

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

**Automatizace platformy Autopen
pomocí Computer vision**

Teodor Machart
Hlavní město Praha

Praha, 2021

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST
Obor č. 18: Informatika

**Automatizace platformy Autopen
pomocí Computer vision**

**Autopen platform automation via
Computer vision**

Autor: Teodor Machart

Škola: Gymnázium Jana Keplera - Parléřova 2, 169 00 Praha 6

Kraj: Hlavní město Praha

Konzultant: Ing. Daniel Krpelík

Praha, 2021

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějsích předpisů.

V Praze dne:

Podpis:

Poděkování

Srdečně děkuji mému trpělivému konzultantu Danielu Krpelíkovi.

Anotace

Cílem této práce je poskytnout nástroje potřebné k plné automatizaci platformy Autopen, která se běžně využívá k reprodukci psaného textu. Výsledný software je následně využit v mobilní aplikaci ovládající podomácku vyrobený Autopen. Taková sestava je schopná během necelé minuty extrahovat křivky z fotografie a následně je zvěčnit na papír. Práce jasně demonstruje, jak zranitelné jsou podpisy jakožto identifikační metoda.

Klíčová slova

Autopen; Stegerův algoritmus; Computer vision; Detekce křivek

Anotation

The aim of this thesis is to provide instruments needed for automation of Autopen platform which is commonly used to reproduce written word. The resulting software is then employed in a mobile application controlling home-made Autopen. Such system is capable of extracting curve information from an image and subsequently drawing it on paper in under a minute. The thesis is a clear demonstration of vulnerability of handwritten signature as an identifier.

Keywords

Autopen; Stegers algorithm; Computer vision; Curve detection

Obsah

Úvod	1
1 Platforma Autopen	2
1.1 Pracovní obálka	3
1.1.1 Definice Pracovní obálky	3
1.1.2 Hranice obálky	3
1.1.3 Upravení hranic obálky s ohledem na 1.2	5
1.1.4 Závislost tvaru obálky na rozměrech Autopenu	6
1.2 Optimální oblast pro kreslení	7
1.2.1 Numerické řešení rovnice (12)	8
1.2.2 Přepočet dvojice (x, y) na dvojici (α, β)	12
2 Stegerův algoritmus	13
2.1 Konvoluce a lokální parciální derivace	14
2.2 Směr kolmý na křivku	15
2.2.1 První derivace ve směru kolmém na křivku	16
2.2.2 Druhá derivace ve směru kolmém na křivku	17
2.3 Spojování nalezených pixelů v křivky	18
2.3.1 Zakomponování globálních vlastností	19
2.3.2 Výběr počátečních bodů S_0	19
2.3.3 Pořadí křivek	21
2.4 Stanovení a provázání konstant	21
3 Android aplikace	22
3.1 GUI - Uživatelské prostředí	22
3.2 Odesílání dat přes Ably	23
4 Konstrukce demonstračního Autopenu	24
4.1 Rozlišení kopie	25
4.2 Srovnání originálů a kopií	26
5 Padělání podpisů	26
Závěr	28

Úvod

Autopen je přístroj užívaný k věrohodné replikaci podpisů, hojně byl využíván vrcholnými americkými politiky již od počátku 19. století. Tehdy byla podpisová informace uchovávána čistě mechanicky v podobě zvlněného dřevěného kotouče. S příchodem výpočetních technologií v druhé polovině 20. století dřevěné kotouče nahradily paměťové disky, což Autopeny značně zdostupnilo, dnes jsou mnohými využívány jak k replikaci podpisů, tak například k „ručnímu“ psaní pohlednic. Kvůli důvěrnému charakteru podpisů se společnosti vyrábějící Autopeny zdráhají sdílet jakékoli informace, proto na internetu není k nalezení jediný pramen, který by diskutoval mechaniku Autopenu nebo způsob, jakým společnosti získávají podpisovou informaci z fotografií, které jsou jim zaslány zákazníkem.

Tato práce si bere za úkol vytvoření teoretických podkladů pro zacházení s Autopenem a jeho automatizování, vychází z obecných parametrů přístroje, tak aby bylo možné poznatky aplikovat na libovolný Autopen. K automatizaci využívá upravený Stegerův algoritmus, který slouží k extrahování podpisu z fotografie. Umožnuje tak samotnému uživateli extrahovat podpisy z fotografií, namísto dosavadních praktik, při kterých bylo nutné zaslat fotografie výrobci Autopenu, který následně provedl extrakci.

Mobilní aplikace vybavená těmito nástroji by pomohla stále rostoucí komunitě kutilů a bastlířů, kteří jsou schopni sestrojit funkční Autopen, ale postrádají schopnosti potřebné k vytvoření softwaru pro takový přístroj. Zmíněná automatizace by mohla pomoci i společnostem provozujícím Autopeny, za předpokladu, že extrakci provádí manuálně, to jest velmi pravděpodobné, jelikož na volně dostupných videích je vidět, že podpisy jsou vždy prováděny jedním tahem zleva doprava.

První kapitola se věnuje geometrickému popisu Autopenu a analýze oblasti, do které Autopen dosáhne, takzvané pracovní obálky. Navazující kapitola shrnuje poznatky o Stegerově algoritmu a nabízí řešení problému, odkud začít spojovat body ležící na křivce do větších segmentů. Kapitoly třetí a čtvrtá pojednávají o aplikaci a Autopenu sestrojených za účelem demonstrovat funkčnost teoretičky předestřených témat z předchozích dvou kapitol.

1 Platforma Autopen

Platformou Autopen je v této práci méněna každá fyzická konstrukce odpovídající geometrickému modelu na obrázku č. 1, kde v bodě C je umístěn hrot pera.

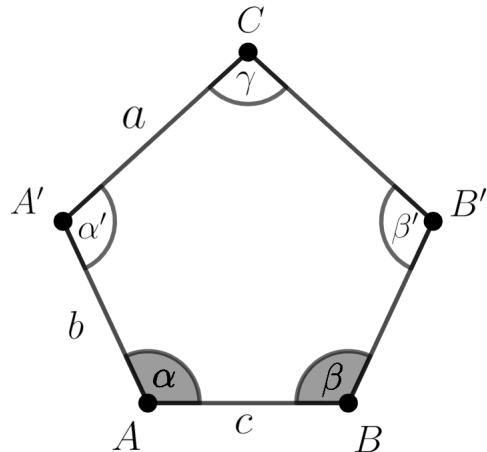
Velikosti všech přímek jsou pevně dané, konstrukce tedy obsahuje podle rovnice vazbové závislosti jen dva stupně volnosti, těmi mohou být například úhly α a β . Na základě těchto dvou úhlů a nutných podmínek:

$$\gamma < 180^\circ ; \alpha' < 180^\circ ; \beta' < 180^\circ, \quad (1)$$

bez kterých by bylo řešení nejednoznačné, lze určit polohu bodu C . Za stejných podmínek lze obdobným způsobem z pozice bodu C určit velikost úhlů α a β . Této vlastnosti je užito v oddílu 1.2 ke přepočítání dvojice (x, y) na jí odpovídající dvojici (α, β) .

