring Device Using Camera

Pokyny pro vypracování:

Na základě analýzy obrazu navrhněte a v prostředí Matlab odlaďte algoritmus pro čtení údaje z analogového panelového měřicího přístroje. Ověřený algoritmus implementujte na vhodné embedded platformě.

Seznam doporučené literatury:

[1] Theodoridis, S., Koutroumbas, K.: Pattern Recognition. ISBN-13: 978-1597492720
[2] www.mathworks.com
[3] www.raspberrypi.org

Jméno a pracoviště vedoucí(ho) bakalářské práce:

prof. Ing. Pavel Zahradník, CSc., katedra telekomunikační techniky FEL

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

Datum zadání bakalářské práce: **01.10.2019** Termín odevzdání bakalářské práce: **22.05.2020**

Platnost zadání bakalářské práce:
do konce letního semestru 2020/2021

prof. Ing. Pavel Zahradník, CSc.		prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	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