



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

ZAŘÍZENÍ PRO NAPODOBENÍ STATICKÝCH A DYNAMICKEJ VLASTNOSTÍ PÍSMA

DEVICE FOR IMITATION OF STATIC AND DYNAMIC HANDWRITING CHARACTERISTICS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAN PAWLUS

VEDOUCÍ PRÁCE
SUPERVISOR

prof. Ing. MARTIN DRAHANSKÝ, Ph.D.

BRNO 2019

Zadání diplomové práce



22229

Student: **Pawlus Jan, Bc.**

Program: Informační technologie Obor: Počítačové a vestavěné systémy

Název: **Zařízení pro napodobení statických a dynamických vlastností písma**

Device for Imitation of Static and Dynamic Handwriting Characteristics

Kategorie: Vestavěné systémy

Zadání:

1. Prostudujte si literaturu týkající se biometrického rozpoznávání statických a dynamických vlastností písma/podpisu, dále pak možnosti, jak s těmito vlastnostmi pracovat.
2. Navrhněte a zkonstruujte snímací zařízení (pero se senzory), které bude umět snímat a ukládat dynamické vlastnosti písma.
3. Navrhněte algoritmický postup pro namapování získaných dynamických charakteristik písma/podpisu na statické a simulujte je na vhodném zařízení.
4. Na menší databázi vzorků ověřte kvalitu napodobeniny písma či podpisu s originálem. K ověření je možné využít mikroskopu a příp. konzultace s písmoznalcem.
5. Dosažené závěry shňte a diskutujte možnosti rozšíření.

Literatura:

- Deselaers T. et al. *Gyropen: Gyroscopes for pen-input with mobile phones*. IEEE Transactions on Human-Machine Systems, 2015, 45.2: 263-271.
- Wróbel M., Starczewski J.T., Napoli C. *Handwriting recognition with extraction of letter fragments*. In: International Conference on Artificial Intelligence and Soft Computing. Springer, Cham, 2017, s. 183-192.
- Dhouibi M., Abdelkrim H., Abdelkrim A. *Neural network modeling and implementation of a handwriting system*. In: 2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET). IEEE, 2018, s. 144-150.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování práce viz <http://www.fit.vutbr.cz/info/szz/>

Vedoucí práce: **Drahanský Martin, prof. Ing., Dipl.-Ing., Ph.D.**

Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.

Datum zadání: 1. listopadu 2018

Datum odevzdání: 22. května 2019

Datum schválení: 24. listopadu 2018

Abstrakt

Tato práce se zabývá navržením a sestrojením systému pro napodobení statických a dynamických vlastností písma. Návrh systému počítá se speciální propiskou, jejíž účelem je sběr informací o vlastnostech písma, následným zpracováním těchto vlastností a jejich napodobením na speciálně upravené 3D tiskárně, na níž je upevněna klasická propiska. Motivací ke vzniku této práce je absence výzkumu zabývajícího se napodobením statických a dynamických vlastností písma jiným způsobem, než lidskou rukou. K tomu, aby se bylo možné bránit takovému falšování písma či podpisu, je nutné rádně znát útok.

Abstract

This project deals with designing and assembling a system for imitation of static and dynamic handwriting characteristics. This system's design takes into account special pen targeted for getting the handwriting characteristics, working with these characteristics and their imitation with a 3D printer altered for this purpose. This topic might be interesting because research about this specific field, which would include a real demonstration of how a signature can be forged using dynamic handwriting characteristics not by forger's hand, barely exists. The problem with preventing forgery is that we need to know the attack well, which is a clear motivation for this project.

Klíčová slova

rozpoznávání písma, rozpoznávání podpisu, podepisovací přístroje, falzifikát podpisu, napodobení podpisu

Keywords

handwriting recognition, signature recognition, signing machines, signature forgery, signature imitation

Citace

PAWLUS, Jan. *Zařízení pro napodobení statických a dynamických vlastností písma*. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Martin Drahanský, Ph.D.

Zařízení pro napodobení statických a dynamických vlastností písma

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana prof. Ing. Martina Drahanského, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čeral.

.....
Jan Pawlus
16. května 2019

Poděkování

Děkuji prof. Ing. Martinu Drahanskému, Ph.D., za vedení, konzultace a cenné poznatky při vypracování této práce. Dále děkuji Ing. Tomáši Goldmannovi za pomoc při 3D tisku a Ing. Michalu Dvořákovi za tvorbu 3D modelu. V neposlední řadě děkuji Marku Širučkovi za asistenci při pájení a zapůjčení *mikropájky* a Mgr. Ondřeji Kovandovi za konzultaci některých matematicko-fyzikálních problémů.

Obsah

1	Úvod	2
2	Rozpoznávání písma a podpisu	4
2.1	Klasifikace systémů	4
2.2	Rozpoznávání písma	6
2.3	Rozpoznávání podpisu	8
2.4	Přehled porovnávacích a rozpoznávacích metod	9
2.5	Podepisovací přístroje	11
2.6	Zpracování obrazu	12
2.7	Senzorika k zaznamenávání pohybu	14
2.8	Post-processing výstupu ze senzorů	16
3	Návrh systému	18
3.1	Požadavky na systém	18
3.2	Výběr komponent systému	19
3.3	Propiska	22
3.4	Analýza získaných dat	24
3.5	Imitace	27
3.6	Implementační software	28
4	Realizace systému	29
4.1	Propiska	29
4.2	Rekonstrukce dynamických vlastností písma	32
4.3	Zpracování originálního písma	35
4.4	Mapování	37
4.5	Generování příkazů G-kódu	41
4.6	Imitace	41
5	Dosažené výsledky	43
5.1	Ukázka použití	43
5.2	Zpětná vazba a potenciální vylepšení	44
6	Závěr	46
	Literatura	48
A	Ukázka výstupu	52

Kapitola 1

Úvod

Díky své dlouholeté historii je podpis v naší společnosti hluboce zakořeněn jako stále nejpožívanější forma projevu souhlasu s právním textem, vyjádřením autorství a jiných. Avšak stejně jako v dalších odvětvích i podpis se postupem času stal terčem falšování za účelem hlavně právního podvodu - člověk k tomu totiž nepotřeboval nákladné vybavení, stačil mu pouze trénink. Problém analýzy ručně psaného textu spadá do oboru *písmoznalectví*, který se zabývá zkoumáním a porovnáváním písemných projevů za účelem potvrzení nebo vyvrácení identity pisatele, a částečně do oboru *grafologie*, který se zabývá projekcí osobnosti a dalších psychických funkcí v ručně psaném písmě.

Nicméně s rozmachem automatizace nastala potřeba analyzovat podpisy strojově, je-li řec o ověření identity jedince. Proto se začaly vyvíjet *biometrické systémy* k rozpoznávání písma či podpisu, což pramenilo v částečný přesun pozornosti tvorby falfifikátů do tohoto digitalizovaného odvětví. Tento fakt je motivací k sestrojení systému, který se zaměřuje na strojové napodobení písma či podpisu, avšak v klasické fyzické formě na papír.

Cílem tohoto projektu je návrh a sestrojení systému, který dokáže automatizovaně zaznamenat potřebné vlastnosti písma či podpisu v reálném čase během psaní, patřičně je zpracovat a následně písmo či podpis napodobit. V prvopočátku je důležité ujasnit si, které vlastnosti je potřeba zaznamenávat, ale zároveň, které vlastnosti bude možné později napodobit, což záleží na zařízení, které bude napodobení provádět. Ideální variantou se stala speciálně upravená 3D tiskárna, a to z důvodu finanční dostupnosti, kdy si dostačující zařízení může člověk koupit a sestrojit s rozpočtem v rádu jednotek tisíců korun českých.

Avšak výběr nenákladné 3D tiskárny s sebou přináší i nezanedbatelná úskalí. 3D tiskárna sice dokáže napodobit rychlosť, trajektorii a částečně i tlak, ale již si neporadí se sklonem, který je důležitou vlastností písma. Za účelem demonstrace principu je však nutné napodobit alespoň některé vlastnosti písma. Proto je třeba navrhnut propisku obsahující vhodnou senzoriku, která bude umět tyto vlastnosti snímat - kromě trajektorie například rychlosť psaní. Jakmile je navržena a sestrojena propiska, je nezbytné nalézt matematický způsob, jak nasnímané údaje zpracovat a reprezentovat jimi vlastnosti písma. Jelikož se nedá očekávat, že zpracovaná data ze senzorů budou dostatečně přesná, je nutné pro zpřesnění tvaru využít obraz originálního písma či podpisu, patřičně jej zpracovat a připojit k němu vlastnosti získané během psaní. Poslední fází pak bude samotné napodobení, kdy je třeba písmo reprezentovat ve formě přijatelné pro 3D tiskárnu.

Jelikož je tento projekt velmi experimentální, neexistují k danému tématu vhodné zdroje, ze kterých by se dalo čerpat - hlavně ve spojitosti s návrhem takového systému. Z tohoto faktu vyplývá, že práce bude probíhat často metodou *pokus-omyl*, což může zapříčinit neúplně ideální výsledky. Vzhledem k rozsahu a složitosti daného tématu se takový

výstup dá očekávat, důležité je však mimo splnění všech bodů zadání prokázat, že systém může na mnou navrženém principu fungovat, shrnout jeho výsledky a navrhnut vylepšení.

Výsledky napodobení budou konzultovány se znalcí v oboru ručně psaného textu. Doufám, že tyto konzultace budou přínosem nejen pro mě v rámci vývoje projektu, ale také pro druhou stranu - objevování nových technik falzifikace je totiž jasnou motivací ke vzniku této práce. Efektivní obranu proti útočníkovi totiž může člověk mít až tehdy, pokud má zmapovány formy útoku.

Kapitola 2 rozebírá aktuální situaci v odvětví rozpoznávání písma či podpisu společně s dalším teoretickým základem pro tuto práci (částečně převzato z [32]), kapitola 3 přináší návrh systému pro napodobení statických a dynamických vlastností písma (částečně převzato z [32]), kapitola 4 popisuje samotnou realizaci a kapitola 5 rozebírá zpětnou vazbu od znalců v oboru ručně psaného písma a navržená vylepšení.

Kapitola 2

Rozpoznávání písma a podpisu

V této kapitole je rozebrána aktuální situace v odvětví rozpoznávání písma či podpisu. Budou zde popsány jednotlivé techniky a metody, které se v současnosti využívají. Následovat budou další teoretické podklady využívané v této práci jako rozbor senzorů k zaznamenávání pohybu a práce s jejich výstupy.

Psaní je jedním z nejstarších a nejvyužívanějších vyjádření člověka. Nejstarší znaky, vyryté do želvích krunýrů, které by se daly považovat za písmo, se datují do 7. tisíciletí př. n. l. v Číně. Od té doby písmo prošlo intenzivním vývojem a v dnešní době je mimo jiné i stejným prostředkem pro komunikaci člověka s počítačem. V reálném světě ale písmo není v podobě, v jaké by s ním mohl pracovat počítač - díky tomu vznikla potřeba rozpoznávání písma.

Oproti tomu podpis je mnohem mladší záležitostí. První využití podpisu se datuje do doby kolem 6. století n. l., avšak standardně se začal hojně využívat až zhruba o tisíc let později. V dnešní době je idea podpisu v naší společnosti hluboce zakořeněna, což je důvodem, proč podpis stále slouží k potvrzení právně závazných textů, jakými jsou typicky smlouvy, či k vyjádření autorství.

Zásadní rozdíl mezi rozpoznáváním písma a podpisu je v tom, že při rozpoznávání písma se dané systémy zaměřují na klasifikaci jednotlivých znaků, utváření slov a vět (jde tedy o *identifikaci* jednotlivých znaků vzhledem k jejich vzorům - písmenům z abecedy). Naopak při rozpoznávání podpisu se systémy soustředí na ověření toho, zda daný podpis psal stejný jedinec, jako jejich vzor (v tomto případě se jedná o *verifikaci*). *Identifikace* u rozpoznávání podpisů zatím nenašla přílišné uplatnění. [11]

2.1 Klasifikace systémů

Základní klasifikací systémů pro rozpoznávání písma či podpisu je, zda systém analyzuje pouze *statické* (tvar) nebo i *dynamické* (rychlosť, přítlak, sklon a jiné) vlastnosti písma či podpisu.

Off-line systémy

K analýze statických vlastností stačí pouze naskenovaný podpis, avšak je zde třeba upřít pozornost k předzpracování obrazového vstupu. To se liší napříč různými implementacemi systémů, obecně se však většinou jedná o kombinaci těchto kroků [17]:

- *morfologické uzavření* (spojení),

- *normalizace velikosti* - zarovnání tak, aby obraz odpovídal velikosti vzoru,
- *rotace* - zarovnání tak, aby byl obraz otočen stejně jako vzor,
- *dolní propust*,
- *prahování* - výběr pixelů, jejichž barevná složka splňuje určitý práh,
- *ztenčení* - transformace shluku bodů do linek s tloušťkou jeden pixel.

On-line systémy

Oproti tomu *on-line* systémy se spoléhají mimo statických vlastností také na dynamické. K tomu ale potřebují dodatečný hardware, jelikož samotný obraz podpisu již nestačí. Obecně se dá říct, že *on-line* systémy vyžadují práci s nástrojem pro zadávání vstupu (propiska, stylus, prst) a nástrojem, který vstup zachycuje (tablet, ale i papír či jiné speciálnizované zařízení). [11]

Tablet (či jakákoli varianta dotykové obrazovky) je nejméně nákladným zařízením hlavně kvůli své rozšířenosti a univerzálnosti. Získávání písma či podpisu zde závisí pouze na aplikaci, kterou systém používá. Teoreticky dokáže zaznamenávat trajektorii a rychlosť, zadávání pak může probíhat buď přímo prstem nebo stylusem. Obyčejný tablet však sám o sobě nedokáže zaznamenávat tlak nebo sklon psaní.

Signature Pads jsou specializovanými zařízeními určenými k zaznamenávání podpisu. Dokáží zaznamenávat dynamické vlastnosti podpisu (rychlosť, přítlak) a zároveň napájet i speciální stylusy se senzorikou, které bývají součástí produktu. Jako příklad může být uveden *Wacom SignaturePad*¹ (obrázek 2.1).



Obrázek 2.1: *Wacom SignaturePad* [převzato z 1].

¹ *Wacom SignaturePad* - <http://signature.wacom.eu/products/signature-pad/>

Stylus je prostředkem k psaní na dotykové plochy. Obyčejný stylus nedokáže sám o sobě zaznamenávat žádné dynamické vlastnosti písma, avšak existují speciální stylusy, které dokáží v kombinaci s patřičným softwarem v tabletu zaznamenávat sklon či přítlak - příkladem může být stylus *HP Tilt Pen*².

SmartPens jsou novým trendem v rozpoznávání písma. Jsou to speciální propisky se zabudovanými senzory sloužící k psaní na obyčejný papír, avšak v průběhu psaní se psaný text ukládá a transformuje do digitální podoby tak, aby uživatel následně mohl za pomocí mobilu, tabletu či počítače vyhledávat ve svých zápisích. Příkladem může být *LiveScribe SmartPen*³ (obrázek 2.3), což je zařízení, které snímá napsaný text s pomocí *infračervené kamery* [37].



Obrázek 2.2: *LiveScribe SmartPen* [převzato z 3].

iskn Slate2+⁴ je specializované zařízení určené ke snímání sklonu a polohy při kreslení. Z technologického pohledu se jedná o prstenec, který se nasazuje na tužku, obsahující 32 magnetometrů po celém obvodu prstence. Z těchto senzorů je v reálném čase spočtena poloha hrotu tužky a sklon, na základě čehož je kreslení digitalizováno. Neumí však zaznamenávat přítlak.



Obrázek 2.3: *iskn Slate2+* [převzato z 4].

2.2 Rozpoznávání písma

Princip rozpoznávání znaků spočívá v jejich lokalizaci v grafickém vstupu a přiřazení vhodného vzoru (písmeno z abecedy či rovnou slovo). Tato potřeba začala vznikat kupříkladu reakcí na vytisklé dokumenty, se kterými neuměl počítač pracovat, zpracování osobních dokladů, rozpoznávání státních poznávacích značek apod.

OCR

První náznaky automatického *off-line* rozpoznávání znaků sahají do 50. let minulého století, výsledky ovšem nebyly nikterak přesvědčivé - zejména z důvodu nekvalitního tisku,

²*HP Tilt Pen* - <https://store.hp.com/us/en/pdp/hp-tilt-pen-p-2my21aa-abl-1>

³*Livescribe SmartPen* - <https://www.livescribe.com/site/livescribe3/>

⁴*iskn Slate2+* - <https://www.iskn.co/eu/features>

využívání nesjednocených typů písma či nevhodných metodik. Tehdejší systémy pracovaly metodou *template matching* (někdy také *matrix matching*), která porovnává pixel po pixelu se vzorem. Na základě skóre shody se vzorem byl vybrán vzorový znak s nejvyšším skóre (na podobném principu funguje rozpoznávání podpisů využívající *Pixel Matching Technique*, popsané v sekci 2.4). Nepříliš dobré výsledky této metody byly způsobeny zmíněnou variabilitou vytiskných typů písma. [38]

Aktuálně využívanou metodou *OCR* je *feature matching* [27], zakládající se na strukturální analýze znaků. Dá se říci, že znak je rozdělen na segmenty, ve kterých je možné identifikovat určité vlastnosti, vztahy mezi vlastnostmi a vztahy mezi segmenty. Tím se *OCR* stává invariantní vůči použití různých typů písma. [38]

K samotné klasifikaci je zde využíváno strojového učení, nejčastěji algoritmu *K-Closest Neighbor*, jenž je popsán v sekci 2.4. *Feature matching* u *OCR* se dá však implementovat různými metodami, další využívanou je například metoda *Hidden Markov Model*, která je popsána v sekci 2.4.

ICR

Intelligent Character Recognition je rozšířením technologie *OCR* soustředící se na ručně psaný hůlkový text který se vyskytuje na předem známých pozicích - uplatnění tedy nachází hlavně ve vyplňených formulářích či dopisech. Pro implementaci klasifikace znaků se používá často algoritmus *K-Closest Neighbor*, který je popsán v sekci 2.4. [10]

Dalším rozšířením je technologie *Intelligent Word Recognition* (*IWR*), která se soustředí na rozpoznávání celých slov namísto jednotlivých znaků.