Nicméně je zřejmé, že ne pro všechny dvojice (x, y) existuje odpovídající dvojice (α, β) . Tento problém je řešen v oddílu 1.1.

Důležitým předpokladem je také symetrie celé platformy, která značně zjednoduší celý problém a zároveň nijak nebrání funkčnosti přístroje. Zavádíme tedy podmínu: $|CA'| = |CB'| \wedge |AA'| = |BB'|$.



Obrázek č. 1: Geometrický model platformy Autopen

Mnohé přístroje dostupné na trhu využívají $|AB| = 0$, to je ovšem jen speciální případ obrázku č. 1, není proto nutné ho uvažovat samostatně.

1.1 Pracovní obálka

V tomto oddílu jsou odvozeny geometrické vlastnosti hranic pracovní obálky, na základě kterých je určena poloha obdélníku se zadaným poměrem stran, tak aby byl co největší. (Bez záruky toho, zda je opravdu ten největší možný.)

1.1.1 Definice Pracovní obálky

Pracovní obálku, neboli „oblast, do které přístroj dosáhne bodem C “, definujeme jako množinu uspořádaných dvojic čísel (x, y) , pro které lze nalézt odpovídající dvojici (α, β) . Počátek souřadného systému položíme do středu přímky AB , osa y je kolmá na AB .

1.1.2 Hranice obálky

V praxi je problematické, když vnitřní úhly (s výjimkou α a β) dosahují hodnot blízkých 180° , kvůli nedokonalosti konstrukce by se ramena mohla „vykloubit“, tedy nedodržela by se podmínka (1), proto je praktické zavést maximální hodnotu těchto úhlů ϕ .

Podobný čistě praktický problém nastává pro velké úhly α a β . Hrozí, že by se ramena srazila s osami motoru atd., proto je praktické zastropovat oba úhly hodnotou ψ .

Stejně tak je dobré omezit úhly α' a β' minimálními hodnotami, jelikož by mohlo dojít ke srážce s osou motoru. (γ nikoliv, minimální hodnoty zde nepředstavují problém) Rozumnou minimální hodnotou je $180^\circ - \phi$.

Díky symetričnosti nám stačí řešit hranice pro pravou polorovinu. Zavedení limitních hodnot se projevuje následovně:

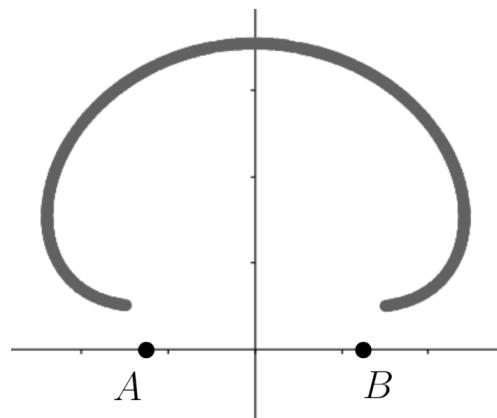
$$C_y < \sqrt{a^2 + b^2 - 2ab \cos(\phi)} - \left(x + \frac{c}{2}\right)^2 = k_1 \quad (2)$$

$$C_y > \sqrt{a^2 - \left(x + a \cos(\psi) - \frac{c}{2}\right)^2} - b \sin(\psi) = k_2 \quad (3)$$

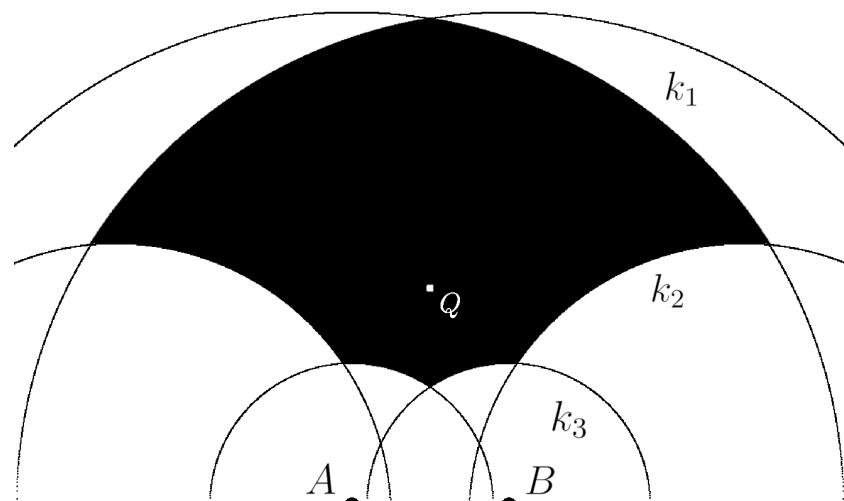
$$C_y > \sqrt{a^2 + b^2 + 2ab \cos(\phi)} - \left(x - \frac{c}{2}\right)^2 = k_3 \quad (4)$$

Výrazy v nerovnicích (2) až (4) jsou rovnicemi půlkružnic. Nicméně omezení maxima γ nelze vyjádřit takto jednoduše. Na obrázku č. 2 jsou vyznačené hrany oblasti, do které nemůžeme bod C kvůli tomuto omezení umístit. Nadále si vystačíme s informací, že pro nejvyšší bod tohoto útvaru platí:

$$Q = \left[0 ; \sqrt{b^2 - \left(a \sin\left(\frac{\phi}{2}\right) - \frac{c}{2} \right)^2} + a \cos\left(\frac{\phi}{2}\right) \right] \quad (5)$$



Obrázek č. 2: Náčrt oblasti nedostupné kvůli omezení maxima γ



Obrázek č. 3: Oblast vyhovující podmínkám (2)(3)(4)

1.1.3 Upravení hranic obálky s ohledem na 1.2

Pro potřeby oddílu 1.2 je nutné, aby f_1 , funkce popisující souřadnici y spodní hranice obálky, byla neklesající. Zavedeme tedy dvě konstanty:

$$p = \begin{cases} Q_y & \text{pro } Q_y \geq \sqrt{a^2 + b^2 + 2ab \cos(\phi)} \\ \sqrt{a^2 + b^2 + 2ab \cos(\phi)} & \text{pro } Q_y < \sqrt{a^2 + b^2 + 2ab \cos(\phi)} \end{cases} \quad (6)$$

$$q = a + b \sin(\psi) \quad (7)$$

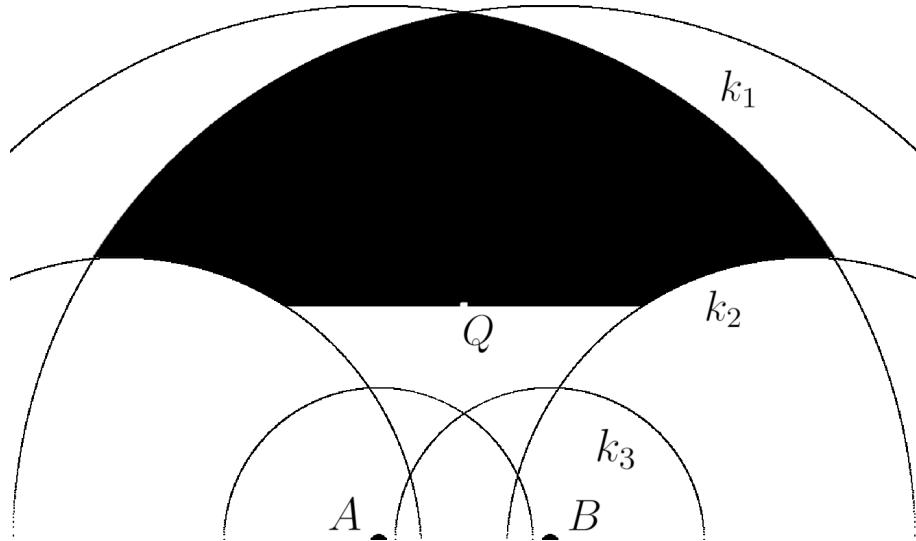
Pro C_y zavádíme podmínky:

$$C_y \geq p \quad (8)$$

$$C_y \geq q \quad \text{pro} \quad C_x \geq \frac{c}{2} - b \cos(\psi) \quad (9)$$

Takto zavedená konstanta p se zbavuje problému se složitým tvarem oblasti z obrázku č. 2 a problémů s možným klesáním v druhé polovině půlkružnice k_3 . Nabízí se možnost místo p zavést dvě přímky, jednu, která by omezovala bod Q a druhou, která by omezovala klesající část půlkružnice k_3 . Z praktického hlediska nemá takovou variantu smysl uvažovat, jelikož by se pracovní obálka zvětšila jen minimálně a celý problém by byl rázem mnohem komplexnější.

Zavedená přímka q se zbavuje problému s klesající částí půlkružnice k_2 . Toto omezení se projevuje především při nízkých hodnotách c .

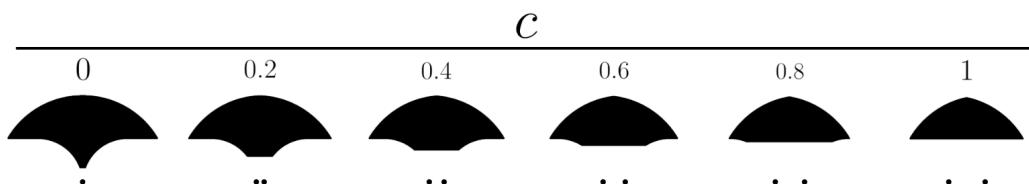


Obrázek č. 4: Oblast vyhovující podmínkám (2)(3)(4) a (8)(9)

1.1.4 Závislost tvaru obálky na rozměrech Autopenu

Výběr ideálních rozměrů závisí především na účelu, za kterým je daný Autopen sestrojen. Nicméně na základě pozorování je možné některé nepraktické rozměry zamítnout. Nejprve postavme a a b rovno jedné a zkoumejme, jak se obálka mění v závislosti na c , viz obrázek č. 5. (Tečky pod obálkou symbolizují body A a B .)

Je patrné, že čím menší je c , tím větší je obálka. Nejlepším řešením by tedy bylo, aby body A a B splynuly. Taková varianta je mechanicky náročná, proto se v praktické části spokojíme s pokud možno co nejmenším nenulovým c .



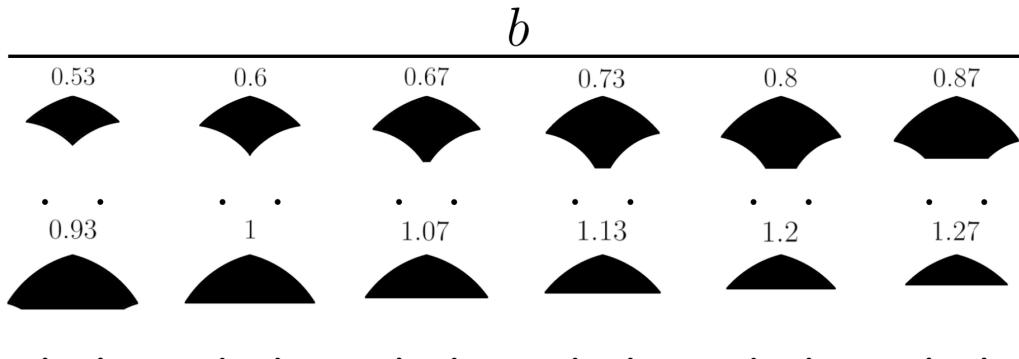
Obrázek č. 5: Závislost tvaru obálky na hodnotě c

Pro závislost tvaru obálky na a a b svážeme obě proměnné vztahem $a + b = 2$. Vzdálenost c položíme rovnou jedné, jelikož nás zajímá ideální poměr. Kdybychom zkoumali a a b nezávisle na sobě, zjistili bychom, že čím větší jsou, tím větší je také obálka, což je bezcenná informace.

Z obrázku č. 6 lze vypozorovat, že poměr $\frac{a}{b} \approx 1.5$ bude ideální pro kreslení vysokých objektů, $\frac{a}{b} \approx 1$ bude ideální pro kreslení širokých objektů a poměr $\frac{a}{b} \approx 1.2$ bude poměrně univerzální, připouštějící jak široké, tak vysoké objekty. Vyloženě nevhodné jsou pak poměry $\frac{a}{b} < 1$.

Je třeba brát v potaz, že se tyto poměry mění, když c nabývá hodnot blízkých nule, nebo naopak výrazně větších než jedna. Pro účel praktické části jsou ovšem odhadnuté hodnoty relevantní.