HWR

Handwriting Recognition cílí na rozpoznávání ručně psaného textu, který se ale oproti *ICR* nemusí vyskytovat na předem známých pozicích a nemusí být napsán hůlkovým písmem. Je zde tedy potřeba mnohem složitější segmentace a předzpracování, čímž se zabývá kupříkladu metoda *Start, Follow, Read* [41], která představuje techniky *Start-Of-Line* hledající začátek řádku a *Line Follower*, jenž následuje nalezený *SOL* a detekuje postupně psaný řádek (i s ohledem na kurzívu), což znázorňuje obrázek 2.4. Pro následnou klasifikaci se používá algoritmus *Recursive Neural Network*, popsaný v sekci 2.4.



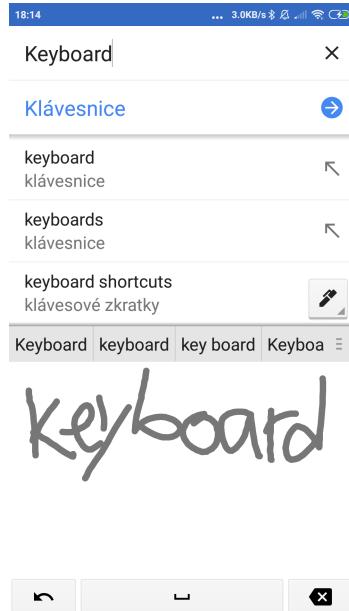
Obrázek 2.4: Techniky *SOL* (červené kolečko) a *LF* (modrá linka) [41, převzato].

On-line metody rozpoznávání písma

Díky rozmachu *smartphonů* je v dnešní době v kurzu také *on-line* rozpoznávání psacího i tiskacího písma za použití mobilního zařízení a prstu (nebo stylusu). Takovou metodu

popisuje například článek [24], s její aplikací je možné se setkat v produktech *Google* (např. *Google Translate* či způsob zadávání textu v systémech *Android* - umí rozpoznávat 97 světových jazyků).

Metoda zaznamenává jak tvar podpisu, tak čas v průběhu psaní. Po předzpracování je nutné text rozdělit na několik segmentů po znacích, k čemuž se využívají techniky *Artificial Neural Networks*. Poté jsou extrahovány lokální a globální rysy napsaných znaků, následná klasifikace je realizována pomocí *Recursive Neural Networks*, kde v některých jazycích (japonština, čínština) je navíc zapojen algoritmus *K-Closest Neighbor* (metody popsány v 2.4). Možnosti každodenního používání jsou ukázány na obrázku 2.5.



Obrázek 2.5: Screenshot rozpoznávání písma v aplikaci *Google Translate*.

2.3 Rozpoznávání podpisu

Podpis je typem vyjádření každého jedince, ale zároveň jde o získanou dovednost. Proto je klasifikován zároveň do anatomických a behaviorálních *biometrických vlastností* jedince [11] (*biometrie* z pohledu informačních technologií představuje automatické rozpoznávání lidí na základě jejich charakteristických anatomických rysů a charakteristického chování. [12])

Díky dlouholetému využívání je velmi dobře akceptován společností (i právním systémem), a proto se stále hojně využívá (témař výhradně pro *verifikaci* - porovnání aktuálního vzorku s jedním referenčním vzorkem [12]). Při vyzrazení je možné jej změnit, což je velká výhoda oproti ostatním biometrickým vlastnostem. Na druhou stranu se u podpisu potýkáme s velkou *vnitrotřídní variabilitou* (člověk se pokaždé podepíše trochu jinak), což pramení jak ve vyšší chybové míry rozpoznávání, tak v nižší rozšířenosti této *biometriky* oproti například rozpoznávání podle otisků prstů. [11]

Vlastnosti podpisu se dělí na statické a dynamické. Statické, zpracovávané *off-line* systémy, jsou zaznamenány jako vyfocený, resp. naskenovaný podpis, a je analyzován pouze jeho tvar. Oproti tomu dynamické, zpracovávané *on-line* systémy, reprezentují rychlosť, přítlač, sklon a jiné, k čemuž je třeba i dodatečný hardware (popsaný v sekci 2.1).

U *off-line* systémů je nutné podpis nejprve předzpracovat (ropsáno v sekci 2.1). V momentě, kdy je k dispozici předzpracovaný podpis, může začít porovnávání se vzorkem z databáze (*verifikace*). Používají se například metody strojového učení *Support Vector Machine*, *Artificial Neural Network* či porovnávací metoda *Pixel Matching Technique* - popis metod lze najít v sekci 2.4. [4]

K *on-line* rozpoznávání podpisu je třeba pracovat s určitými hardwarovými nástroji (ropsané v 2.1). Po získání podpisu je možné extrahovat tzv. rysy. Extrakce rysů probíhá dvěma způsoby - globálně a lokálně. Globální rysy popisují podpis jako celek - mohou to být například průměrný přítlak, průměrná rychlosť psaní, počet *strok* (část podpisu mezi přiložením a zvednutím pera [11]) a jiné. Lokální rysy popisují podpis v určitém momentu psaní (aktuální poloha, rychlosť a další). [34]

Následná *verifikace* pomocí extrahovaných rysů se pak provádí různými metodami, které závisí na konkrétní implementaci systému - často to bývají metody *Hidden Markov Model*, *K-Closest Neighbor*, *Dynamic Time Warping* či *Artificial Neural Networks* [33]. Tyto metody jsou popsány v 2.4. *On-line* systémy jsou oproti *off-line* systémům robustnější - díky analýze více vlastností, než jen samotného tvaru podpisu, je složitější tvorba falzifikátu, zároveň je možné provádět přesnější *verifikaci*.

Jak bylo již zmíněno, problémem podpisu jakožto *verifikačního* nástroje je poměrně snadná možnost zfalšování, bavíme-li se o *off-line* systémech analyzujících pouze tvar podpisu. Jak ukazuje výzkum [29], *off-line* systémy postrádají oproti *on-line* systémům velkou část bezpečnosti, která je utvářena zakomponováním dynamických vlastností podpisu. Z něj také vyplynulo, že lidé účastníci se tohoto experimentu nebyli schopni dosáhnout přes 80% skóre shody s originálem ani skrže trénink, pokud byly dynamické vlastnosti podpisu brány v potaz při *verifikaci*. Tyto informace pramení v požadavek využití dynamických vlastností podpisu v této práci. Bohužel neexistují výzkumy v oblasti automatizovaného falšování podpisu, což je předmětem této práce. Návrh systému tedy bude nutné vytvořit experimentálně.

2.4 Přehled porovnávacích a rozpoznávacích metod

Množství technologií pro rozpoznávání písma či podpisu využívá *strojové učení*, což jsou algoritmy, které se dokáží učit z dat a přizpůsobovat se změnám. Existuje mnoho typů strojového učení, avšak pro účel rozpoznávání písma a podpisu je nutné znát dva modely, *generativní* a *diskriminativní*. [28]

Generativní modely strojového učení

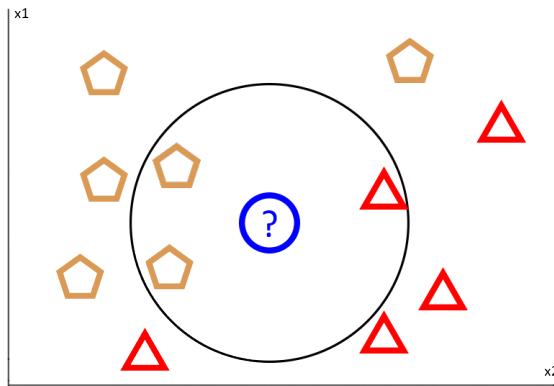
Generativní modely pracují s rozložením jednotlivých tříd a dívají se mimo jiné na to, jak byl model dat vygenerován.

Hidden Markov Model (Skrytý Markovův model) modeluje *Markovův proces*, který má skryté stavы, na kterých je závislý viditelný výstup. Využívá se k predikci sekvence změn stavů na základě sekvence pozorovaných výstupů. Mimo rozpoznávání písma či podpisu se používá také často k rozpoznávání řeči. [42]

Diskriminativní modely strojového učení

Diskriminativní modely se učí hranice mezi jednotlivými třídami a pouze na základě těchto hranic poté klasifikují určitý vzorek. Patří sem následující metody používané pro *verifikaci* podpisů (jedna třída jsou originální podpisy, druhá třída padělky) či *identifikaci* písmen:

K-Closest Neighbor (*K-nejbližších sousedů*) je algoritmus strojového učení, jenž funguje tak, že je zvoleno číslo k (nejčastěji liché, aby sousedi nebyli v poměru 1:1), a podle zastoupení tříd v k nejbližších nalezených sousedech určí výslednou třídu, jak je ukázáno na obrázku 2.6 (v tomto případě pětiúhelník). [8]



Obrázek 2.6: Algoritmus *K-Closest Neighbor*, kde $k = 3$ a třídy mají vlastnosti x_1 a x_2 .

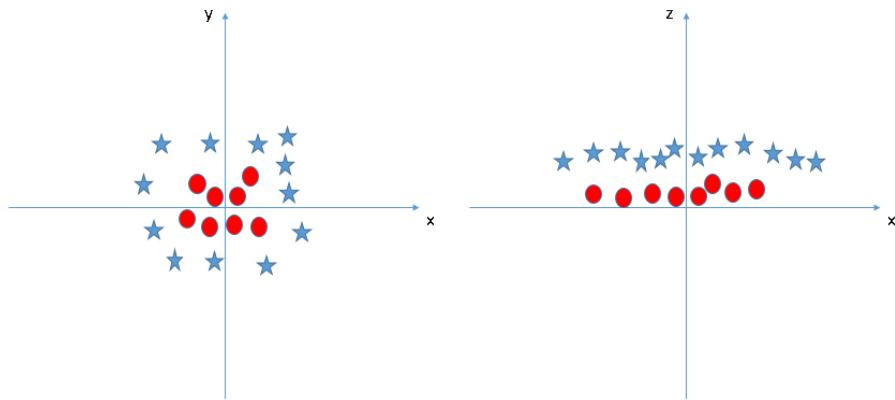
Artificial Neural Networks jsou neuronové sítě, v případě rozpoznávání písma a podpisu natrénované za pomocí algoritmu *back-propagation*. [23]

Recurrent Neural Network (*rekurzivní neuronové sítě*) pracují na rekurzivním principu - výstup z neuronů skryté vrstvy je přiváděn zpět na vstup neuronů vstupní vrstvy, což pramení v kvalitnější, ale pomalejší klasifikaci. [15]

Support Vector Machine je binárním klasifikátorem pracujícím s učitelem (*supervised learning*). Algoritmus se snaží vykreslit každou datovou položku jako bod v n -dimenzionálním prostoru (n značí počet vlastností), kde hodnota každé vlastnosti představuje hodnotu dané souřadnice. Následně je nalezena tzv. *hyperroviná*, která má za úkol oddělit třídy co nejlépe, ale zároveň musí mít co největší okraj (*margin*) od obou tříd. Pokud nelze nalézt lineární *hyperrovinu*, *SVM* přidá další vlastnost (rozměr) - např. na obrázku 2.7 je přidán rozměr $z = x^2 + y^2$. [40]

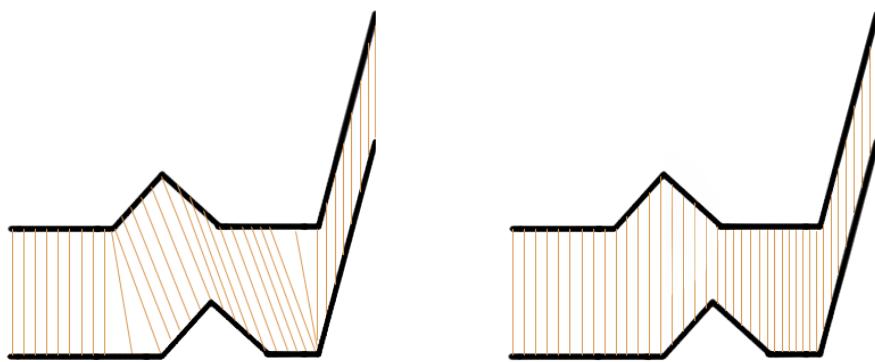
Ostatní metody

Pixel Matching Technique je jednoduchou technikou, jež srovnává předzpracovaný podpis se svým vzorem pixel po pixelu. Pro každý bílý, resp. černý pixel inkrementuje počítadlo shodných bílých, resp. černých pixelů. Výsledkem je poté porovnání počitadel s určitým prahem. [4]



Obrázek 2.7: Příklad nelineární sady dat před a po přidání rozměru [40, převzato]).

Dynamic Time Warping je algoritmus pro srovnávání podobnosti dvou vzorků dat nasbíraných v čase. Jeho výhodou oproti například *Eukleidovské vzdálenosti* je fakt, že je invariantní vůči roztahování či smršťování časové osy, jak je ukázáno na obrázku 2.8. Stejně jako *HMM* se používá při rozpoznávání řeči a výhodou je absence potřeby velké trénovací sady. [3]



Obrázek 2.8: Porovnání dvou sad dat pomocí *Dynamic Time Warping* a *Eukleidovské vzdálenosti*.

2.5 Podepisovací přístroje

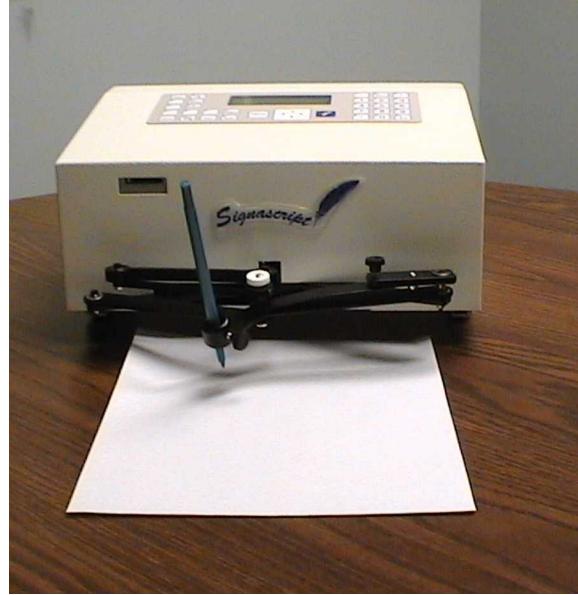
Historie podepisovacích přístrojů (*signing machines*) sahá k počátkům 19. století, kdy se objevil první patent takového přístroje. Hned záhy jej začal používat pro podepisování dokumentů *Thomas Jefferson*. Starší, mechanické verze pracovaly s plastickým modelem podpisu, který replikovaly.

Novější verze již využívají elektronického modelu podpisu uloženého na paměťovém úložišti, avšak jelikož jde téměř výhradně o komerční řešení (jde kupříkladu o *Damilic*⁵, obrázek 2.9, či *Automated Signature Technology*⁶), výrobci neuvádějí podrobnější popis

⁵Damilic - <http://www.realsig.com/index.htm>

⁶Automated Signature Technology <http://signaturemachine.com/>

technologií, například jaké informace jsou uloženy v modelu, zda je využíváno i dynamických vlastností podpisu a jiné.



Obrázek 2.9: Ukázka podepisovacího přístroje od firmy *Damilic* [5, převzato].

2.6 Zpracování obrazu

V *off-line* systémech pro rozpoznávání písma a podpisu je jedním z klíčových úkonů zpracování obrazu obsahující dané písmo či podpis, a to díky nutnosti získat určitou standardizovanou formu daného obrazu tak, aby bylo možné porovnávat vzorky s originálem. V následujícím textu budou rozebírány takové úkony, které budou využity i v této práci.

Prahování neboli *thresholding* je formou *binarizace* obrazu, tedy převodem obrazu do černé a bílé. Existuje množství různých metod implementujících *prahování*, avšak princip mají všechny metody společný - porovnávání hodnot jednotlivých pixelů s hodnotou zvanou *práh* (*threshold*). Na základě toho, zda je hodnota pixelu menší, či větší než *práh*, je pixel nahrazen buď bílým, nebo černým pixelem. [35]

Rotace spočívá v aplikaci transformační matice R (2.1) na všechny obrazové body. Výsledkem je obraz otočený o úhel α . [44]

$$R = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \quad (2.1)$$

Ztenčení neboli *thinning* označuje algoritmus, který se snaží ztenčit shluky pixelů tak, aby byla výsledkem jediná linie o tloušťce jednoho pixelu. V současné době se hojně využívají především dva algoritmy: *Guo-Hall* a *Zhang-Suen*.

Zhang-Suen

Algoritmus *Zhang-Suen* funguje na následujícím principu. Předpokládejme, že černý pixel má hodnotu 1, zatímco bílý hodnotu 0. Mějme tabulkou reprezentující okolí bodu P_1 :

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5

Následně definujme dvě hodnoty:

$A(P_1)$ = počet přechodů z bílého pixelu do černého v pořadí $P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_2$.

$B(P_1)$ = počet černých sousedních pixelů bodu P_1 .

Každý pixel je následně testován na tyto podmínky:

- pixel je černý a má 8 černých sousedů,
- $2 \leq B(P_1) \leq 6$
- $A(P_1) = 1$
- alespoň jeden z bodů P_2, P_4 a P_6 je bílý
- alespoň jeden z bodů P_4, P_6 a P_8 je bílý

Po celém průchodu obrazem jsou poté nastaveny pixely splňující alespoň jednu z těchto podmínek na bílý. V dalším kroku jsou pixely testovány stejně až na poslední dvě podmínky, které se změní následovně:

- alespoň jeden z bodů P_2, P_4 a P_8 je bílý
- alespoň jeden z bodů P_2, P_6 a P_8 je bílý

Po celém průchodu obrazem jsou opět vyhovující pixely nastaveny na bílé. Tento postup pokračuje až do té doby, než nejsou změněny žádné pixely. Algoritmus přejat z [43].

Guo-Hall

Algoritmus *Guo-Hall* je oproti algoritmu *Zhang-Suen* méně intuitivní a poněkud složitější. Mějme opět tabulkou okolí bodu P_1 :

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5

Algoritmus dále definuje několik hodnot, na základě kterých se rozhoduje:

$$C(P_1) = \neg P_2 \wedge (P_3 \vee P_4) + \neg P_4 \wedge (P_5 \vee P_6) + \neg P_6 \wedge (P_7 \vee P_8) + \neg P_8 \wedge (P_1 \vee P_2)$$

$$N_1(P_1) = (P_9 \vee P_2) + (P_3 \vee P_4) + (P_5 \vee P_6) + (P_7 \vee P_8)$$

$$N_2(P_1) = (P_2 \vee P_3) + (P_4 \vee P_5) + (P_6 \vee P_7) + (P_8 \vee P_9)$$

$$N(P_1) = \min[N_1(P_1), N_2(P_1)]$$

Algoritmus následně prochází celým obrázkem a rozhoduje se, zda nahradí černý pixel bílým, což nastane, pokud platí podmínky:

- $C(P_1) = 1$
- $2 \leq N(P_1) \leq 3$
- $(P_2 \vee P_3 \vee \neg P_5) \wedge P_4 = 0$ v lichých iteracích,
- $(P_6 \vee P_7 \vee \neg P_9) \wedge P_8 == 0$ v sudých iteracích.