Pro speciální případ $c = 0$ pak dostáváme nejvyšší i nejširší obálku s hodnotou $\frac{a}{b} \approx 1$



Obrázek č. 6: Závislost tvaru obálky na hodnotě b

1.2 Optimální oblast pro kreslení

V této části je prezentován algoritmus, který má za účel najít uvnitř obálky největší obdélník se zadáným poměrem stran a jednou dvojicí stran rovnoběžnou s úsečkou AB . Hodnota poměru bude vyplývat z rozměrů množiny bodů získaných Stegerovým algoritmem v kapitole 2, tento poměr výšky (rozdíl největšího a nejmenšího y množiny) ku šířce (rozdíl největšího a nejmenšího x množiny) nazývejme k . Pokud je šířka nulová, přidělíme k hodnotu takovou, že $k \gg 1$. Stejně tak pro nulovou výšku přidělme hodnotu takovou, že $k \ll 1$. (Případ, kdy je výška i šířka nulová, řešit nemusíme, jelikož výstupem Stegerova algoritmu budou vždy množiny s více něž jedním bodem.)

Levý ze dvou bodů, ve kterých k_2 nabývá hodnoty p pojmenujme P .

$$P_x = \frac{c}{2} - b \cos(\phi) - \sqrt{a^2 - (p - b \sin(\phi))^2} \quad (10)$$

Stanovme funkci $f_1(x)$ popisující souřadnici y spodní hrany obálky a funkci $f_2(x)$ popisující souřadnici y horní hrany obálky:

$$f_1(x) = \begin{cases} p & \text{pro } p \geq q \\ p & \text{pro } p < q \wedge x \leq P_x \\ k_2 & \text{pro } p < q \wedge P_x < x < \left(\frac{c}{2} - b \cos(\phi)\right) \\ q & \text{pro } p < q \wedge x > \left(\frac{c}{2} - b \cos(\phi)\right) \end{cases} \quad (11)$$

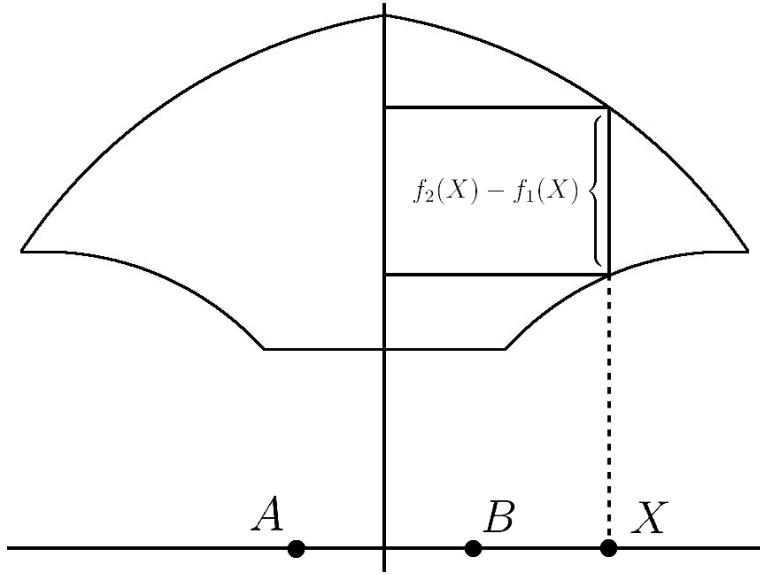
$$f_2(x) = k_1$$

Díky požadavkům zavedeným na C_y v oddílu 1.1.3 víme, že f_1 je neklesající funkce. Zároveň lehce ověříme, že funkce f_2 je ostře klesající na intervalu $(0 ; \sqrt{a^2 + b^2 - 2ab \cos(\phi)} - \frac{c}{2})$, viz rovnice (2). Rozdíl ostře klesající a neklesající funkce, neboli součet ostře klesající a nerostoucí funkce, je vždy funkci ostře klesající, z toho vyplývá, že $f_2 - f_1$ je funkci ostře klesající. Takovou funkci lze vynásobit libovolným kladným číslem a zachová si svou monotónnost, tedy funkce $\frac{f_2(x) - f_1(x)}{x}$ pro $x > 0$ je opět funkci ostře klesající.

Jednou z vlastností ryze monotónních funkcí je, že nabývají každé hodnoty jen jednou, z toho můžeme usoudit, že rovnice (12) má jedno jediné řešení, nazveme ho X .

$$\frac{f_2(x) - f_1(x)}{x} = 2k \quad (12)$$

Rovnice (12) je zápisem toho co se děje na obrázku č. 7. Hledáme obdélník s požadovaným poměrem výšky ku délce. Je důležité nezapomenout, že pracujeme pouze s pravou polovinou a požadovaný poměr k musíme tedy zdvojnásobit.



Obrázek č. 7: Ilustrace rovnice (12)

1.2.1 Numerické řešení rovnice (12)

Rovnice (12) buď není obecně analyticky řešitelná, nebo jen s velkými obtížemi, proto bude nejjednodušší ji řešit numericky. Nejdřív upravíme rovnici (12) tak, aby měla na pravé straně nulu a následně budeme hledat kořeny funkce $h(x)$, která zbyla na levé straně:

$$f_2(x) - f_1(x) - 2kx = 0 \quad (13)$$

Vycházíme z podmínky $x \neq 0$, proto je tento krok ekvivalentní úpravou.

$$h(x) = f_2(x) - f_1(x) - 2kx \quad (14)$$

K vyřešení je užita dobré známá Newton-Raphsonova metoda, tak jako je popsaná zde: [1]. Takto nastavená iterační metoda pro správně zvolené x_0 konverguje ke kořeni funkce $h(x)$:

$$x_{n+1} = x_n - \frac{h(x_n)}{h'(x_n)} \quad (15)$$

Otázka, pro která x_0 bude metoda konvergovat, není zdaleka triviální, proto si v této práci vystačíme s pozorováním a jeho potvrzením pomocí pokusu.

Kdybychom v rovnici (13) ponechali x ve jmenovateli, druhá derivace této funkce by nabývala velkých hodnot, což by neumožňovalo splnit podmínu kvadratické konvergence [2]:

$$\sup_{x \in I} \frac{1}{2} \left| \frac{h''(x)}{h'(x)} \right| \cdot \varepsilon_0 < 1 \quad (16)$$

Kde $\varepsilon_0 = |X - x_0|$ a kde I je interval $\langle X - \varepsilon_0 ; X + \varepsilon_0 \rangle$.

Funkce na obrázku č. 8 byla vybrána tak, aby byla co nejobecnější, f_1 nabývá všech svých možných hodnot p , q a k_2 . Místo, kde f_1 mění svou hodnotu z p na k_2 je označené M . Místo, kde f_1 mění svou hodnotu z k_2 na q je označené N .

Z grafu je patrné, že druhá derivace je největší v jeho pravé části, proto pokud chceme dosáhnout konvergence i pro kořeny nacházející se v této pravé části (k nabývá malých hodnot), musíme zmenšit vzdálenost ε_0 . proto je rozumné x_0 zvolit takové, že:

$$x_0 \rightarrow \sqrt{a^2 + b^2 - 2ab \cos(\phi)} - \frac{c}{2} \quad (17)$$

Tedy, x_0 blížící se ke hranici definičního oboru a ke konci půlkružnice k_1 . Jestliže by takové omezení nestačilo, bylo by příhodné zmenšit Newtonovský krok.

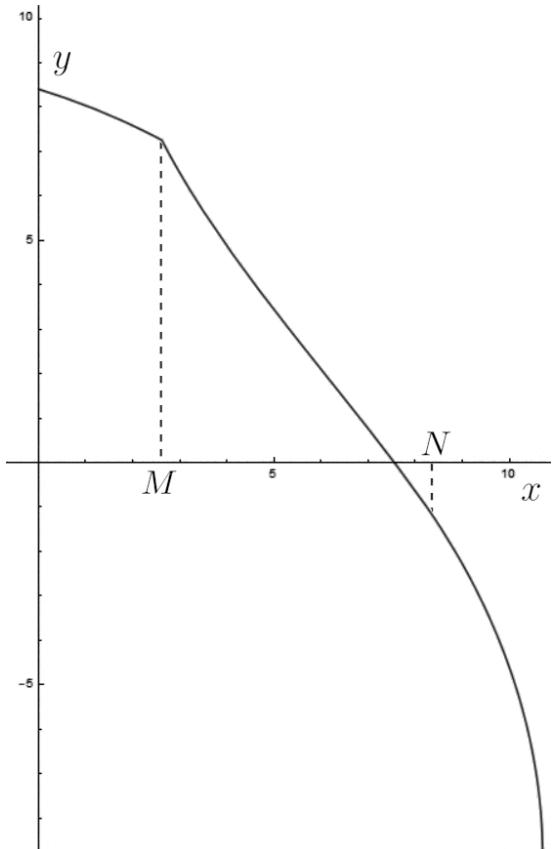
Problémem je, že $h(x)$ nemá v bodech M a N spojitou první ani druhou derivaci. Nesplňuje tedy jednu z podmínek [2]. To ovšem v praxi nevadí, jak ukazuje následující zkouška.

Pro takto definovanou Newtonovu metodu byl proveden test, během kterého bylo každé z proměnných a , b , c , ϕ , ψ a k přiřazeno osm hodnot co nejlépe reprezentujících reálné přístroje a zároveň i extrémní hodnoty v případě k . Následně program iteroval přes všechny možné kombinace těchto hodnot, tedy celkem 8^6 možností a kontroloval, zda pro dané hodnoty Newtonova metoda konverguje. Z kombinací hodnot, které splňovaly trojúhelníkovou nerovnost, konvergovaly naprostě všechny i bez použití zmenšeného Newtonovského kroku. Proto může tato Newtonova metoda být v praxi používána beze strachu z případné nekonvergence. Pokud by však obavy přetrvávaly nadále, je možné Newtonovský krok zmenšit například na polovinu a zvětšit množství iterací.

Šest iterací se pro normální Newtonovský krok ukazuje být jako dostačující počet pro určení X na tři desetinná místa. Operace ovšem není výpočetně náročná, proto si můžeme dovolit počet iterací zvýšit na 20, abychom měli jistotu, že obdržíme vždy správné číslo.

Ilustrační hodnoty

a	b	c	ϕ	ψ	k
6.7	5.7	3.0	160°	180°	0.1



Obrázek č. 8: Graf funkce $h(x)$ s význačnými body M a N

Dosazení ze (14) do rovnice (15) naleznete v Přílohách pod číslem 1, řešení je zdlouhavé a ve své podstatě triviální. Tohoto dosazení je využito v následujícím programu napsaném v jazyce Java, jehož vstupem jsou parametry a , b , c , ϕ , ψ a k a výstupem je X . (Program počítá se zadáním úhlů v radiánech.)

```
public static double newtonovaMetoda(a, b, c, phi, psi, k){

    double R, zx, zy, p, Px, e, pi, Qy;
    boolean pMimoK2 = false;

    pi = 3.1415926535
    R = Math.sqrt(a*a + b*b - 2 * Math.cos(psi) * a * b);
    zx = Math.cos(phi - pi) * b + c / 2;
    zy = -Math.sin(phi - pi) * b;
    Qy = sqrt(b*b - pow(a * sin(phi) - c/2, 2)) + a * cos(phi);
    r = a;

    if(Qy > sqrt(a*a + b*b - 2 * a * b * cos(pi - phi))) {
        p = Qy;
    }
    else {
        p = sqrt(a*a + b*b - 2 * a * b * cos(pi - phi));
    }

    if(p > (zy + a)) {
        pMimoK2 = true;
    }
    else {
        Px = zx - sqrt(a*a - pow(p - zy, 2));
    }

    e = R - c / 2 - 0.000001;

    for(int i = 0; i < 20; i++) {
        if(e < Px || pMimoK2) {
            e -= (sqrt(R*R - pow(e + c/2, 2)) - p - 2 * k * e)/(-(e
                + c/2) / (sqrt(R*R - pow(e + c/2, 2))) - 2*k);
        }
        else if(e > zx) {
            e -= (sqrt(R*R - pow(e + c/2, 2)) - zy - a - 2 * k * e)
                / (-(e + c/2) / (sqrt(R*R - pow(e + c/2, 2))) - 2*
                    k);
        }
        else {
            e -= (sqrt(R*R - pow(e + c/2, 2)) - sqrt(a*a - pow(e -
                zx, 2)) - 2 * k * e - zy) / ((e - zx) / (sqrt(a*a -
                    pow(e - zx, 2))) - (e + c/2) / (sqrt(R*R - pow(e +
                        c/2, 2))) - 2*k);
        }
    }
    return e;
}
```

Algoritmus č. 1: Program určený k nalezení optimálního obdélníku

1.2.2 Přepočet dvojice (x, y) na dvojici (α, β)

V předchozích oddílech jsme se ujistili, že pro všechny body uvnitř obdélníku s levým horním rohem M a pravým dolním rohem N existuje dvojice (α, β) .

$$\begin{aligned} M &= [-X ; k_1(X)] \\ N &= [X ; k_1(X) - 2kX] \end{aligned} \quad (18)$$

Do tohoto obdélníku lze libovolně namapovat body získané Stegerovým algoritmem. V demonstrační aplikaci je umožněno zadat přesnou výšku nebo délku kresleného objektu, který je poté přeškálován na tuto velikost a položen k hornímu okraji obdélníku MN. Taková operace je ovšem jednoduchá a rutinní, proto není třeba ji uvádět.