Algoritmus skončí, pokud není změněn v iteraci ani jeden bod (přejato z [16]). Výsledek ztenčení za pomocí algoritmu *Guo-Hall* lze vidět na obrázku 2.10.



Obrázek 2.10: Ukázka vstupu a výstupu algoritmu *Guo-Hall*.

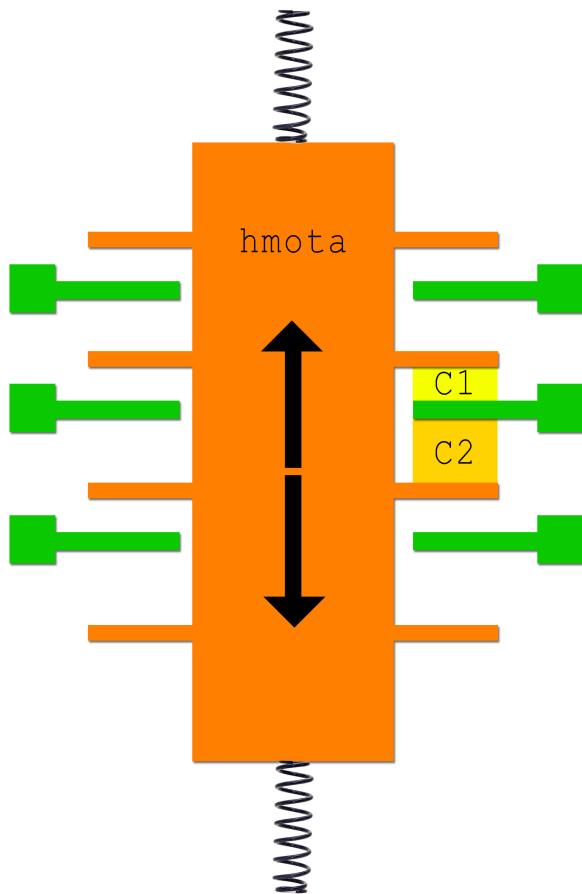
2.7 Senzorika k zaznamenávání pohybu

Jelikož bude náplní tohoto projektu práce s pohybem propisky, je nezbytné rozebrat vhodnou senzoriku k zaznamenávání pohybových informací. K tomuto účelu slouží senzory nazývající se *Inertial Measurement Units (IMU)*, jejichž úkolem je zaznamenávat gravitační síly působící na senzor (pohyb), úhlové zrychlení či magnetické pole. Těmito senzory jsou *akcelerometr, gyroskop a magnetometr* [19], který však pro potřeby této práce lze vynechat.

V dnešní době jsou nejdostupnějším typem senzory vyrobené technologií *MEMS (Micro-ElectroMechanical Systems)*, jejíž základní myšlenkou je kombinovat mechanické principy s elektronickými, využívajícími křemík. Výhodou této technologie je relativně nízká výrobní cena a velmi malý rozměr senzoru. [26]

Akcelerometry slouží k měření síly, která působí na měřené těleso. Samotné měření je nejčastěji realizováno buď na základě *piezoelektrického jevu*, či *kapacitních detektorů polohy*. *Akcelerometry* pracující na *kapacitním* principu pak fungují na základě měření *kapacit* (C_1 , C_2) měnících se díky pohybu hmoty, jež je upevněna na pružinách (viz obrázek 2.11). [7]

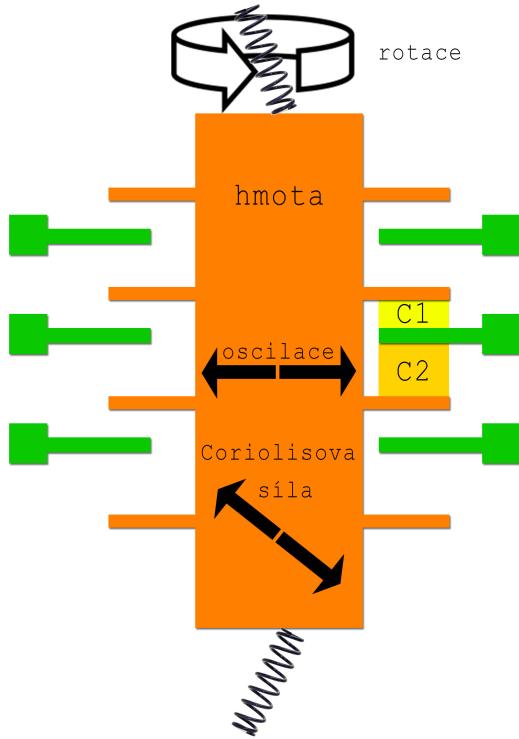
Výstupem kombinací tohoto principu ve třech dimenzích je poté možné získat informaci o působení síly na senzor. Zároveň je však nutné počítat i s gravitační silou působící na těleso díky gravitaci Země. Po odstranění tohoto vlivu je výstupem tzv. *lineární akcelerace*, ze které je po integraci podle času možné získat rychlosť, po další integraci podle času také polohu. [7]



Obrázek 2.11: Ukázka principu měření *kapacitního akcelerometru*.

Gyroskop je senzor určený k měření úhlové rychlosti. Princip měření spočítá v tom, že hmota je oproti *akcelerometru* neustále v pohybu (*oscilace*), díky čemuž na hmotu začne při rotaci tělesa působit *Coriolisova síla*. Změnu měřených kapacit (C_1, C_2) pak udávají posuvy hmoty na základě této síly (viz obrázek 2.12). Výsledkem takového měření jsou hodnoty $\text{rad} \cdot \text{s}^{-1}$, kdy po integraci podle času dostáváme hodnotu v *radiánech* popisující natočení *gyroskopu* v dané ose. [13]

IMU moduly nachází využití v letadlech, autech, dronech, mobilních telefonech a v mnoha dalších. *MEMS* senzory se však potýkají s poměrně vysokou chybou měření, která je způsobena z velké části šumem. Pokud je například výstup *akcelerometru* integrován, chyba se projeví lineárně, po další integraci dokonce exponenciálně. Proto se tyto senzory většinou nepoužívají k měření polohy, nebo bývají kombinovány například s *GPS* systémy. Pro systémy, kde je třeba vysoká přesnost, se využívají senzory fungující na principu *optických vláken*, jejich pořizovací cena je však mnohem vyšší. [30]



Obrázek 2.12: Ukázka principu měření *kapacitního* gyroskopu.

2.8 Post-processing výstupu ze senzorů

Samotný výstup ze *IMU* senzorů je ve většině případů nutné patřičně upravit například aplikací vhodných filtrů k eliminaci šumu a jiných rušivých elementů. Existuje velké množství možností, jakými lze výstupy ze senzorů upravit, tato sekce se však zaměří na takové úpravy, které se více či méně dotýkají této práce.

Filtry

Dolní propust je původně analogovým filtrem, který byl realizován za pomocí hardware. Jeho originálním účelem bylo odstranění vysokých frekvencí z analogového signálu [5]. Našel však uplatnění i v digitální podobě. [39]

Savitzky-Golay filtr je digitální filtr využívaný za účelem vyhlazování sekvence digitálních dat. Toho je dosaženo díky *konvoluci*, kdy jsou navazující úseky vstupních dat prokládány *polynomem* za použití *metody nejmenších čtverců*. Z proloženého *polynomu* jsou poté ohodnoceny body vyhlazeného signálu. Parametry takového filtru jsou *stupeň polynomu* a *velikost okna*, tedy délka každého úseku vstupních dat, který je prokládán *polynomem*. [36]

Median filter je *nelineárním* digitálním filtrem, jehož výstupem pro každou hodnotu je *medián* spočtený z okolních hodnot, jak ukazuje vzorec 2.2, kde x_n značí vstupní hodnotu, y_n výstupní hodnotu a k udává velikost okna, což je také jediný parametr tohoto algoritmu. [21]

$$y_n = \text{med}[x_{n-k}, x_{n-k+1}, \dots, x_n, \dots, x_{n+k-1}, x_{n+k}] \quad (2.2)$$

Fúze senzorů

Dalším přístupem pro zkvalitnění výstupu senzorů může být jejich *fúze*. V principu jde o kombinaci výstupů dvou různých senzorů, které spolu však nějakým způsobem souvisí, vedoucí k jejich zpřesnění. V případě *IMU* je to kombinace *akcelerometru* a *gyroskopu*.

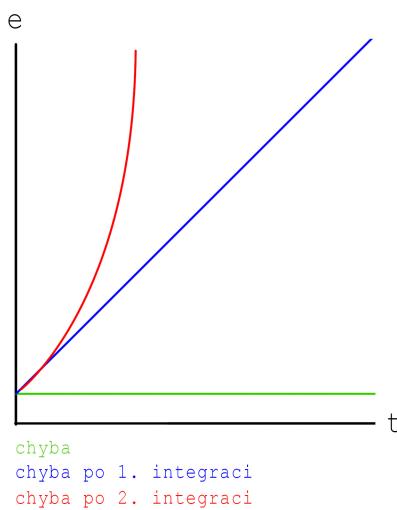
Kalmanův filtr je přesně takovým typem filtru. Ač je jeho pozadí matematicky složité, principem je jednoduchý - na základě vstupní hodnoty jednoho senzoru dochází k předpovědi výstupu na základě jak jeho hodnoty, tak na základě hodnoty druhého senzoru, který s prvním určitým způsobem souvisí. Příkladem může být zpřesnění jedné osy *akcelerometru* na základě korespondující osy *gyroskopu*, čímž se dá částečně zredukovat šum a chyba. [18]

Complementary filter je mnohem jednodušší a rychlejší, než *Kalmanův filtr*. V případě *IMU* se jedná o zpřesnění výstupu *gyroskopu* pomocí *akcelerometru* tím způsobem, že výsledná hodnota rotace v dané ose je spočtena váženým součtem hodnot obou senzorů (například dle vztahu 2.3, kde $gyro_{out}$ značí výstup *gyroskopu* v jedné ose, acc_{out} výstup *akcelerometru* v jedné ose). [18]

$$gyro_{out} = 0,98 \cdot gyro_{out} + 0,02 \cdot acc_{out} \quad (2.3)$$

Odstranění trendu

V případě, kdy je senzor již řádně zkalibrován a i přesto vykazuje chybu či šum i v kličkovém stavu, je možné vyzkoušet *odstranění trendu*, hlavně tehdy, kdy je třeba výstup senzoru integrovat. První integrace totiž způsobí lineární promítnutí chyby do výsledku, druhá dokonce exponenciální (obrázek 2.13). [6]



Obrázek 2.13: Ukázka promítnutí chyb po integraci výstupu ze senzoru.

Pokud výstup není integrován, stačí chybu zprůměrovat a odečít od výstupu. Pokud však výstup integrován je, je třeba chybu zintegrovat, proložit ji *polynomem* takového stupně, kolikrát je třeba výstup senzoru integrovat, a následně odečítat hodnoty *polynomu* chyby od zintegrovaného výstupu senzoru v odpovídajících si časech. [6]

Kapitola 3

Návrh systému

Tato kapitola se zabývá návrhem systému pro napodobení statických a dynamických vlastností písma a podpisu. Systém se skládá z několika částí, a to ze zařízení napodobující podpis, ze speciální propisky, která dané statické a dynamické vlastnosti snímá za pomocí vhodných senzorů, a z *Arduina*, které výstupy ze senzorů zpracovává.

Jelikož jde o experimentální projekt, nejsou k nalezení existující řešení. Pro ilustraci si lze tento systém představit jako kombinaci podepisovacích přístrojů a *SmartPens* či *iskn Slate2+* (oboje popsány v sekci 2.1).

Zpočátku je nutné specifikovat požadavky na systém, vybrat komponenty, které bude systém obsahovat, a na základě těchto komponent poté navrhnout propisku, která bude snímat statické a dynamické vlastnosti písma či podpisu. Nakonec je možné navrhnout práci s výstupem této propisky a samotné napodobení.

3.1 Požadavky na systém

Ze zadání práce vyplývá několik požadavků na vyvýjený systém. Aby bylo možné napodobit výše uvedené vlastnosti písma či podpisu, je nutné je nejprve zaznamenat.

Dynamické vlastnosti písma se dají získat různými způsoby. *On-line* systémy pro rozpoznávání písma a podpisu využívají pro získání těchto vlastností například tablet se stylusem (viz sekce 2.3), kdy jsou tyto vlastnosti zaznamenávány rovnou v digitální podobě. Tato práce však pojímá písmo tradičním způsobem, a to na papír. Jedním ze zásadních požadavků se tedy stala potřeba sestrojit propisku, se kterou bude možné psát klasickým způsobem, tedy na papír, zároveň však tato propiska musí být schopná zaznamenávat dynamické vlastnosti písma.

Statické vlastnosti písma lze získat pomocí zpracování obrazu písma a jedná se převážně o tvar. Tento požadavek koresponduje s činností *off-line* systémů pro rozpoznávání písma a podpisu. Jedná se o sken či foto originálního písma a jeho patřičné zpracování, popsané v sekci 2.6.

Dále je třeba spojit získané statické a dynamické vlastnosti písma a utvořit z nich digitální reprezentaci daného písma tak, aby jej bylo možné napodobit. K tomu je třeba patřičně zpracovat data zaznamenané propiskou a korektně je namapovat na tvar písma získaný ze zpracovaného obrazu.

Posledním požadavkem je samotná realizace napodobení písma, k čemuž je nutné vybrat správný přístroj. Na základě výběru tohoto přístroje je dále nutné převést reprezentaci písma do takové formy, aby jej bylo možné napodobit.

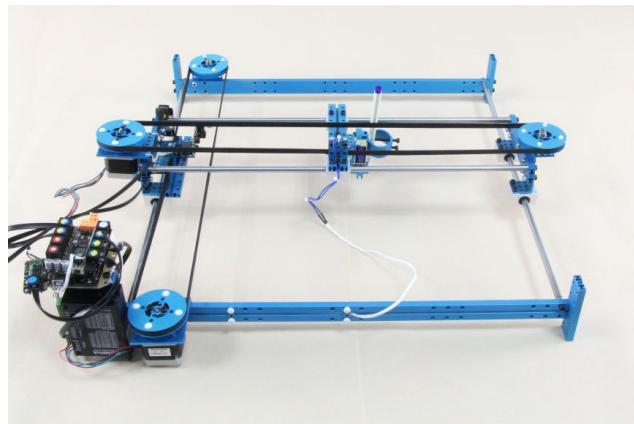
3.2 Výběr komponent systému

Prvotní věcí, kterou bylo třeba při návrhu tohoto systému řešit, bylo to, jaké vlastnosti písma a podpisu bude vhodné napodobovat. Ty jsou omezeny hlavně přístrojem, který bude provádět napodobení.

Přístroj k napodobování

Pro výběr zařízení, které bude napodobovat písmo či podpis, přicházelo v úvahu několik možností.

Plotter je přístroj pracující ve dvou rozměrech, jenž vykresluje propiskou či jiným nástrojem zadaná vstupní data (obrázek 3.1). Díky tomu, že umí pracovat s propiskou či jiným nástrojem, se jeví jako ideální kandidát, stejně jako díky tomu, že jej lze i poměrně jednoduše sestavit¹. Jeho limitací je však práce ve dvou rozměrech. Sice by bylo možné napodobit rychlosť v určitých úsecích, ale jelikož je podpis často rozdělen na více *stroku*, nebylo by možné zvedat propisku z povrchu a opět ji přikládat na jiném místě, což je příliš velkou překážkou pro jeho použití.



Obrázek 3.1: Ukázka 2D plotteru [1, převzato].

Robotická paže pracuje ve třech rozměrech, umí tedy napodobit více *stroku* podpisu. Zároveň by s ní bylo možné napodobit i sklon psaní spolu s rychlosťí v různých úsecích. Nevýhodou je však poněkud složitější ovládání. Pořizovací cena takového zařízení, které by bylo dostatečně stabilní a silné k imitaci písma, by navíc byla mimo rozpočet tohoto projektu².

¹How to make a XY plotter with Makeblock - <https://www.instructables.com/id/How-to-make-a-XY-plotter-with-Makeblock/>

²ReactorX 200 Robot Arm - <https://www.trossenrobotics.com/reactorx-200-robot-arm.aspx>



Obrázek 3.2: Ukázka robotické paže [2, převzato].

3D tiskárna je zlatou střední cestou mezi předchozími variantami. V dnešní době je již pořízení málo nákladné, umí pracovat ve třech rozměrech a umí měnit rychlosť v jednotlivých úsecích. Díky velké popularitě také existuje množství různých *firmwarů*, což ulehčuje práci se zařízením. Tím, že není určena k účelu tohoto projektu, se dá zároveň využít i pro její originální účel - 3D tisk. Nevýhody spočívají v tom, že s její pomocí není možné napodobit sklon psaní a že je potřeba ji patřičně upravit pro držení propisky. I přes tyto nevýhody byla 3D tiskárna vybrána jakožto nejvhodnější přístroj k napodobení vlastností písma či podpisu.

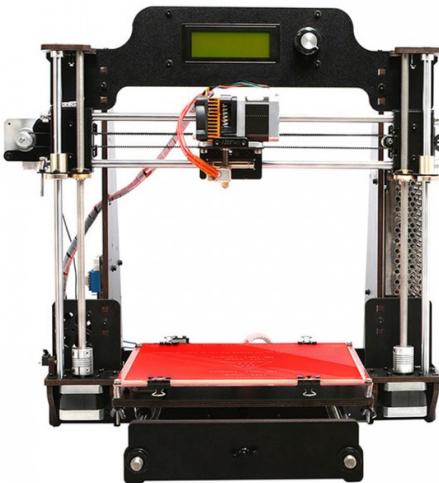
3D tiskárny pracují na principu třech krokových motorů, kde každý ovládá jednu ze tří os. Samotný tisk poté probíhá skrze *extruder*, což je součást tiskárny zahřívající *filament* (tiskový materiál). Díky zmíněným krokovým motorům lze pak *extruder* (který obsahuje čtvrtý krokový motor, jenž má za úkol vylučovat *filament*) dostat do jakéhokoliv bodu nad tiskovou plochu. [1]

Obrátil jsem se na svého známého z břeclavské firmy *VALVAN s r. o.* zabývající se 3D tiskárnami a vytvořil specifikaci takové tiskárny, která bude zakoupena *Fakultou informačních technologií VUT v Brně* k účelu této práce. Principem specifikace je to, že na *extruderu* bude umístěn flexibilní držák pro propisku či jiný nástroj. Díky krokovým motorům pak bude možné simulovat pohyb propisky ve třech rozměrech. Tato tiskárna je založena na modelu *Geeetech Prusa I3 Pro*³ (obrázek 3.3). Nejdůležitějším parametrem pro tuto práci je nejmenší možný krok, který dokáží krokové motory provést. V případě této tiskárny je nejmenším krokem $0,1\text{ mm}$, trajektorii písma či podpisu je tedy teoreticky možné napodobit s touto přesností.