Pro takové dvojice (x, y) získáme dvojici (α, β) následujícím způsobem:

$$\alpha_0 = \arccos \left(\frac{a^2 - b^2 - y^2 - (x + \frac{c}{2})^2}{-2b\sqrt{y^2 + (x + \frac{c}{2})^2}} \right) + \arctan \left(\frac{y}{\frac{c}{2} + x} \right) \quad (19)$$

$$\alpha = \begin{cases} \alpha_0 & \text{pro } x \geq -\frac{c}{2} \\ \alpha_0 + 180^\circ & \text{pro } x < -\frac{c}{2} \end{cases}$$

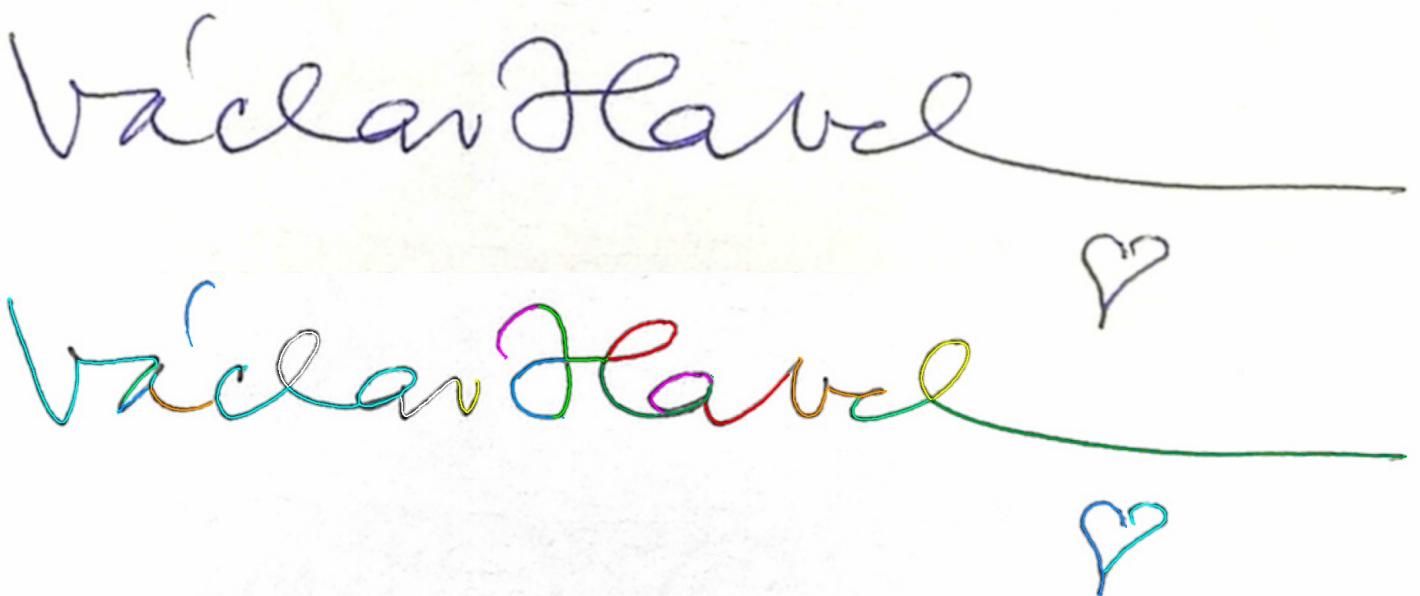
$$\beta_0 = \arccos \left(\frac{a^2 - b^2 - y^2 - (x - \frac{c}{2})^2}{-2b\sqrt{y^2 + (x - \frac{c}{2})^2}} \right) + \arctan \left(\frac{y}{\frac{c}{2} - x} \right) \quad (20)$$

$$\beta = \begin{cases} \beta_0 & \text{pro } x \leq \frac{c}{2} \\ \beta_0 + 180^\circ & \text{pro } x > \frac{c}{2} \end{cases}$$

2 Stegerův algoritmus

K získání informace z obrázků byl zvolen Stegerův algoritmus popsaný v práci: „An Unbiased Detector of Curvilinear Structures“[3] a jeho vylepšení popsané v práci: „Curve Tracing and Curve Detection in Images“[4]. Tato kapitola se zaměří na vysvětlení celého algoritmu za použití informací ze zmíněných prací. Následně představí problémy, které je třeba překonat, aby mohl být celý proces automatizován. V průběhu bude demonstrovat, jak si algoritmus vede při praktickém použití v aplikaci z kapitoly 3.

Stegerův algoritmus slouží k detekci bodů náležících křivkám ve fotografii a k jejich následnému trasování, tedy spojení jednotlivých bodů křivek do větších souvislých celků. (Jak je vidět na obrázku č. 9, kde jsou jednotlivé extrahované křivky barevně odlišeny).



Obrázek č. 9: Ilustrace vstupu a výstupu Stegerova algoritmu

Prvním krokem je určení lokálního směru křivky pomocí parciálních derivací I_x , I_y , I_{xx} , I_{yy} , I_{xy} každého z pixelů. Ty se získávají konvolucí obrázku derivacemi dvojrozměrné Gaussovy funkce. Se znalostí parciálních derivací a lokálního směru křivky na pixelu je možné určit, zda je první derivace ve směru kolmém na křivku rovna nule již v tomto pixelu. Jestliže ano, pixel leží na křivce. Takto získané pixely se spojují do souvislé křivky pokud leží vedle sebe a lokální směr křivky na nich se o mnoho neliší.

Práce [4] pak vylepšuje algoritmus pomocí globálních vlastností obrázku, při spojování pixelů bere ohled na předešlý směr křivky, který následuje, pokud se již v blízkosti posledního pixelu nenachází žádní vhodní kandidáti. To vede k lepšímu následování křivek na křížovatkách a při spojování přerušovaných křivek.

2.1 Konvoluce a lokální parciální derivace

Většina programovacích jazyků obsahuje knihovny, které umí aplikovat konvoluci velmi výpočetně efektivně, proto nemá smysl se zde zabývat jejím programovým řešením. Postačíme si s odkazem na nativní knihovnu **OpenCV**, která je napsána v C++ a je plně kompatibilní s android prostředím.

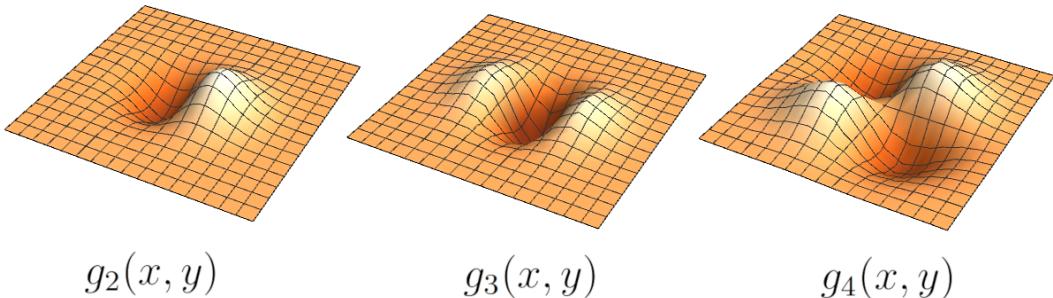
Ke konvoluci je zapotřebí filtr (kernel). Jak získat takový filtr Steger popisuje jen pro 1D situaci. U 2D situace hovoří o jejich využití, nikoliv však o jejich získání, což je pracný úkon, proto jsou v této části pouze výsledky, postup naleznete v Přílohách pod číslem 2.

$$g_1(x, y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (21)$$

$$g_2(x, y) = \frac{\partial g_1}{\partial x} = \frac{-x}{\sigma^3\sqrt{2\pi}} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (22)$$

$$g_3(x, y) = \frac{\partial^2 g_1}{\partial x^2} = \frac{x^2-\sigma^2}{\sigma^5\sqrt{2\pi}} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (23)$$

$$g_4(x, y) = \frac{\partial^2 g_1}{\partial x \partial y} = \frac{xy}{\sigma^5\sqrt{2\pi}} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (24)$$



Obrázek č. 10: Grafy derivací Gaussovy funkce

Filtry jsou po částech konstantní funkce, proto je třeba z uvedených funkcí vzít plošný integrál v okolí jednotlivých pixelů se souřadnicemi $a, b \in \mathbb{Z}$:

$$\iint_D g_2 dx dy = \frac{1}{2}(1 - e^{\frac{a}{\sigma^2}}) e^{-\frac{(a+0.5)^2}{2\sigma^2}} \left(\operatorname{erf}\left(\frac{b+0.5}{\sigma\sqrt{2}}\right) - \operatorname{erf}\left(\frac{b-0.5}{\sigma\sqrt{2}}\right) \right)$$

Kde $D = \{(x, y) : a - 0.5 < x \leq a + 0.5 \wedge b - 0.5 < y \leq b + 0.5\}$ a erf je chybová funkce (Většina programovacích jazyků obsahuje knihovnu pro její výpočet.) Analogicky i pro další dva filtry:

$$\iint_D g_3 \, dx \, dy = \frac{(e^{\frac{a}{\sigma^2}}(a - 0.5) - (a + 0.5))}{2\sigma^2} e^{-\frac{(a+0.5)^2}{2\sigma^2}} \left(\operatorname{erf}\left(\frac{b+0.5}{\sigma\sqrt{2}}\right) - \operatorname{erf}\left(\frac{b-0.5}{\sigma\sqrt{2}}\right) \right)$$

$$\iint_D g_4 \, dx \, dy = \frac{(e^{\frac{a}{\sigma^2}} - 1)(e^{\frac{b}{\sigma^2}} - 1)}{\sigma\sqrt{2\pi}} e^{-\frac{(a+0.5)^2 + (b+0.5)^2}{2\sigma^2}}$$

Pro získání filtru $\frac{\partial g_1}{\partial y}$ lze provést cyklickou záměnu (a, b) za (b, a) u plošného integrálu g_2 , nebo jednoduše filtr g_2 otočit o 90° , stejný postup lze uplatnit při získávání filtru $\frac{\partial^2 g_1}{\partial y^2}$ z filtru g_3 .

Pro urychlení výpočetního času je ideální mít filtr co nejmenší, proto je dobré brát v úvahu pouze určité množství pixelů, na kterých tyto funkce nabývají relevantních hodnot. To je třeba dělat s ohledem na velikost σ , které se v praktických případech pohybuje v intervalu $(1.5 ; 6.0)$. Rozumná velikost filtru pro tyto hodnoty je 35×35 . (Filtr musí být samozřejmě symetrický, tedy čtverec o straně s lichým počtem pixelů.)

Když filtr přesahuje ven z fotografie, takzvaně ho normalizujeme, tedy jeho hodnoty uvnitř fotografie naškálujeme tak, aby jejich součet byl jedna. Tím docílíme zachování ideální světlosti. (Většina knihoven provádějící konvoluci používá tento způsob oříznutí filtru jako výchozí.)

Po konvoluci fotografie výše uvedenými filtry dostáváme žádané parciální derivace pro jednotlivé pixely. Nabízí se ještě jeden přístup, jak takové derivace získat a to sice konvolucí se samotnou Gaussovou funkcí a následně takzvaným Sobelovým operátorem [5], nebo nějakým jemu podobným. Takové řešení je kvůli dvojí konvoluci výpočetně náročnější a zároveň méně přesné, proto se nehodí pro tuto aplikaci.

2.2 Směr kolmý na křivku

Steger uvádí, že směr, ve kterém je druhá derivace nejvyšší, je zároveň směr kolmý na křivku a také, že je to ten samý směr, kterým ukazuje vlastní vektor Hessovy matice H , který koresponduje k vlastnímu číslu s největší absolutní hodnotou. Jednotkový vektor v tomto směru značme (n_x, n_y) .

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix} \quad (25)$$

Steger doporučuje vlastní vektory získat numericky Jacobiho rotací [6]. Není ovšem důvod si tímto stěžovat práci, jelikož knihovna **OpenCV** obsahuje funkci *eigen*, která vrací jak všechny jednotkové vlastní vektory, tak k nim korespondující vlastní čísla.

2.2.1 První derivace ve směru kolmému na křivku

Jak je zmíněno v úvodu této kapitoly, jednou z vlastností pixelů křivky je, že první derivace ve směru (n_x, n_y) je nulová uvnitř tohoto pixelu. Nalezení místa, kde přesně je tato derivace rovna nule, řeší Steger následovně: Křivku approximujeme Taylorovým polynomem druhého stupně v bodě $(x+tn_x, y+tn_y)$:

$$z(x+tn_x, y+tn_y) = I(x, y) + \begin{bmatrix} tn_x & tn_y \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} tn_x & tn_y \end{bmatrix} H \begin{bmatrix} tn_x \\ tn_y \end{bmatrix} \quad (26)$$

Kde $z(x, y)$ je funkci naší původní fotografie, ale neceločíselné souřadnice jsou approximovány Taylorovým polynomem druhého stupně pro pixel jemuž náleží. (Steger používá matematicky čistší definici ve svojí původní publikaci [3], pro účely této práce však stačí takto intuitivní definice.) První derivaci výrazu (26) podle vektoru (tn_x, tn_y) položme rovnou nulovému vektoru. Takto zjistíme v jaké vzdálenosti t ve směru (tn_x, tn_y) je první derivace nulová.

$$\begin{bmatrix} I_x + \frac{n_y I_y}{n_x} \\ I_y + \frac{n_x I_x}{n_y} \end{bmatrix} + t \begin{bmatrix} n_x I_{xx} + 2n_y I_{xy} + \frac{n_y^2 I_{yy}}{n_x} \\ \frac{n_x^2 I_{xx}}{n_y} + 2n_x I_{xy} + n_y I_{yy} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (27)$$