Důležitou vlastností této 3D tiskárny je také zvolený *firmware*. Rozhodl jsem se pro *open-source firmware Marlin*⁴, jelikož se intenzivně vyvíjí, nabízí pohodlné ovládání zařízení a existuje k němu kvalitně zpracovaná dokumentace.

³ Geeetech Prusa I3 Pro - <https://www.geeetech.com/specialpage/i3/i3.html>

⁴ Marlin firmware - <http://marlinfw.org/>



Obrázek 3.3: *Geeetech Prusa I3 Pro* [3, převzato].

Senzory a elektronika ke snímání

Dalším krokem je výběr vhodných komponent k sestrojení propisky snímající dynamické vlastnosti písma. Vzhledem k výběru 3D tiskárny jakožto napodobovacího nástroje vyplývá, že propiska by měla umět snímat rychlosť (zrychlení) v jednotlivých úsecích písma, ale zároveň je nutné, aby poskytovala možnost následné rekonstrukce trajektorie. Podobná řešení byla popsána v kapitole 2.1, avšak od podobných přístupů jako snímání s pomocí *infračervené kamery* či velkého množství *magnetometrů* bylo upuštěno na základě přílišné složitosti či vysoké pořizovací ceny. Aby bylo vůbec možné stihnout splnit všechny body zadání, výběr padl na *akcelerometr* a *gyroskop* (popsány v sekci 2.7).

Nejvhodnějším kandidátem z kategorie nejdostupnějších senzorů se stal modul *MPU-6050* (obrázek 3.4), který kombinuje *akcelerometr* a *gyroskop*, k tomu ale přidává speciální *Digital Motion Processor (DMP)*, jenž umí v reálném čase velmi rychle pracovat s nasbíranými daty ze senzorů a post-processingem z nich vyrobit užitečná data, kterými se budu zabývat v následujících sekcích. [2]



Obrázek 3.4: Modul *MPU-6050* [2, převzato].

Mimo snímání samotných vlastností písma či podpisu je ale také nutné detektovat samotné psaní (případně jednotlivé *stroky*), k čemuž byl vybrán *mikrospínač*, který sepne,

když na něj zatlačí písící tuha. Nabízela se varianta využití *tlakoměru*, který by stejně jako *mikrospínač* dokázal detektovat psaní a jednotlivé *stroky* a přinášel by navíc další dynamickou vlastností písma - přítlak, avšak vzhledem ke složitosti práce byl zařazen do potenciálních vylepšení. Veškerá elektronika bude připojena kabelem k *Arduinu* (model *Uno*⁵), které bude číst potřebné informace a poté s nimi dále pracovat.

3.3 Propiska

Po výběru potřebné elektroniky ke snímání je možné začít s návrhem samotné propisky. Klíčovým problémem je zde umístění této elektroniky v propisce. Ta v sobě musí ukrývat jak *mikrospínač*, tak modul *MPU-6050*. Zároveň ale musí plnit standardní funkci propisky, kterou je psaní.

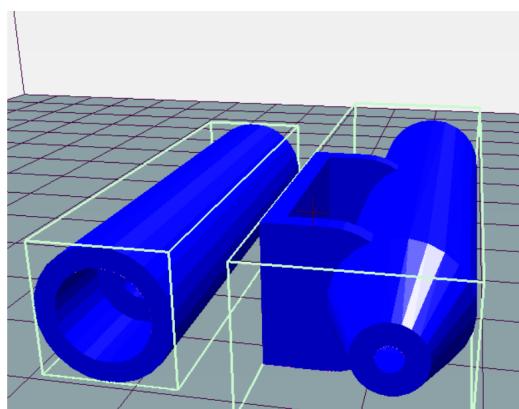
Torzo

Torzo propisky je rozděleno na dvě oddělitelné části, aby byl možný přístup do vnitřní části například za účelem bezproblémové výměny náplně.

Spodní část obsahuje na povrchu kryt pro modul *MPU-6050*. Dále je třeba brát v potaz míry standardní náplně vyskytující se v propiskách, aby přesně pasovala do spodního otvoru - ten musí omezit propustnost náplně tak, aby prošel pouze kovový hrot, kterým se píše.

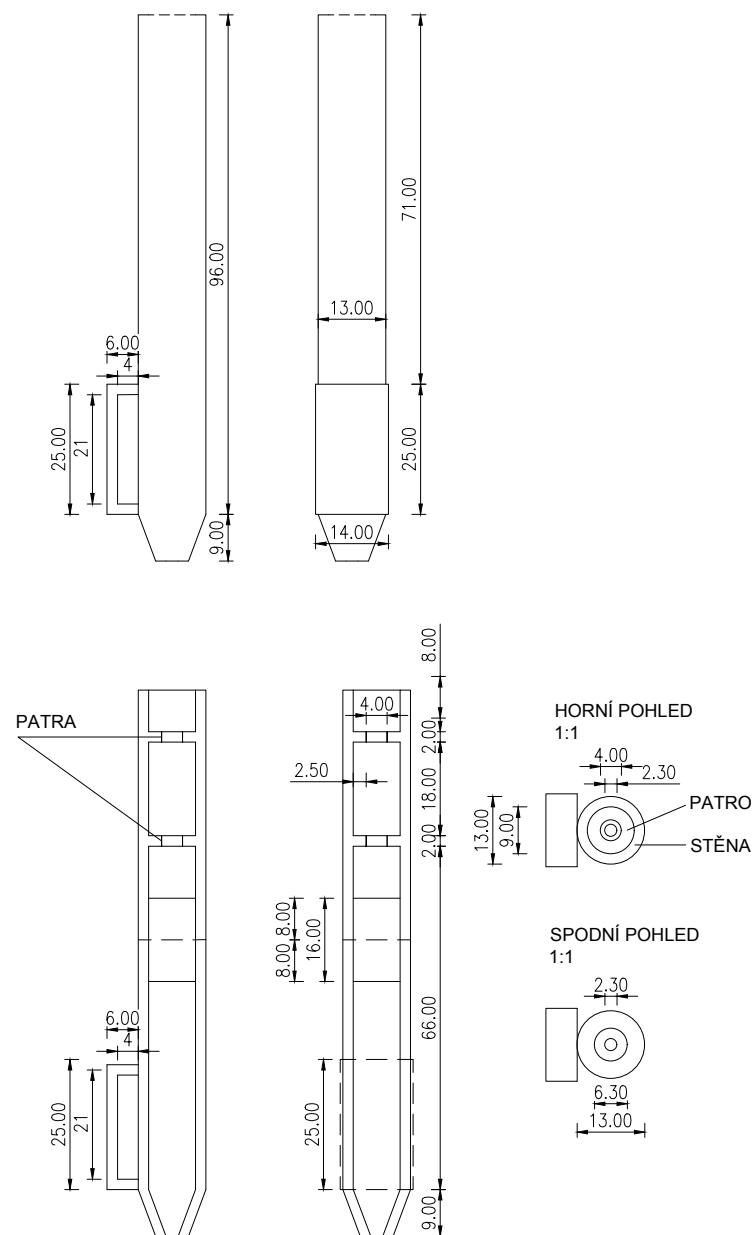
Vrchní část ukrývá dvě patra, z nichž druhé slouží k usazení *mikrospínače* tak, aby byl sepnut, pokud na něj začne tlačit náplň, tedy když se s propiskou píše. Spodní patro slouží k usměrnění náplně tak, aby byla vystředěna v rámci torza.

Technické schéma takového návrhu lze vidět na schématu 3.6, vytvořený 3D model určený k 3D tisku na obrázku 3.5.



Obrázek 3.5: 3D model propisky pro snímání statických a dynamických vlastností písma.

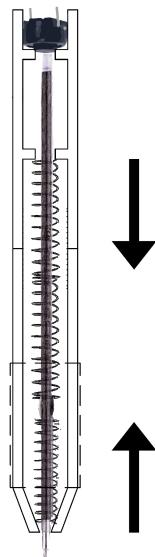
⁵Arduino Uno - <https://store.arduino.cc/arduino-uno-rev3>



Obrázek 3.6: 2D návrh propisky pro snímání statických a dynamických vlastností písma (rozměry v milimetrech).

Mechanismus

Ve spodní části torza klasické propisky figuruje pružina, která je zachycena o plastové výstupky náplně tak, aby tlačila tuhu směrem nahoru. Díky tomu lze propisku otevřít a zavírat. Vzhledem k návrhu propisky pro snímání vlastností písma či podpisu je ale potřeba přidat pružinu i do vrchní části torza tak, aby pružiny tlačily náplň jak směrem nahoru, tak dolů - je tomu tak proto, aby byl po skončení psaní rozepnut *mikrospínač*. Vrchní pružina je zachycena zespodu výstupky na tuze, ze shora prvním patrem v torzu propisky. Tento mechanismus je znázorněn na obrázku 3.7.



Obrázek 3.7: Mechanismus pružin. Šipky udávají směr síly, která díky pružinám tlačí náplň směrem vzhůru, respektive dolů.

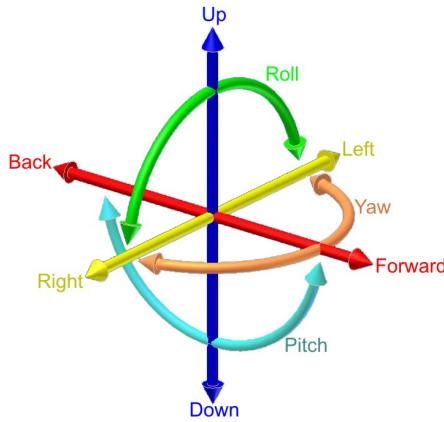
Propiska v této podobě by samozřejmě nemohla být využita k podstrčení člověku za účelem zaznamenání jeho podpisu, avšak pokud by byly senzory zakomponovány přímo vně torza a přenášení dat by bylo realizováno bezdrátově, bylo by možné propisku takto zneužít.

3.4 Analýza získaných dat

Po adekvátním návrhu propisky je třeba navrhnout práci s výstupními daty senzorů. Díky zabudovanému *DMP* v modulu *MPU-6050* existuje několik variant, jak zpracovat data tak, aby poskytla náležité vlastnosti písma. Rekonstruovanými vlastnostmi písma budou rychlosť, trajektorie, kterou je třeba získat kvůli nutnosti informace o pořadí jednotlivých *strok* a bodů na nich, a sklon, který je třeba získat kvůli rekonstrukci trajektorie, i když jej kvůli výběru 3D tiskárny nebude možné napodobit.

Rekonstrukce sklonu

Jedním z možných výstupů, které poskytuje *DMP*, jsou úhly *yaw*, *pitch* a *roll* (spočtené z dat *gyroskopu*), jež udávají rotaci tělesa kolem své osy v osách X, Y a Z (obrázek 3.8). Ve své podstatě jsou tyto úhly integrací výstupu *gyroskopu* podle času. [25]



Obrázek 3.8: Ukázka významu úhlů *yaw*, *pitch* a *roll* [22, převzato].

V každém měřeném momentu lze tedy sestrojit matice rotací tělesa. Pro osu Z ($yaw = \alpha$) to bude matice $R_z(\alpha)$ (vztah 3.1):

$$R_z(\alpha) = \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3.1)$$

Pro osu Y matice $R_y(\beta)$ (vztah 3.2, *pitch* = β):

$$R_y(\beta) = \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix} \quad (3.2)$$

Pro osu X matice $R_x(\gamma)$ (vztah 3.3, *roll* = γ):

$$R_x(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & -\sin \gamma \\ 0 & \sin \gamma & \cos \gamma \end{pmatrix} \quad (3.3)$$

Pro získání orientace propisky v 3D prostoru je třeba vynásobit matice rotace *yaw*, *pitch* a *roll* - získaná matice R (vztah 3.4, $R_z(\alpha)R_y(\beta)R_x(\gamma) = R$): [25]

$$R = \begin{pmatrix} \cos \alpha \cos \beta & \cos \alpha \sin \beta \sin \gamma - \sin \alpha \cos \gamma & \cos \alpha \sin \beta \cos \gamma + \sin \alpha \sin \gamma \\ \sin \alpha \cos \beta & \sin \alpha \sin \beta \sin \gamma + \cos \alpha \cos \gamma & \sin \alpha \sin \beta \cos \gamma - \cos \alpha \sin \gamma \\ -\sin \beta & \cos \beta \sin \gamma & \cos \beta \cos \gamma \end{pmatrix} \quad (3.4)$$

Rekonstrukce rychlosti psaní

K analýze rychlosti písma slouží *akcelerometr*, který poskytuje údaje o zrychlení ve třech osách. Pokud je zrychlení zintegrováno podle času, výsledkem je rychlosť, po další integraci je výsledkem poloha. Problémem je, že výstup z *akcelerometru* obvykle obsahuje větší množství šumu. Na výstup tedy bude nutné uplatnit vhodný *post-processing*, což bude popsáno v následující kapitole.

Jelikož akcelerometr modulu *MPU-6050* pracuje v jednotkách *G-force* ($1g = 9.82m \cdot s^{-2}$), bude nutné provést převod na jednotky $m \cdot s^{-2}$. Akcelerometry pracují s různými rozsahy citlivostí - v mém případě modul snímá akceleraci v rozsahu $\pm 4 g$. V klidovém stavu by

na ose Z měla být naměřena hodnota $1\ g$, což při citlivosti modulu $4\ g$ odpovídá hodnotě 8192 . [7]

Jsou-li známy tyto informace, lze poté převést jednotky za použití vzorečku 3.5, kde x_g znázorňuje výstup jedné osy *akcelerometru* v jednotkách *G-force*, $x_{m \cdot s^{-2}}$ tento výstup převedený do jednotek $m \cdot s^{-2}$ a G gravitační konstantu ($\approx 9,82$).

$$x_{m \cdot s^{-2}} = x_g / 8192 \cdot G \quad (3.5)$$

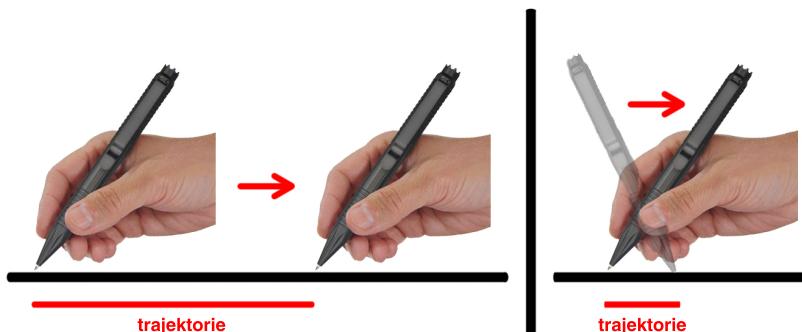
Modul poskytuje data každých $10ms$, tedy po následné integraci podle času je výstupem rychlosť ve třech osách. Ke spočtení celkové rychlosti je možné vyněchat třetí rozměr (osa Z) a spočítat celkovou rychlosť psaní v jako velikost vektoru rychlostí os X (v_x) a Y (v_y), viz vzorec 3.6.

$$v = \sqrt{v_x^2 + v_y^2} \quad (3.6)$$

Rekonstrukce trajektorie

Nejprve je nutné uvědomit si, jakým způsobem vlastně lidé píší. Teoreticky jde o dvě varianty (prakticky jde o jejich kombinaci):

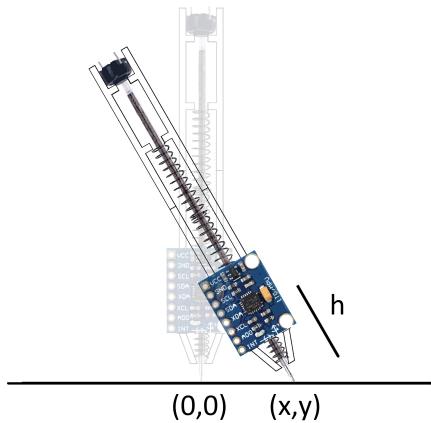
- nulová rotace propisky a změna polohy zápěstí či ruky (obrázek 3.9 vlevo),
- rotace propisky kolem své osy bez změny polohy ruky (obrázek 3.9 vpravo),
- kombinace obou variant - propisku při psaní rotujeme, zároveň však posouváme zápěstí.



Obrázek 3.9: Způsoby psaní.

Uvažujme následující situaci zjednodušenou do dvou rozměrů (popsanou obrázkem 3.10). Hrot propisky se nachází v poloze $(0, 0)$. Pokud člověk něco napiše teoreticky druhým způsobem (rotováním propisky), dostane hrot do polohy (x, y) . Jakmile je známa orientace modulu *MPU-6050* v prostoru (viz sekce 3.4) a vzdálenost modulu od hrotu h , je možné spočítat novou polohu hrotu tak, že je sestavená rotační matice vynásobená vektorem d (vzorec 3.7).

$$d = \begin{pmatrix} 0 \\ h \\ 0 \end{pmatrix} \quad (3.7)$$



Obrázek 3.10: Ukázka pohybu s propiskou.

První způsob (psaní pouze změnou polohy ruky) dokáže modelovat *akcelerometr* sám o sobě, kdy je druhou integrací výstupu podle času získána poloha. Prakticky jde téměř vždy o kombinaci obou variant - v takové situaci lze sečít spočtené vektory polohy. Tímto způsobem je teoreticky možné rekonstruovat trajektorii z nasbíraných dat, zároveň je zaznamenáván i sklon propiskou a rychlosť.

Součástí práce je možnost využití naskenovaného písma, které bylo napsáno navrženou propiskou. Tím pádem lze eliminovat šum a nepřesnosti z měření a výpočtu týkající se tvaru písma. Bude tedy třeba využít poznatky z *off-line* systémů pro rozpoznávání písma a podpisu, zpracovat naskenované či vyfocené písmo a namapovat na něj informace, které byly zrekonstruovány ze senzorů. Tato část však bude popsána až v další kapitole, jelikož při návrhu systému nebylo jasné, jak moc bude zrekonstruované písmo podobné reálnému.

3.5 Imitace

3D tiskárna, která bude imitovat písmo, operuje s *firmwarem Marlin*. Proto je nutné nastudovat si, jak s ním správně komunikovat. Taková tiskárna je ovládána pomocí tzv.

*G-kódu*⁶. Z *datasheetu* lze vidět dva využitelné způsoby, kterými ovládat motory jednotlivých os - standardní přímý pohyb (*G0*) či *B-Spline* (*G5*). Příklad jednoho přímého pohybu vypadá takto:

`G0 X<x> Y<y> Z<z> F<f>,`

kde *<x>*, *<y>* znázorňují souřadnice bodu, *<z>* ovládá přiložení či nepřiložení propisky k povrchu (informace z *mikrospínače*) a *<f>* udává tzv. *feed rate*, což je údaj, kolik milimetrů za minutu má tiskárna urazit, platící od aktuálního příkazu až do další změny *feed rate*. Po převodu rychlosti psaní *v* do jednotek $m \cdot s^{-1}$, popsaným v sekci 3.4, se již *<f>* spočítá jednoduše na základě vzorce 3.8.

$$f = v \cdot 1000 \cdot 60 \quad (3.8)$$

⁶Marlin G-Code - <http://marlinfw.org/meta/gcode/>

K imitaci bude tedy třeba vygenerovat sekvenci příkazů *G-kódu* reprezentující vlastnosti písma či podpisu. Pro ovládání 3D tiskárny byl vybrán software *Repetier-Host*⁷, který umí mimo samotného zprostředkování 3D tisku provádět sekvenci příkazů *G-kódu* uloženou v souboru.

3.6 Implementační software

Jelikož modul *MPU-6050* komunikuje s *Arduinem* skrze protokol *I₂C* [31], nabídla se možnost využít knihovny *Wire*⁸ sloužící k tomuto účelu. K modulu *MPU-6050* navíc existuje knihovna *i2cdevlib*⁹ vytvořena *Jeffem Rowbergem*, která dramaticky usnadňuje manipulaci s modulem. Programovacím jazykem pro obsluhu *Arduina* a modulu *MPU-6050* se tedy stal jazyk *C* s využitím výše zmíněných knihoven.

Pro zbytek implementace byl vybrán skriptovací jazyk *Python3*¹⁰, a to hlavně z důvodu jednoduchosti, rozšířitelnosti velkým množstvím knihoven a komunitní podporou. Mimo standardní moduly jazyka *Python3* budou využity následující knihovny:

- *NumPy*¹¹ pro práci s maticemi, poli a dalšími matematickými úkony,
- *SciPy*¹² pro práci s křivkami a pro využití některých filtrů,
- *OpenCV*¹³ pro práci s obrázky,
- *thinning*¹⁴ pro ztenčení obrázku a
- *matplotlib*¹⁵ pro vykreslení mezivýsledků.

Samotná implementace bude rozdělena do sémanticky oddělených modulů, přičemž každý modul bude mít na starost jeden z nutných úkonů pro napodobení statických a dynamických vlastností písma a podpisu. Těmito moduly jsou:

- **dp** jakožto hlavní ovládací modul zpracovávající argumenty,
- **SerialReader** pro čtení dat z propisky (*Arduina*) skrze sériový port,
- **FileParser** pro transformaci uložených dat z propisky na dynamické vlastnosti písma,
- **ImageProcessor** pro zpracování obrazu a transformaci na statické vlastnosti písma,
- **Mapper** pro mapování získaných dynamických vlastností písma na statické a
- **CodeGenerator** pro vygenerování sekvence příkazů *G-kódu* reprezentujících písmo.

⁷ Repetier-Host - <https://www.repetier.com/>

⁸ Wire library - <https://www.arduino.cc/en/Reference/Wire>

⁹ i2cdevlib library - <https://github.com/jrowberg/i2cdevlib>

¹⁰ Python3 - <https://www.python.org/download/releases/3.0/>

¹¹ NumPy - <https://www.numpy.org/>

¹² SciPy - <https://www.scipy.org/>

¹³ OpenCV - https://docs.opencv.org/3.4.3/d0/de3/tutorial_py_intro.html

¹⁴ thinning - <https://pypi.org/project/thinning>

¹⁵ matplotlib - <https://matplotlib.org/>

Kapitola 4

Realizace systému

V této kapitole je popsán průběh realizace této práce. V první fázi se jedná o sestrojení propisky pro snímání statických a dynamických vlastností písma - od tisku 3D modelu přes pájení konektorů až po vytvoření komunikace s *Arduinem*. Následuje zpracování získaných dat, rekonstrukce dynamických vlastností písma a mapování těchto vlastností na foto či sken originálu písma. Finálním krokem je poté samotné napodobení za pomocí 3D tiskárny.

4.1 Propiska

Za dobu vývoje proběhlo sestrojení tří prototypů propisky - v následujícím textu bude rozebrán postup vzniku všech verzí a vysvětleny rozdíly mezi nimi.

První prototyp

První verze prototypu sloužila k otestování principu mechanismu propisky, avšak jelikož prozatím nebyl vytvořen 3D model, byl prototyp vytvořen z torza klasické propisky, do jejíž vrchní části byl umístěn *mikrospínač* a kolem náplně vloženy pružiny. Tím pádem bylo docíleno toho, že *mikrospínač* byl sepnut, jakmile propiska začala psát.

Jako kabel využitý pro propojení součástek a *Arduina* posloužil osmidrátkový *ether-netový* kabel. Po odizolování konců byly tyto drátky připájeny k *mikrospínači* a modulu *MPU-6050* za pomocí *mikropájky Basetech ZD-99*. Z důvodu zlepšení mobility byly zakoupeny konektory *CANON*¹, díky kterým je možné propisku jednoduše odpojit a připojit k *Arduinu*.

Princip zapojení spočíval v tom, že napájení modulu *MPU-6050* ovládal *mikrospínač*, modul tedy fungoval jen tehdy, pokud byl *mikrospínač* sepnut. Toto řešení se ukázalo jako špatné, jelikož při testování jsem se dozvěděl, že modulu *MPU-6050* trvá po zapnutí několik sekund, než se zkalibruje. Ukázkou prototypu lze vidět na obrázku 4.1. Tento přístup také přinesl nemožnost určení pozice jednotlivých *stroků*, neboť při zvednutí propisky a přesunu na jiné místo nebyl modul napájen.

Hlavním problémem se hned ze začátku ukázalo být to, že jsem postrádal základní dovednosti pájení - díky tomu se stala tato část práce velmi časově náročnou.

¹Konektor CANON 15 V - <https://www.gme.cz/konektor-can-15-v>



Obrázek 4.1: Ukázka prvního prototypu propisky.

Druhý prototyp

Tento prototyp byl sestrojen v rámci projektu do předmětu *Biometrické systémy*. Hlavní změnou oproti prvnímu prototypu byla skutečnost, že již byl vytvořen 3D model torza. Samotný tisk proběhl na školní 3D tiskárně *BCN3D Sigmax R19*². Tisk zabral zhruba šest hodin, a to kvůli opakováným problémům se stabilitou modelu při tisku. *Filamentem* byl materiál *PLA*.

Proběhla také změna v zapojení - *mikrospínáč* byl již zapojen nezávisle na modulu *MPU-6050* s využitím interního *pull-up rezistoru* *Arduina* pro možnost digitálního čtení hodnot *LOW* a *HIGH* pro detekci sepnutí. Schéma zapojení lze vidět na obrázku 4.2.

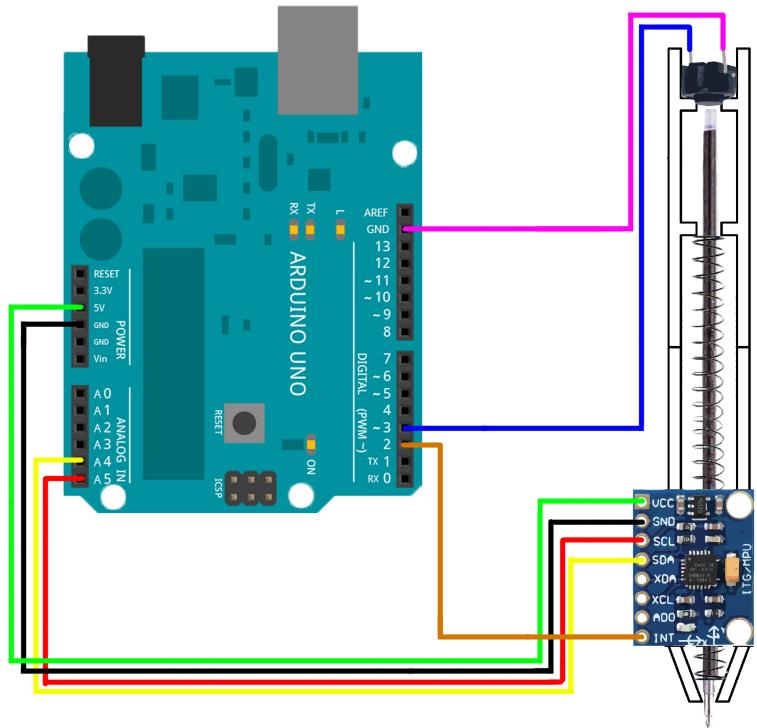
Problémem tohoto prototypu byla stálé špatná kvalita pájení a fakt, že po zafixování připájeného modulu *MPU-6050* již neexistovala možnost manipulovat s vnitřním uspořádáním propisky. Torzo sice bylo rozdelené na půlky, kabel byl však vcelku. Tento fakt znamenal nemožnost výměny náplně propisky.

Třetí prototyp

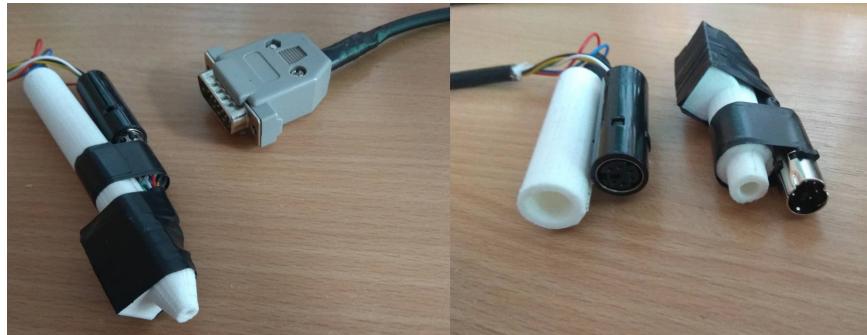
Třetí a finální prototyp byl sestrojen na základě poznatků z obhajoby projektu v předmětu *Biometrické systémy*. Hlavním důvodem vzniku bylo zkvalitnění pájení a skrytí kabelu, který sice nelze vzhledem k aktuálnímu návrhu a rozměrům skrýt dovnitř propisky, byl však zakoupen konektor *Mini-DIN*³, díky kterému je možné kabel rozdělit tak, aby se dala propiska otevřít zároveň s jeho odpojením. Konektor navíc schovává kabel po celé délce propisky. V něm byly využity *kabelové zakončovací dutinky*, jejichž využití eliminovalo nutnost pájení. Ukázkou lze vidět na obrázku 4.3.

²BCN3D Sigmax R19 - <https://www.bcn3dtechnologies.com/en/3d-printer/bcn3d-sigmax/>

³Konektor Mini-DIN - <https://www.gme.cz/konektor-mini-din-mdd6st>



Obrázek 4.2: Schéma zapojení modulu *MPU-6050* a mikrospínače do *Arduina*.



Obrázek 4.3: Pohled na třetí prototyp propiský.

Software

Po sestrojení propisky bylo nutné zaměřit se na ovládání modulu *MPU-6050* a zpracování výstupních dat. Kalibrace modulu byla provedena na základě kalibračního kódu obsaženého v knihovně *i2cdevlib*.

Princip programu běžícího na *Arduinu* tkví v obsluze přerušení modulu *MPU-6050* (modul poskytuje data každých 10 ms), kdy po příchodu přerušení, znamenající data na výstupu modulu, čte program data pomocí *I₂C* sběrnice a zároveň hľadá výstupní hodnotu z *mikrospínače*. Pokud je sepnut, začne program posílat skrze sériový port vybraná data z modulu - jmenovitě *akceleraci* a úhly *yaw*, *pitch* a *roll*, navíc také informaci o sepnutí či rozepnutí *mikrospínače*.

Jelikož člověk při psaní zvedá propisku a rozepíná tím *mikrospínač*, čímž tvoří více *stroků*, bylo nutné implementovat časovač tak, aby data z modulu proudila i mezi jednotlivými *stroky*. Časovač je resetován při dalším sepnutí *mikrospínače*. Pokud však časovač přesáhne určitou hodnotu, program ukončí odesílání dat a bere tím psaní za ukončené.

Pro ukládání dat do počítače slouží modul `SerialReader`, jenž má za úkol číst data ze sériového portu (specifikace portu udává argument skriptu) a ukládat proudící sekvenci dat, reprezentující vlastnosti písma, do souborů.

4.2 Rekonstrukce dynamických vlastností písma

Tato sekce popisuje implementaci rekonstrukce dynamických vlastností písma z dat získaných ze senzorů, která je obsahem modulu `FileParser`. Hlavním problémem je fakt, že levné *MEMS* senzory jsou velmi náchylné k chybám. [30]

Zpracování dat akcelerometru

Na základě návrhu rekonstrukce trajektorie a rychlosti psaní (popisáno v sekci 3.4) byla data z *akcelerometru* jednou zintegrována podle času, z čehož byla získána rychlosť psaní, a podruhé pro zisk polohy. *DMP* navíc umí odstranit z dat *akcelerometru* vliv gravitace Země, čímž je sám vyřešen jeden problém - je získána *lineární akcelerace*.

Velkou překážkou však byl fakt, že tato data jsou extrémně náchylná k šumu a chybám (viz obrázek 4.4 - ukázka hodnot *akcelerometru* v klidovém stavu kolem začátku psaní v osách X, Y a Z). Po první integraci se chyba promítne lineárně, po druhé už exponenciálně. Znamená to, že takto jsou data z *akcelerometru* téměř nepoužitelná, i když byl řádně zkaliarován. V situaci, kdy je v datech přítomen šum, je možné standardně postupovat aplikací určitých filtrů jako dolní propust (viz sekce 2.8). Pokud je však chyba velmi výrazná, tyto filtry nepřináší žádnou přidanou hodnotu.

H	-19	-513	208	2.5023	0.3305	0.0506
H	-92	-549	239	2.5059	0.3262	0.0437
H	-154	-600	200	2.5076	0.3220	0.0352
H	-99	-507	269	2.5087	0.3181	0.0265
H	-68	-277	225	2.5104	0.3139	0.0178
H	-55	-161	273	2.5125	0.3103	0.0103
H	-91	-160	239	2.5140	0.3069	0.0030
L	-106	-49	274	2.5139	0.3038	-0.0058
L	-21	83	383	2.5131	0.3010	-0.0142
L	-138	-9	283	2.5140	0.2977	-0.0219
L	-214	-77	303	2.5136	0.2949	-0.0300
L	-193	-148	365	2.5109	0.2930	-0.0374
L	-305	-142	293	2.5056	0.2922	-0.0446
L	-199	-220	326	2.5008	0.2913	-0.0513
L	-160	-215	350	2.4931	0.2921	-0.0567
L	-150	-165	339	2.4874	0.2929	-0.0602
L	-125	-190	378	2.4847	0.2928	-0.0614
L	-50	-210	402	2.4840	0.2920	-0.0602

Obrázek 4.4: Pohled na šum *akcelerometru* při sepnutí spínače propisky.

Dalším potenciálním řešením by mohla být *fúze senzorů*, kdy se data z jednoho senzoru využijí pro korekci dat druhého senzoru, což je v případě využití *akcelerometru* a *gyroskopu*

teoreticky možné (viz sekce 2.8). Avšak po implementaci *complementary filteru* byla data stále nepoužitelná.

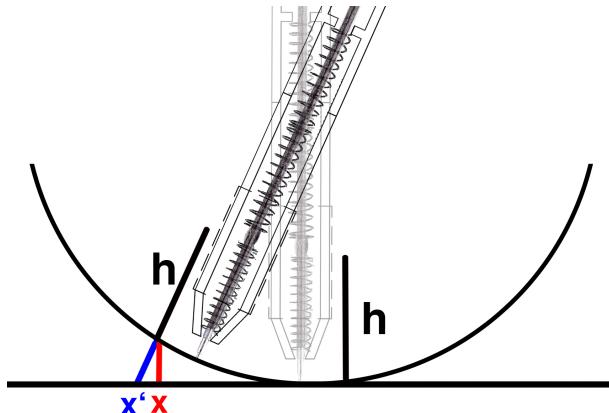
Nejlepším řešením se ukázala být manuální implementace odstranění *lineárního trendu* (viz sekce 2.8). V takovém případě je nutno zaznamenat několik hodnot z *akcelerometru* před začátkem psaní (v klidové poloze) a následně tyto hodnoty zintegrovat podle času. Tím je získán *lineární trend* chyby, která se vyskytuje již v klidovém stavu. Tento trend je následně proložen přímkou, jejíž hodnoty jsou odečteny z první integrace hodnot *akcelerometru* získaných při psaní.

Tento krok značně zpřesnil data z *akcelerometru*, avšak po druhé integraci podle času byla poloha stále nepřesná. Bohužel tento stav se mi již nepodařilo nijak vylepšit. Tímto postupem byla získána přibližná rychlosť psaní a také částečná přibližná trajektorie písma.

Zpracování dat gyroskopu

Ke zpracování dat z *gyroskopu* je třeba sestrojit při každém čtení úhlů *yaw*, *pitch* a *roll* rotační matici (popsáno v sekci 3.4), která udává orientaci modulu (a tedy i propisky) v prostoru. Pokud je tato rotační matice vynásobena vektorem udávajícím vzdálenost hrotu propisky od modulu, je získán bod v prostoru (jeden bod trajektorie).

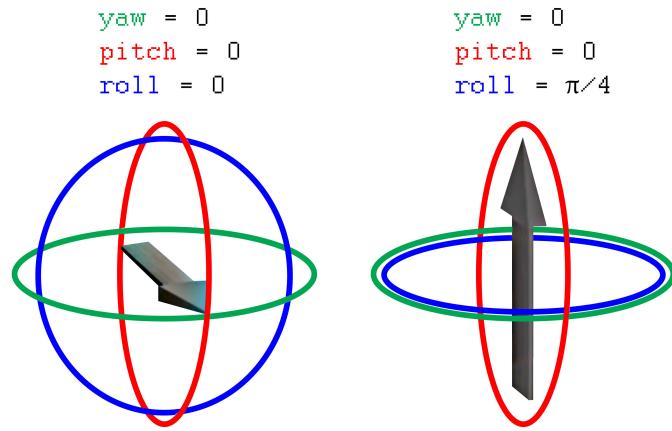
Problémem však je, že takto získaný bod v prostoru je promítán na kouli, jak ukazuje obrázek 4.5. Důvodem je způsob úchytu propisky a práce s ní - čím více je propiska skloněna, tím více se její těžiště, tedy i modul *MPU-6050*, posouvá směrem dolů, jinak by se hrot nedotýkal povrchu. S tímto však teoretický model nepočítá - pokud by tato situace zůstala neošetřena, vypočteným bodem by byl x , nikoliv x' .



Obrázek 4.5: Promítání spočtených bodů na kouli.

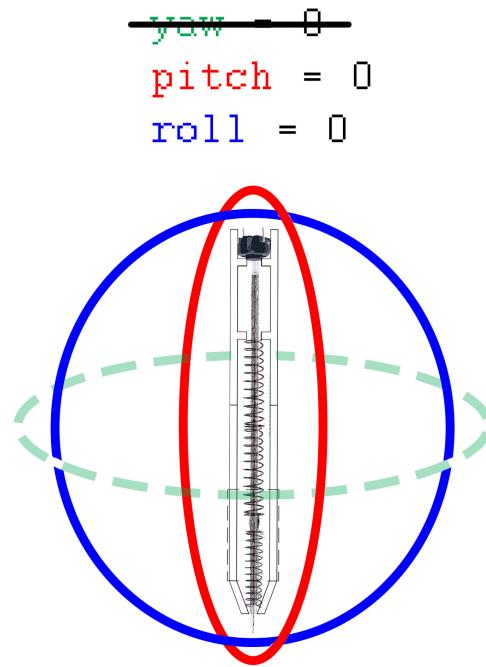
Tato otázka byla vyřešena tak, že čím více je modul nakloněn, tím více je ve výpočtu zvětšována vzdálenost hrotu propisky od modulu tak, aby byl spočten bod x' . Proměnná tohoto výpočtu je parametrizována, tedy při spuštění skriptu je možno zadat, jak moc bude vzdálenost h zvětšována s nakloněním propisky. Pevnou hodnotu nelze nastavit, jelikož se mění s úchytem propisky v ruce.

Další překážkou se ukázal být fakt, že modul *MPU-6050* je u určen k využívání v horizontální poloze, avšak v mé návrhu se počítá s vertikálním umístěním. Vertikální umístění totiž zapříčiňuje projev tzv. *gimbal locku*. Ten nastává, pokud je modul otočen v jedné z os o 90 stupňů (viz obrázek 4.6), kdy dojde ke ztrátě jedné dimenze. [20]



Obrázek 4.6: Gimbal lock.

Tato situace byla ošetřena úpravou kódu v knihovně *i2cdevlib* tak, aby výstupní hodnoty senzoru udávaly nulovou rotaci v momentě, kdy je modul otočen v jedné z os o 90 stupňů, což pramení v dojem, že výchozí poloha modulu *MPU-6050* je vertikální. Samotný *gimbal lock* po této úpravě může být zanedbán tím, že sklon propisky bude počítán pouze ze dvou dimenzí. Tím se předpokládá, že se propiska nebude rotovat kolem osy *yaw*, jelikož tento úhel při psaní nedělá žádný rozdíl. Výsledné nastavení modulu (a tedy i propisk) v prostoru popisuje obrázek 4.7.



Obrázek 4.7: Zaznamenávání rotace propisky bez jedné dimenze.

Stejně jako *akcelerometr*, i *gyroskop* trpí šumem, avšak zde již z principu nelze uplatnit nic jako odstranění *lineárního trendu*. Dostatečným se ukázal být *median filter*, který vyhlažuje hodnoty vstupu dle zadané velikosti okna (viz sekce 2.8). Dle výše uvedených

matematických postupů je tak spočten jak sklon propisky při psaní, tak bod, kde se hrot propisky dotýká povrchu, tedy částečná trajektorie.

Post-processing

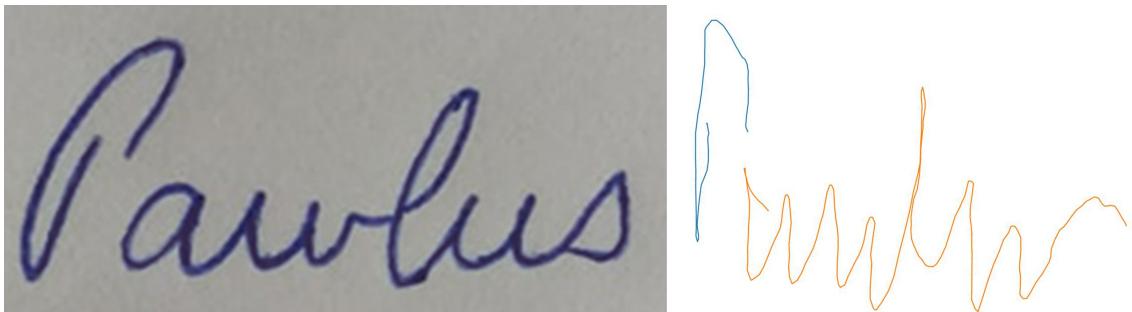
Po zpracování dat ze senzorů jsou známy tyto věci:

- přibližná rychlosť psaní (*akcelerometr*),
- částečná trajektorie zapříčiněná pohybem ruky po povrchu (*akcelerometr*),
- sklon propisky (*gyroskop*),
- částečná trajektorie zapříčiněná rotací propisky v ruce (*gyroskop*),
- informace o tom, kdy se hrot dotýká povrchu (*mikrospínač*).

Z těchto informací lze sečtením obou částečných trajektorií sestavit kompletní trajektorii a tu poté rozdělit do *stroků* díky informaci z *mikrospínače*. Následně byly implementovány transformační funkce, které určitým způsobem upravují spočtené body trajektorie, z nichž nejvýznamnější je rotace (viz sekce 2.6).

Další možností je například ořez bodů zprava a zleva. Všechny tyto funkce jsou parametřitelné, aplikují se tedy na základě zadaných parametrů při spuštění skriptu. Jeho výstupem je pak grafická reprezentace spočtené trajektorie, což znamená, že uživatel může parametry dynamicky upravovat podle dosaženého výsledku.

Jako příkladem budiž můj zjednodušený podpis a jeho zrekonstruovaná trajektorie na obrázku 4.8.



Obrázek 4.8: Ukázka mého podpisu a jeho zrekonstruované trajektorie.

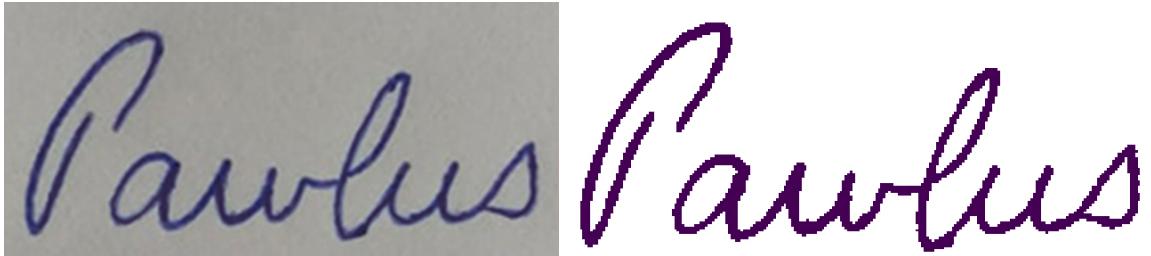
Díky nepřesnosti a šumu senzorů nebylo možné ze samotné propisky dostat lepší data, bylo tedy nutné trajektorii výrazně zpřesnit za pomoci obrázku originálu písma.

4.3 Zpracování originálního písma

Ke zpřesnění zrekonstruované trajektorie z dat senzorů je nutné patřičně zpracovat obrázek s originálním písmem. Tento postup je obsahem modulu `ImageProcessor`.

Jako první je na obrázek aplikováno *prahování* tak, aby byly jeho obsahem pouze pixely černé a bílé. K tomu je využita funkce `threshold` z knihovny `opencv`. Následně je využita funkce `argwhere` z knihovny `NumPy` pro detekci černých bodů. Na základě této informace

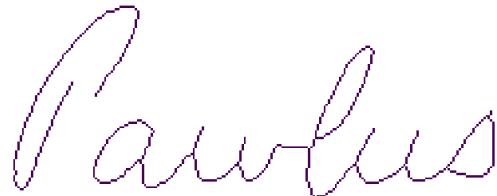
je pak obrázek ořezán funkcí `crop` z knihovny *OpenCV*. Výsledkem je černobílé ořezané písmo (obrázek 4.9).



Obrázek 4.9: Obrázek originálního písma a jeho zpracování *prahováním* a *ořezem*.

K tomu, aby bylo možné obrázek využít pro získání tvaru písma, je potřeba aplikovat *ztenčení* tak, aby bylo písmo reprezentováno pouze jedinou linií se šířkou jednoho pixelu. Jako použitelné varianty byly zváženy dva algoritmy - *Zhang-Suen* a *Guo-Hall* (více v sekci 2.6). Vzhledem k malé velikosti obrázků bylo možné zanedbat výběr na základě rychlosti těchto algoritmů a soustředit se pouze na výsledky, kde druhý jmenovaný dosahoval v mém případě lepších výsledků a je navíc již implementován jako knihovna *thinning*.

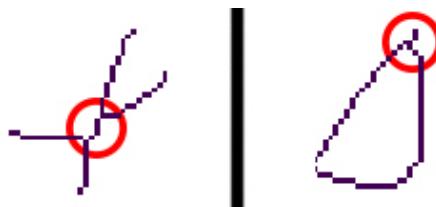
Výsledek *ztenčení* za použití algoritmu *Guo-Hall* lze vidět na obrázku 4.10.



Obrázek 4.10: Písma po aplikaci *ztenčení*.

Ztenčení však přináší také komplikace v podobě nepřesnosti kvůli použitému algoritmu. První se dotýká oblastí, kde se určitým způsobem kříží pixely - například v psacím písmenu *l* se po aplikaci *ztenčení* vytvoří mezera (obrázek 4.11 vlevo), která zapříčinuje dojem, že písmo zde na sebe plynule nenavazovalo. Tento problém by šel teoreticky vyřešit implementací detekce takového křížení, odstraněním mezery a posunutím zbytku navazujících pixelů, avšak na toto řešení již bohužel nebyl prostor.

Dalším problémem jsou přebytečné pixely (obrázek 4.11 vpravo). Vzhledem k tomu, že jsou obrázky uživateli prezentovány i s informací o souřadnicích jednotlivých pixelů, nabídlo se jednoduché řešení, a to zadání pixelů, které mají být z obrázku umazány, jako další parametr skriptu.



Obrázek 4.11: Obrázek po aplikaci *ztenčení*.

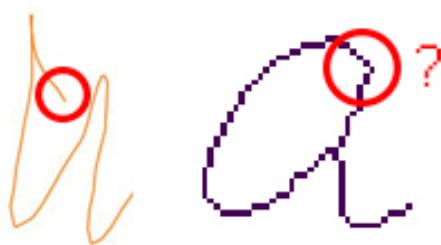
4.4 Mapování

Poté, co je k dispozici jak zrekonstruovaná trajektorie, tak tvar ze zpracovaného obrázku originálu písma, je třeba tyto dvě informace spojit tak, aby bylo možné výrazně zpřesnit tvar zrekonstruované trajektorie. Ta se výrazně liší od originálu písma (viz obrázek 4.8). Zatímco u zrekonstruované trajektorie je k dispozici časový údaj, tedy pořadí, kterými body propiska procházela, obrázek poskytuje pouze tvar bez časové informace. Obsahem této sekce je snaha najít v obrázku takovou cestu, která odpovídá zrekonstruované trajektorii, což pramení v zisk přesné trajektorie písma. Tento proces je obsahem modulu **Mapper**.

Jako první byly vyzkoušeny různé formy *feature matchingu* [27], které jsou implementovány v knihovně *OpenCV*. Výsledky však vůbec nekorespondovaly s realitou a nalezené odpovídající si body si ve skutečnosti vůbec neodpovídaly. Nakonec byl navržen a implementován algoritmus, který po zjištění počátečních a koncových bodů v obrázku prochází sousedící pixel po pixelu až do koncového bodu, a v případě, že je po cestě na výběr více pixelů, využije se pro rozhodování informace ze zrekonstruované trajektorie.

Určení počátečních a koncových bodů

Počáteční a koncové body jednotlivých *strok* jsou známy ze zrekonstruované trajektorie. Nabízí se tedy automatizované řešení, a to vyhledání počátečních a koncových pixelů v obrázku, které mají pouze jednoho souseda v oblasti zhruba odpovídající výskytu počátečního, resp. koncového bodu *stroku*. Tento přístup by však nefungoval v případě takového písma, kde počátečním, resp. koncovým bodem, prochází i další úsek písma. Příkladem může být psací písmeno *a* - ve zrekonstruované trajektorii je jasné vidět, kde počáteční bod je (obrázek 4.12 vlevo), avšak ve zpracovaném obrázku již nikoliv - takový počáteční bod má totiž více než jednoho souseda (obrázek 4.12 vpravo).



Obrázek 4.12: Problém s detekcí počátečního bodu.

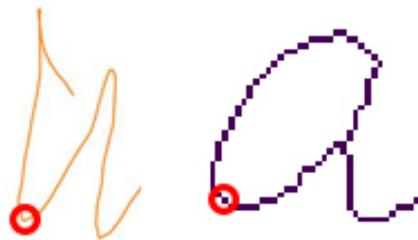
Jak bylo již zmíněno při odstraňování přebytečných pixelů obrázku, uživatel při práci se skriptem vidí souřadnice jednotlivých pixelů. Z tohoto důvodu se tedy opět nabídlo manuální řešení, a to zadání počátečních a koncových bodů jednotlivých *strok* jako parametrů skriptu při spuštění.

Hledání cesty

Jakmile je znám počáteční a koncový bod *stroku* v obrázku, je implementován algoritmus, který prochází sousedící pixely. K jejich nalezení je využita třída **KDTree**⁴ z knihovny *OpenCV*, která dokáže poměrně rychle hledat sousedící pixely, jelikož souřadnice pixelů

⁴ *KDTree* - https://docs.opencv.org/ref/2.4/df/d23/classcv_1_1KDTree.html

převede do stromové struktury. Při přechodu na každý další pixel je inkrementována proměnná udávající počet pixelů, které již byly navštívěny (zároveň jsou ukládány do seznamu, který v konečném důsledku obsahuje zpřesněnou trajektorii). Tento index slouží k přístupu do pole bodů zrekonstruované trajektorie tak, aby si proporcionálně odpovídaly (tak, jak je znázorněno na obrázku 4.13).



Obrázek 4.13: Procházení pixelů obrázku zároveň s body zrekonstruované trajektorie.

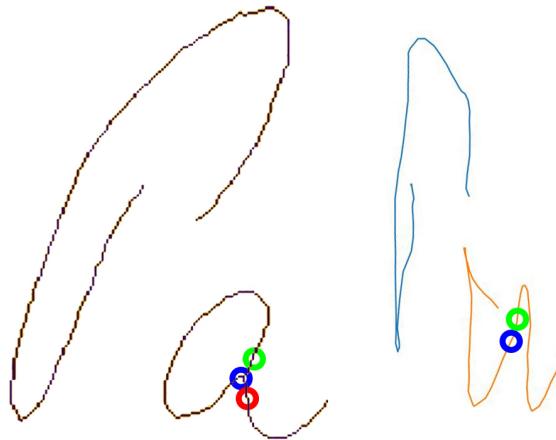
Problémem však je, že počet bodů zrekonstruované trajektorie je jiný než počet pixelů na obrázku, tedy index bodu zrekonstruované trajektorie nebude proporcionálně odpovídat aktuálně procházenému pixelu na obrázku. Řešením se ukázala být *interpolace* [9] bodů zrekonstruované trajektorie *kubickým B-Splinem* [14] za pomocí funkce `splprep`⁵ z knihovny *SciPy*. Z takové křivky lze následně ohodnotit kolik bodů tak, aby počet seděl s počtem bodů obrázku (počet ohodnocených bodů musí být o něco vyšší než počet pixelů obrázku, neboť se musí počítat s tím, že některé pixely jsou procházeny vícekrát, viz psací a).

Následně může začít procházení od počátečního bodu. Algoritmus určuje cestu následujícím způsobem:

- pokud má aktuální pixel jen jediného souseda, který ještě nebyl navštíven, je vybrán tento pixel,
- pokud má aktuální pixel více sousedů, kteří dosud nebyli navštíveni, je spočtena *směrnice* cesty vedoucí od každého z tohoto pixelů a porovnána se směrnicí odpovídajících dvou bodů zrekonstruované trajektorie,
- pokud má aktuální pixel kolem sebe pouze pixely, které již byly procházeny, rozhoduje se opět na základě směrnice odpovídajících bodů zrekonstruované trajektorie.

Princip rozhodování na základě směrnic je ukázán na obrázku 4.14. Pokud není jasné, kterou cestu (zelené a červené kolečko) má prohledávací algoritmus vybrat z aktuálního pixelu (modré kolečko), jsou analyzovány obě cesty, které přichází v úvahu. Z obou variant je nalezeno následujících 10 pixelů, z nichž je spočtena přibližná směrnice cesty, která je porovnána se směrnicí odpovídajících bodů zrekonstruované trajektorie. Algoritmus se následně rozhoduje pro cestu na základě podobnosti směrnic (na obrázku zelený bod). Prohledávání je ukončeno, pokud byl nalezen koncový bod a zároveň pokud počet bodů, které byly navštívěny, odpovídá počtu bodů zrekonstruované trajektorie (ohodnocených z křivky).

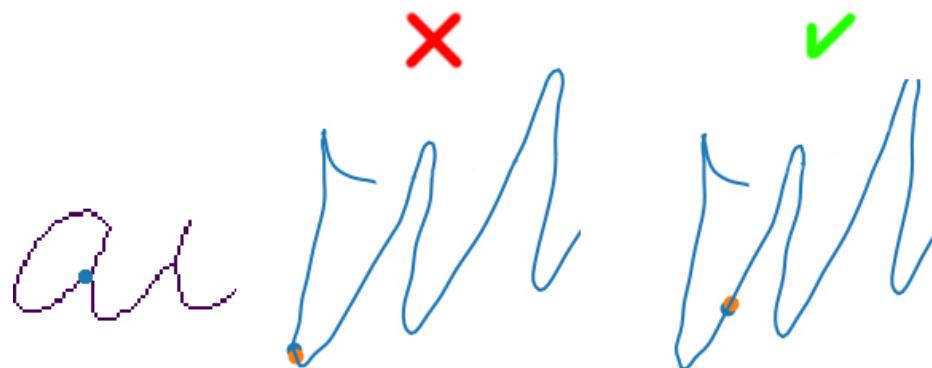
⁵SciPy B-Spline interpolation - <https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.interpolate.splprep.html>



Obrázek 4.14: Rozhodování se v cestě na základě směrnic.

Optimalizace pro delší *stroky*

Tento algoritmus funguje pro jednodušší vstupní písmo, kde jsou jednotlivé *stroky* kratší, avšak kupříkladu na můj zjednodušený podpis, použitý pro ukázkou, nefunguje dostatečně dobře. Je to dáno tím, že body obrázku nemusí, a často ani nesedí, proporcionálně s body zrekonstruovaného písma, i když je k dispozici stejný počet bodů (toho bylo dosaženo interpolací křivkou a ohodnocením správného počtu bodů), jak ukazuje obrázek 4.15. V prvním případě je díky neodpovídajícím si proporcím vybrána špatná cesta v obrázku (směrem dolů), zatímco ve druhém správná (směrem nahoru).



Obrázek 4.15: Ukázka proporčně neodpovídajícího a odpovídajícího zrekonstruovaného písma.

Problém byl eliminován zavedením prvku náhodnosti do ohodnocení bodů křivky. V algoritmu popsáném výše byly body křivky ohodnoceny rovnoměrně, avšak pokud s takovými body není nalezen koncový bod na obrázku, dojde k znovuohodnocení bodů na křivce, ale s různou hustotou bodů v různých částech křivky (tyto parametry jsou náhodně vygenerovány) tak, jak lze vidět na obrázku 4.16.

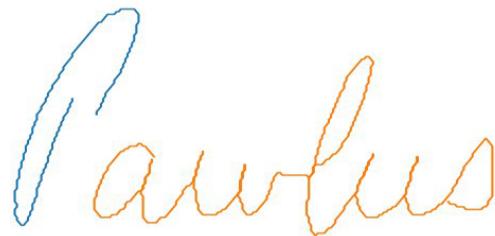
Následně proběhne další pokus o vyhledání cesty v obrázku. Tento postup může zaříčinit například situaci v obrázku 4.15, kdy poprvé mapování selže, podruhé však může vyjít. Znovuohodnocení je prováděno v nekonečné smyčce, dokud není mapování úspěšné.

Při úspěšném mapování skript vypíše informaci o hustotě bodů v různých částech křivky tak, aby při dalším spuštění bylo možné zopakovat mapování se správnou hustotou bodů.



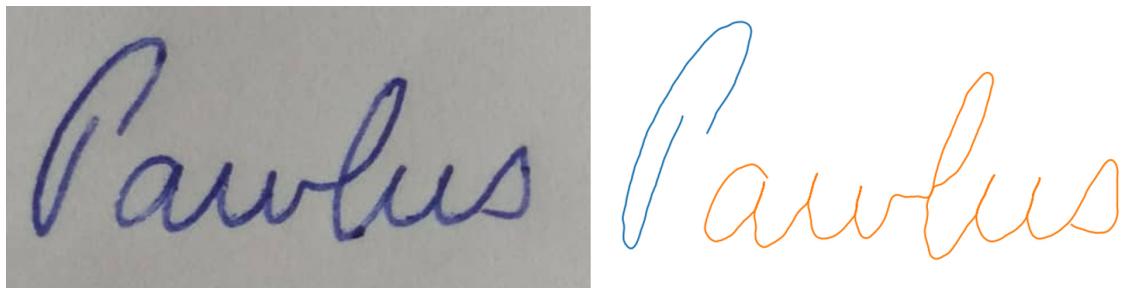
Obrázek 4.16: Rozdíly v hustotě bodů v jednotlivých částech zrekonstruované trajektorie.

Po nalezení trajektorie v obrázku je nutné vyřešit rozměry písma tak, aby imitace odpovídala i velikostí. To nelze provést jinak než zadat skriptu při spuštění další parametry, kterými jsou výška a šířka originálního písma. Jakmile je k dispozici tato informace, je možné body trajektorie transformovat do těchto rozměrů. Zrekonstruovaná trajektorie poté vypadá tak, jak ukazuje obrázek 4.17.



Obrázek 4.17: Ukázka úspěšně namapované zrekonstruované trajektorie na obrázek originálnu písma.

Jelikož souřadnice pixelů pracují pouze s celými čísly, působí trajektorie neplynule a kostrbatě. Elegantním řešením se ukázala být aplikace *Savitzky-Golay* filtru (viz sekce 2.8), jež dokáže efektivně vyhladit trajektorii. Porovnání originálního písma a výsledné zrekonstruované trajektorie je k vidění na obrázku 4.18.



Obrázek 4.18: Porovnání originálního písma a výsledné zrekonstruované trajektorie.

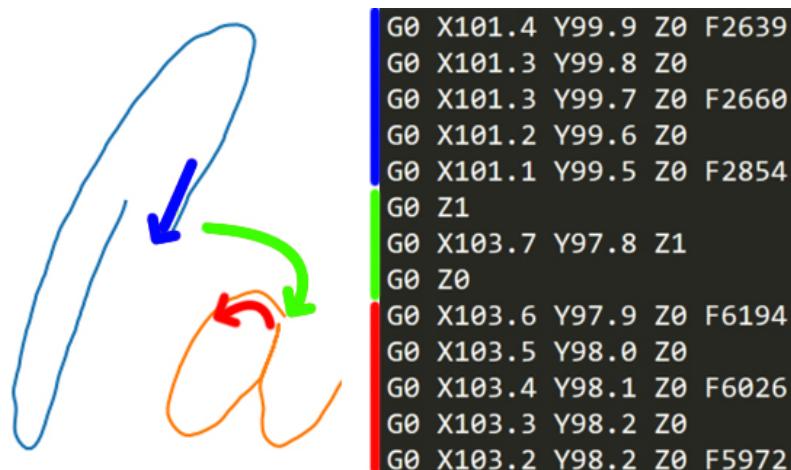
Na závěr je trajektorie posunuta do počátku souřadnicového systému, tedy aby počáteční bod prvního stroku odpovídal souřadnici $(0, 0)$.

4.5 Generování příkazů G-kódu

S ohledem na jednoduchost byl vybrán způsob ovládání 3D tiskárny za pomocí přímých pohybů motorů jednotlivých os (příkaz **G0**). Ovládání za pomocí *B-Spline* křivek (příkaz **G5**) by nepřineslo žádné výhody, protože ve výsledku se ukázalo, že finální trajektorie písma obsahuje dostatečné množství bodů na to, aby je tiskárna procházela s nejmenším možným krokem ($0,1\text{ mm}$) - v případě mého podpisu bylo bodů k dispozici ještě více. Generování příkazů *G-kódů* je předmětem modulu **CodeGenerator**.

Vstupem do této části je již přesná trajektorie písma rozdělená na jednotlivé *stroky* a odpovídající přibližná rychlosť psaní. Souřadnice bodů trajektorie navíc odpovídá rozměrům originálu písma v milimetrech. Generování kódu tedy probíhá tak, že pro každý bod je vygenerován jeden **G0** příkaz obsahující souřadnice tohoto bodu a rychlosť v milimetrech za minutu. Protože může být bodů více, než dokáže tiskárna imitovat, jsou ignorovány body, jejichž souřadnice nejsou posunuty alespoň o celého $0,1\text{ mm}$ některým směrem od předchozího bodu (to samé platí o rychlosti).

V prvopočátku je nutné odblokovat motory 3D tiskárny (příkaz **M17**). Před začátkem každého *stroku* je motor osy Z posunut o 1 mm vzhůru (zvednutí propisky), následně motory os X a Y na počáteční bod *stroku* a poté osa Z opět o 1 mm dolů (přiložení propisky). Tento postup je možné pozorovat na obrázku 4.19.



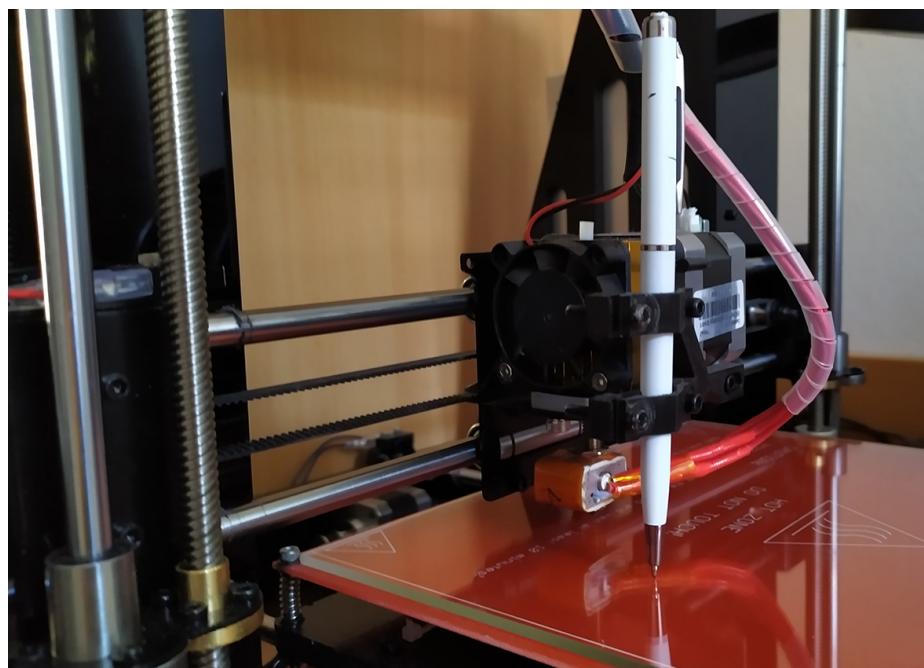
Obrázek 4.19: Princip generování příkazů *G-kódů*.

4.6 Imitace

Aby byla tiskárna schopna správného fungování, bylo třeba ji nejprve řádně zkalibrovat. Ze začátku totiž nereagovala na sepnutí tzv. *end-stopů*, což jsou spínače (umístěné na kraji každé osy), jejichž sepnutí znamená zastavení motoru dané osy. Tento princip je využíván pro posunutí motorů všech os do počátku souřadnicového systému 3D tiskárny. Dalším problémem byl nezkoordinovaný pohyb osy Z.

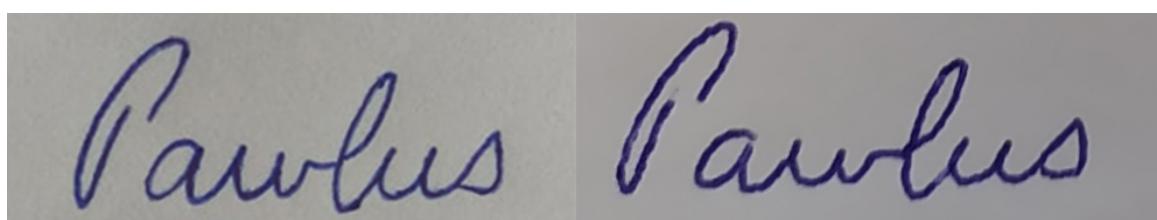
Pro vyřešení obou problémů bylo třeba přepsat firmware tiskárny - motorům os X a Y bylo nutné nastavit invertovaný pohyb, motor osy Z měl pro změnu nastavenou hodnotu *kroků/mm* pro úplně jiný typ motoru. Jelikož 3D tiskárnu ovládá modul na bázi *Arduino Mega 2560*⁶, bylo nutné najít správné konfigurační soubory, vhodně je přepsat a nahradit zpět skrze *Arduino IDE*. Poté již 3D tiskárna začala fungovat správně.

Pro samotnou imitaci je nutné upevnit propisku do speciálního držáku tak, aby se při nastavení motoru osy Z do souřadnice 0 dostatečně dotýkala povrchu, na kterém je upevněn papír (obrázek 4.20). Tímto nastavením lze také částečně ovládat přítlak psaní, který však v této práci není zohledněn.



Obrázek 4.20: Pohled na 3D tiskárnu se speciálním držákem na propisku.

Poté je již jen vybrán soubor s vygenerovanými příkazy *G-kódu* a vykonán za pomocí programu *Repetier-Host*. Finální porovnání originálního a napodobeného písma lze vidět na obrázku 4.21.



Obrázek 4.21: Porovnání originálního písma a výsledné zrekonstruované trajektorie.

⁶Arduino Mega 2560 - <https://store.arduino.cc/mega-2560-r3>

Kapitola 5

Dosažené výsledky

V této kapitole bude prezentována ukázka využití systému, následovaná rozborem zpětné vazby od znalce ručně psaného písma.

5.1 Ukázka použití

Skript, který byl vyvinut, pracuje na *inkrementální* bázi. Znamená to, že je třeba po každém spuštění kontrolovat výstup, měnit parametry dle potřeby a při správném mezivýsledku přidávat další parametry až do doby, kdy je vygenerován soubor s příkazy *G-kódu* pro 3D tiskárnu. V prvopočátku je nutné nasbírat data o písmu v reálném čase:

```
python3 dp.py -s ttyS4 pawlus -t 20,
```

kde parametr **-s** specifikuje sériový port, ke kterému je připojeno *Arduino* s propiskou, a název souboru, do kterého budou data uložena. Parametr **t** udává počet hodnot, které mají být zaznamenány za účelem odstranění *lineárního trendu akcelerometru*. Po samotném aktu psaní lze posléze výstupní soubor začít analyzovat a rekonstruovat dynamické vlastnosti písma:

```
python3 dp.py -p pawlus -w 1 -g 1 1,
```

kde parametr **-p** specifikuje vstupní soubor s daty ze senzorů, **-w** udává váhu při odstraňování *lineárního trendu* a **-g** udává hodnoty pro ošetření promítání bodů z *gyroskopu* na kouli (více v sekci 4.1). Skript následně vykreslí obrázek se zrekonstruovanou trajektorií, na kterou je následně možné aplikovat některé úpravy:

```
python3 dp.py -p pawlus -w 1 -g 1 1 -r 20 -c 5 10,
```

kde **-r** aplikuje transformační matici rotace na spočtené body (hodnota udává stupně) a **-c** procentuální useknutí bodů zleva a zprava (více v sekci 4.2). Výsledkem je opět vykreslená trajektorie dle zadaných parametrů. Pokud je již výstup v pořádku, může začít zpracování originálního písma:

```
python3 dp.py -p pawlus -w 1 -g 1 1 -r 20 -c 5 10 -i pawlus.jpg  
18 10 145 120,
```

kde **-i** očekává název obrazového souboru, šířku a výšku originálního písma v milimetrech a hodnoty pro aplikaci *prahování* na obrázek. Výstupem je poté zpracovaný obrázek. Pokud

byly hodnoty pro *prahování* vybrány správně, uživatel vidí úspěšně ztenčenou linii písma. Následně si na základě spočtené trajektorie a ztenčeného obrázku zapíše počáteční a koncové body jednotlivých *stroků* a zároveň body, které je vhodné z obrázku odstranit (více v sekci 4.3). Následně může být započato mapování:

```
python3 dp.py -p pawlus -w 1 -g 1 1 -r 20 -c 5 10 -i pawlus.jpg  
18 10 145 120 -m 21 45 89 12 -d 45 53
```

kde **-m** očekává seznam počátečních a koncových bodů jednotlivých *stroků* a **-d** seznam pixelů, které mají být z obrázku odstraněny. Po úspěšném mapování je výstupem výsledná trajektorie a také seznam nalezených parametrů mapování, aby je mohl uživatel ručně zadat do kódu při dalším spuštění (více v sekci 4.4). Zároveň se vygeneruje soubor s *G-Code* příkazy pro 3D tiskárnu, který může být spuštěn programem *Repetier-Host* (více v sekci 4.5).

5.2 Zpětná vazba a potenciální vylepšení

Jedním z výstupů této práce je papír obsahující několik dvojic originál - imitace písma (viz příloha A), který byl konzultován se znalcem z oboru ručně psaného písma. Výsledná zpětná vazba zní:

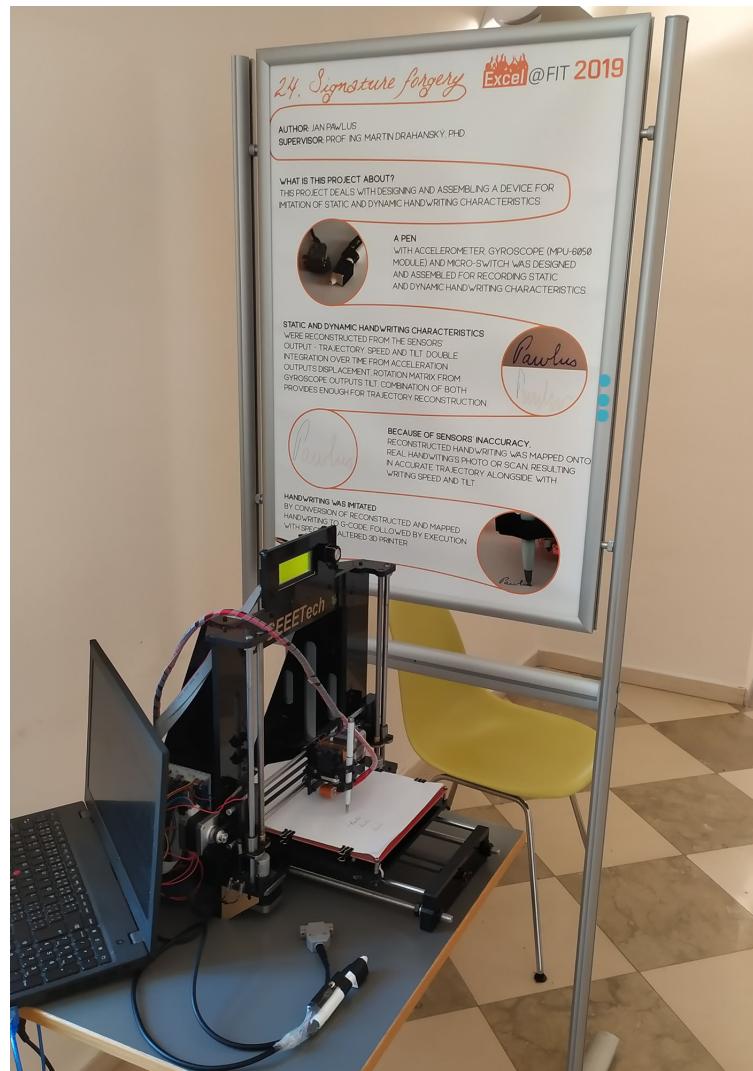
„Na první pohled jsou znatelné rozdíly, roztřepaný tah písma (tiskárna). Písmo (originál) je spíše malované (pomalu psané), což je pro tiskárny relativně jednoduché napodobit. Pokud píšete plynule a rychle, je vše mnohem složitější. Tiskárna neřeší “zeovou” současnosti, tj. přítlač, vlásečnicové napojování apod. Nicméně jsme přesvědčeni, že díky současným technologiím bude v brzké době falzum k nerozeznání od originálu.“

Tyto problémy ukazují na několik věcí, z nichž nejzásadnější jsou dvě - nedostatek času a nedostatečně kvalitní součástky. V následujícím textu budou rozebrány možná řešení těchto problémů.

- **roztřepaný tah písma** - tento problém je způsoben výběrem 3D tiskárny jakožto zařízení pro napodobení. Pokud by bylo obstaráno vhodnější zařízení s pevnějším úchytem, tento problém by byl odstraněn - motory 3D tiskárny nejsou navrženy k podobné zátěži. Navíc by bylo vhodné vybrat takové zařízení, které je schopno imitovat také sklon psaní (který je již nyní k dispozici).
- **pomalu psané písmo** - vzorky byly napsány pomaleji z důvodu nedostatečné kvality senzorů. Při velkých rychlostech psaní již modul *MPU-6050* není schopen dodávat data, na která by bylo možné namapovat originál písma. Potenciálním řešením je jednoznačně využití kvalitnějších senzorů (ne *MEMS*), případně využití více senzorů tak, aby bylo možné výstup alespoň průměrovat a redukovat tímto způsobem chyby. Bylo by dokonce možné vydat se úplně jinou cestou a vzít si příklad z existujících řešení, rozebraných v sekci 2.1, a využít například *infračervenou kameru* nebo pole *magnetometrů* pro zaznamenávání vlastností písma.
- **přítlač** - zde se nabízí využití *tlakoměru* místo *mikrospínače*. Ten umí plnit stejnou funkci jako *mikrospínač*, tedy detekci psaní, zároveň ale poskytuje informaci o přítlaku. *Tlakoměr* by bylo možné zasadit i do aktuálního návrhu mechanismu namísto *mikrospínače* a samotný přítlač by bylo částečně možné simulovat i na 3D tiskárně (pohyb o 0,1 mm vzhůru či dolů motorem osy Z).

Ze závěru však na druhou stranu vyplývá, že projekt má potenciál a při odstranění výše uvedených neduh se mohou falzifikáty stát poměrně kvalitními. Pokud by se navíc propiska vytvořila tak, že by byly senzory zabudovány přímo v torzu, nemusel by člověk ani poznat, že píše s nebezpečným nástrojem.

Projekt byl prezentován formou plakátu na konferenci *Excel@FIT 2019*¹ (projekt číslo 24). Ocenění sice nezískal, avšak byl zajímavým například pro PR oddělení *Vysokého učení technického*, kdy byl domluven rozhovor o práci na portál *zVUT.cz*². Zároveň vyšel dne 16. 5. 2019 článek o práci na titulní straně v *Deníku Právo*³. Dá se tedy říct, že projekt navíc úspěšně reprezentuje *Vysoké učení technické*, resp. *Fakultu informačních technologií*. V rámci konference bylo možné vidět živou demonstraci imitace za pomoci 3D tiskárny a také sestrojenou propisku (obrázek 5.1).



Obrázek 5.1: Ukázka stanoviště na konferenci *Excel@FIT 2019*.

¹Excel@FIT 2019 - <http://excel.fit.vutbr.cz/>

²zVUT.cz - <http://zvut.cz/>

³Deník Právo - <https://www.pravo.cz/>

Kapitola 6

Závěr

Cílem této práce bylo zpracování tématiky týkající se rozpoznávání písma či podpisu, navržení systému pro napodobení písma či podpisu, realizace takového systému a konzultace výsledků se znalci v oboru psaného textu.

V počátcích byly diskutovány současné trendy v odvětví rozpoznávání písma a podpisu - byly popsány různé typy systémů pro jejich rozpoznávání z pohledu principu funkčnosti a potřebného hardwaru, často využívané *klasifikační* (písmo) a *verifikační* (podpis) metody a podepisovací přístroje. Následoval teoretický rozbor dalších podkladů využívaných v této práci.

Následně byl představen návrh systému pro napodobení vlastností písma či podpisu. Tento proces zahrnoval ujasnění si požadavků na systém, výběr jednotlivých komponent, vhodného nástroje pro napodobování a senzoriky pro snímání vlastností písma či podpisu. Na základě tohoto výběru byl představen model propisky, jenž snímá tyto vlastnosti, mechanismus, kterým propiska pracuje, a poté samotná realizace této propisky.

Dále byl navržen teoretický model zpracování výstupních dat propisky, pramenící v zisk kýžených dynamických (sklon, rychlosť) vlastností písma a podpisu. Realizace tohoto modelu s sebou přinesla také potvrzení toho, že výstup *MEMS* senzorů není pro tento účel vhodný kvůli náchylnosti na chyby. To s sebou přineslo nutnost implementace odstranění *lineárního trendu* a jiných vhodných transformací získaných dat, aby bylo vůbec možné zrekonstruovanou trajektorii zpřesnit originálem písma ve formě obrázku. Samotné mapování pak znamenalo nutnost vyvinutí speciálního algoritmu, jenž využívá k úspěšnému dokončení prvky náhodnosti a práci s *B-Spline* křivkami.

Nakonec byly všechny získané vlastnosti písma či podpisu, které je možné napodobit na 3D tiskárně, převedeny do sekvence příkazů *G-kódu* a napodobeny na této 3D tiskárně. Výsledky napodobení byly konzultovány se znalci v oboru psaného písma a na základě těchto poznatků byly navrženy změny tohoto systému tak, aby mohly být vytvořené falzifikáty vylepšeny. Tímto byly splněny všechny body zadání této práce.

Díky experimentálnímu pojetí zadání a také jeho značné neurčitosti a složitosti bylo třeba potýkat se s řadou problémů. Můj návrh práce přinesl nutnost ponořit se do mnoha různých odvětví, a snad i proto má práce velmi výrazný osobní přínos - přes výběr vhodných senzorů, návrh a sestrojení propisky (kdy největším problémem byl akt pájení, se kterým jsem neměl dostatečné zkušenosti), navržení matematického modelu nutného k získání požadovaných vlastností písma či podpisu z výstupních dat senzorů, zpracování těchto dat a jejich potřebné zpřesnění, zpracování obrázku až po potřebu důkladně pochopit princip práce 3D tiskárny. Nutnost osvojení si všech těchto problematik mi bezesporu rozšířila

obzory neocenitelným způsobem, svou práci z tohoto pohledu vnímám jednoznačně velmi kladně.

Vědecký přínos práce podle znalců v oboru psaného textu existuje - díky jejich zpětné vazbě bylo navíc možné navrhnout vylepšení tohoto systému tak, aby mohly být falzifikáty dále vylepšovány. Znamená to, že práce má smysl také v případě potenciálního pokračování výzkumu v této oblasti, jelikož bylo prokázáno, že navržený princip může dobře fungovat, potřebuje však vyšší kvalitu komponent systému.

Literatura

- [1] Anderson, T.: *Anatomy of a 3D Printer: How Does a 3D Printer Work?* 2016, [Online; navštíveno 6. 5. 2019].
URL <https://www.matterhackers.com/articles/anatomy-of-a-3d-printer>
- [2] Arduino: *MPU-6050 Accelerometer + Gyro*. [Online; navštíveno 10. 1. 2019].
URL <https://playground.arduino.cc/Main/MPU-6050>
- [3] Berndt, D. J.; Clifford, J.: Using dynamic time warping to find patterns in time series. In *KDD workshop*, ročník 10, Seattle, WA, 1994, s. 359–370.
URL <https://www.aaai.org/Library/Workshops/1994/ws94-03-031.php>
- [4] Bhattacharya, I.; Ghosh, P.; Biswas, S.: Offline Signature Verification Using Pixel Matching Technique. *Procedia Technology*, ročník 10, 2013: s. 970 – 977, ISSN 2212-0173, first International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA) 2013.
URL <http://www.sciencedirect.com/science/article/pii/S2212017313006075>
- [5] Butterworth, S.: On the theory of filter amplifiers. *Wireless Engineer*, ročník 7, č. 6, 1930: s. 536–541.
URL https://www.changpuak.ch/electronics/downloads/On_the_Theory_of_Filter_Amplifiers.pdf
- [6] Chan, K. H.; Hayya, J. C.; Ord, J. K.: A note on trend removal methods: the case of polynomial regression versus variate differencing. *Econometrica (pre-1986)*, ročník 45, č. 3, 1977: str. 737.
URL <https://search.proquest.com/docview/214668633?pq-origsite=gscholar>
- [7] Corinne, T.: *Accelerometer Basics*. [Online; navštíveno 10. 1. 2019].
URL <https://learn.sparkfun.com/tutorials/accelerometer-basics/all>
- [8] Cover, T. M.; Hart, P. E.; aj.: Nearest neighbor pattern classification. *IEEE transactions on information theory*, ročník 13, č. 1, 1967: s. 21–27.
URL
<https://www.cs.bgu.ac.il/~adsmb182/wiki.files/borak-lecture%20notes.pdf>
- [9] Davis, P. J.: *Interpolation and approximation*. Courier Corporation, 1975.
- [10] Deodhare, D.; Suri, N. R.; Amit, R.: Preprocessing and Image Enhancement Algorithms for a Form-based Intelligent Character Recognition System. *IJCSA*, ročník 2, č. 2, 2005: s. 131–144.

- [11] Drahanský, M.: *Rozpoznávání podle písma a podpisu*. 2005. Přednáška v předmětu Biometrické systémy. Vysoké učení technické v Brně, Fakulta informačních technologií.
- [12] Drahanský, M.: *Úvod do biometrických systémů*. 2018. Přednáška v předmětu Biometrické systémy. Vysoké učení technické v Brně, Fakulta informačních technologií.
- [13] Esfandyari, J.; De Nuccio, R.; Xu, G.: *Introduction to MEMS gyroscopes*. 2010, [Online; navštívěno 6. 5. 2019].
URL <https://electroiq.com/2010/11/introduction-to-mems-gyroscopes/>
- [14] Gordon, W. J.; Riesenfeld, R. F.: B-spline curves and surfaces. In *Computer aided geometric design*, Elsevier, 1974, s. 95–126.
- [15] Graves, A.; Fernández, S.; Gomez, F.; aj.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, s. 369–376.
- [16] Guo, Z.; Hall, R. W.: Parallel thinning with two-subiteration algorithms. *Communications of the ACM*, ročník 32, č. 3, 1989: s. 359–373.
- [17] Hans, S.; Gupta, S.: Preprocessing Algorithm for Offline Signature Verification System. 04 2012.
- [18] Higgins, W. T.: A comparison of complementary and Kalman filtering. *IEEE Transactions on Aerospace and Electronic Systems*, , č. 3, 1975: s. 321–325.
- [19] HillcrestLabs: *What is an IMU sensor?* 2018, [Online; navštívěno 6. 5. 2019].
URL <https://www.hillcrestlabs.com/posts/what-is-an-imu-sensor>
- [20] Hoag, D.: Apollo guidance and Navigation: Considerations of apollo imu gimbal lock. *Cambridge: MIT Instrumentation Laboratory*, 1963: s. 1–64.
- [21] Hwang, H.; Haddad, R. A.: Adaptive median filters: new algorithms and results. *IEEE Transactions on image processing*, ročník 4, č. 4, 1995: s. 499–502.
- [22] Ionescu, H.: *6 degrees of freedom*. [Online; navštívěno 10. 1. 2019].
URL https://en.wikipedia.org/w/index.php?title=File:6DOF_en.jpg
- [23] Karnin, E. D.: A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, ročník 1, č. 2, June 1990: s. 239–242, ISSN 1045-9227.
- [24] Keysers, D.; Deselaers, T.; Rowley, H. A.; aj.: Multi-Language Online Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 39, č. 6, June 2017: s. 1180–1194, ISSN 0162-8828.
- [25] LaValle, S. M.: *Yaw, pitch and roll rotations*. [Online; navštívěno 10. 1. 2019].
URL <http://planning.cs.uiuc.edu/node102.html>
- [26] Memsexchange: *What is MEMS?* [Online; navštívěno 6. 5. 2019].
URL <https://www.mems-exchange.org/MEMS/what-is.html>

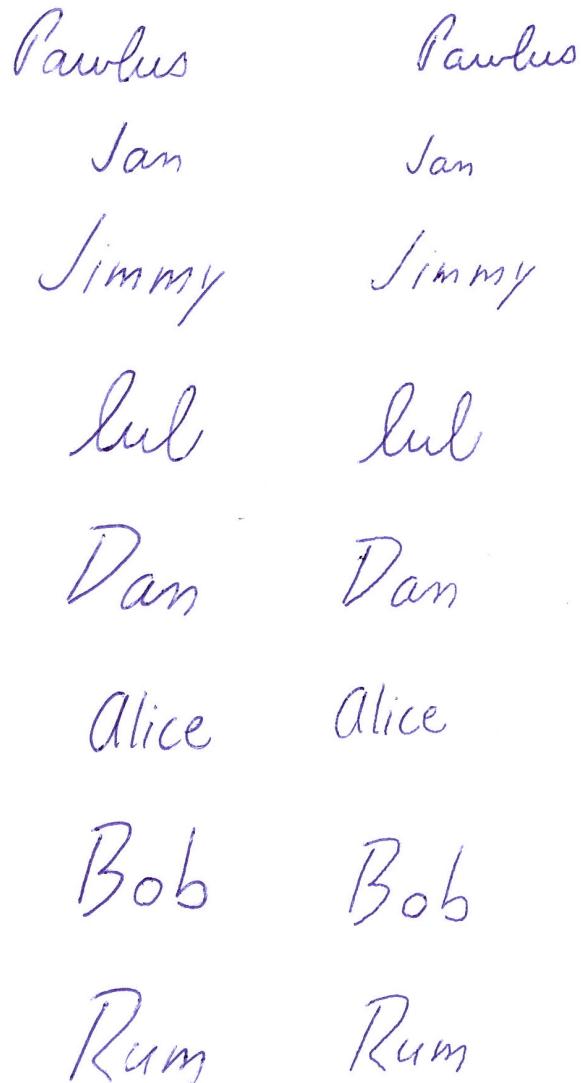
- [27] Miksik, O.; Mikolajczyk, K.: Evaluation of local detectors and descriptors for fast feature matching. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, IEEE, 2012, s. 2681–2684.
- [28] Ng, A. Y.; Jordan, M. I.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, 2002, s. 841–848.
- [29] Felipe Belem de Oliveira, L.; Guest, R.: An assessment of dynamic signature forgery creation methodology and accuracy. 06 2015.
- [30] Pachwicewicz, M.; Weremczuk, J.; Danielewski, K.: MEMS inertial sensors measurement errors. 2018.
URL <https://doi.org/10.11117/12.2501612>
- [31] Paret, D.; Fenger, C.: *The I2C Bus: From Theory to Practice*. 1997, ISBN 0471962686.
- [32] Pawlus, J.: *Zařízení pro napodobení statických a dynamických vlastností písma*. Brno, 2019. Semestrální projekt. Vysoké učení technické v Brně, Fakulta informačních technologií.
- [33] Rak, R.; Porada, V.: *Verifikace osoby na základě ověřování jejího podpisu*. 2016.
URL <http://www.sinz.cz/archiv/docs/si-2006-06-341-350.pdf>
- [34] Richiardi, J.; Katabdar, H.; Drygajlo, A.: Local and Global Feature Selection for On-line Signature Verification. 01 2005, s. 625–629, doi:10.1109/ICDAR.2005.152.
- [35] Sahoo, P. K.; Soltani, S.; Wong, A. K.: A survey of thresholding techniques. *Computer vision, graphics, and image processing*, ročník 41, č. 2, 1988: s. 233–260.
- [36] Savitzky, A.; Golay, M. J.: Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, ročník 36, č. 8, 1964.
- [37] Shoda, M.; Ishizuya, T.: Thermal infrared camera. Říjen 17 2000, uS Patent 6,133,569.
- [38] Smith, R. W.: *The extraction and recognition of text from multimedia document images*. Dizertační práce, University of Bristol, 1987.
- [39] Smith III, J. O.: *Introduction to Digital Filters: with Audio Applications*. 1997, ISBN 0974560715.
- [40] Sunil, R.: *Understanding Support Vector Machine algorithm*.
URL <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>
- [41] Wigington, C.; Tensmeyer, C.; Davis, B.; aj.: Start, Follow, Read: End-to-End Full-Page Handwriting Recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, s. 367–383.
- [42] Yang, L.; Widjaja, B.; Prasad, R.: Application of hidden Markov models for signature verification. *Pattern Recognition*, ročník 28, č. 2, 1995: s. 161 – 170, ISSN 0031-3203.
URL <http://www.sciencedirect.com/science/article/pii/003132039400092Z>

- [43] Zhang, T.; Suen, C. Y.: A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, ročník 27, č. 3, 1984: s. 236–239.
- [44] Španěl, M.; Hulík, R.; Havel, J.: *Geometrické transformace ve 2D a 3D*. 2019. Přednáška v předmětu Základy počítačové grafiky. Vysoké učení technické v Brně, Fakulta informačních technologií.

Příloha A

Ukázka výstupu

Ukázka výstupu dvojic originál a falzifikát písma či podpisu lze vidět na obrázku A.1.



Obrázek A.1: Ukázka výstupního listu obsahujícího dvojice originál (vlevo) a falzifikát (vpravo) písma či podpisu.