Díky symetrii x a y ve výrazu (27) jsou oba kořeny stejné, po vyřešení pro t dostaváme:

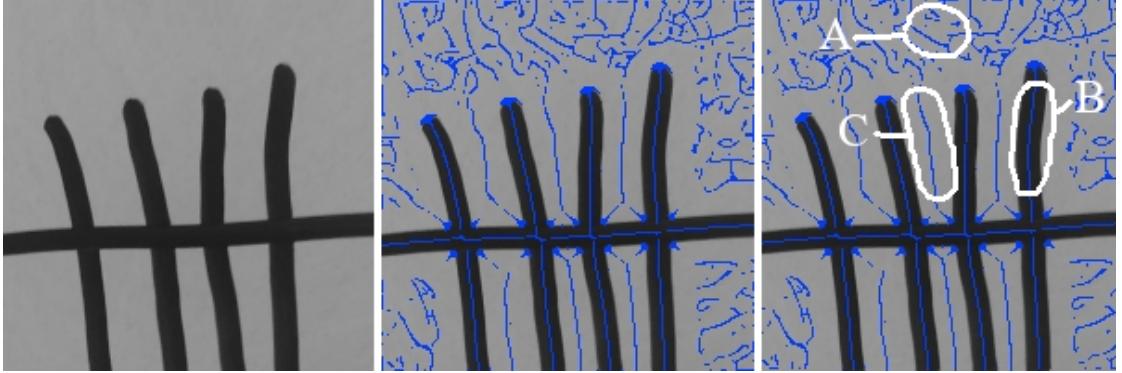
$$t = -\frac{n_x I_x + n_y I_y}{n_x^2 I_{xx} + 2n_x n_y I_{xy} + n_y^2 I_{yy}} \quad (28)$$

Tedy druhá derivace v kolmém směru zaniká na konci vektoru (tn_x, tn_y) s počátkem ve středu daného pixelu. Takové místo náleží danému pixelu, jestliže bude platit:

$$|tn_x| < \frac{1}{2} \wedge |tn_y| < \frac{1}{2} \quad (29)$$

Pixel splňující podmínu (29) je pixelem křivky. Takové pixely jsou vidět vybarveny modře na obrázku č. 11 vytvořeném v programu z kapitoly 3.

Takto stanovené podmínce ovšem odpovídají i nežádoucí pixely z oblastí A a C . V oblasti C jsou pixely odpovídající světlé křivce. Jak vybrat jen tmavé nebo jen světlé křivky, je popsáno v oddílu 2.2.2. V oblasti A převažuje šum, ten se objevuje vždy v oblastech, které nejsou podobné křivkám.



Obrázek č. 11: Pixely splňující podmínu (28)

2.2.2 Druhá derivace ve směru kolmému na křivku

Této problematice se Steger věnuje jen letmo, zatímco Rughupathy [4] jí pojde podrobněji. Pro výpočet druhé derivace ve směru jednotkového vektoru \bar{v} používá vzorec:

$$I_{vv} = \bar{v}^T H \bar{v} \quad (30)$$

za \bar{v} dosad'me (n_x, n_y) :

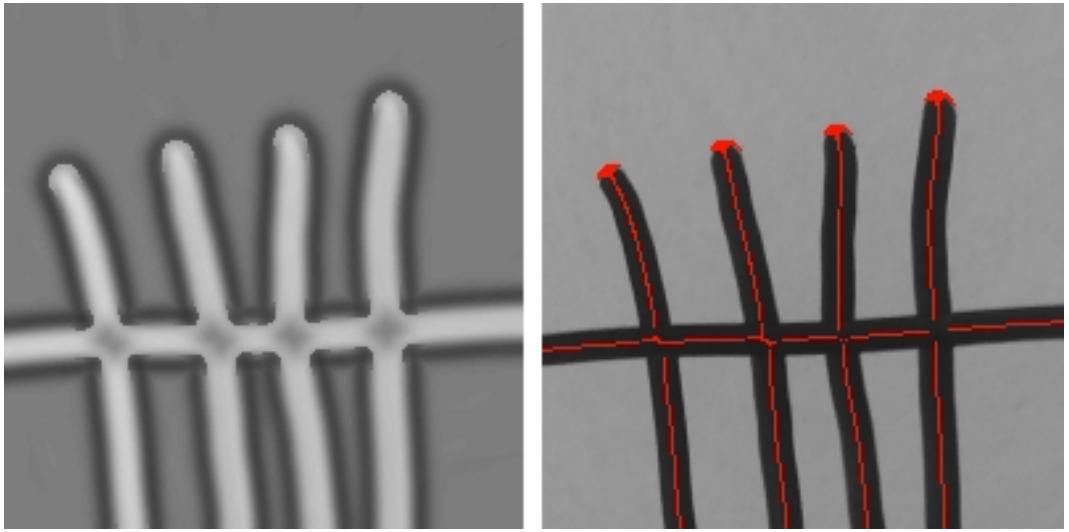
$$I_{nn} = \begin{bmatrix} n_x & n_y \end{bmatrix} H \begin{bmatrix} n_x \\ n_y \end{bmatrix} \quad (31)$$

Po roznásobení dostáváme:

$$I_{nn} = n_x^2 I_{xx} + 2n_x n_y I_{xy} + n_y^2 I_{yy} \quad (32)$$

Tato hodnota nám pomůže vyřešit problém s nechtěnými oblastmi A a C z předchozího oddílu. Obrázek č. 12 vlevo ukazuje, že I_{nn} nabývá velkých kladných hodnot na povrchu tmavých křivek a velkých záporných hodnot na povrchu světlých křivek. Zavedením jisté minimální hodnoty pro I_{nn} jsme schopni izolovat jen žádané pixely (viz obrázek č. 12 vpravo).

Povšimněte si také velkého množství rudých pixelů na koncích křivek. Lze se ho zbavit zvýšením minimální hodnoty I_{nn} , tím ovšem zanikají i body na křižovatkách, proto to není ideální. Namísto toho se jich zbavíme při trasování křivek v oddílu 2.3.



Obrázek č. 12: Graf I_{nn} vlevo ; žádané pixely vpravo

2.3 Spojování nalezených pixelů v křivky

Nyní zvolme počáteční bod S_0 ležící na křivce. Následující bod S_1 budeme volit tak, aby se jeho orientace co nejvíce podobala předchozímu bodu S_0 . Obdobným způsobem budeme hledat následující body S_{m+1} pro všechny S_m . Orientace značená θ se vypočte následovně:

$$\theta(x, y) = \arctan\left(\frac{-n_x}{n_y}\right) \quad (33)$$

Důležité je si povšimnout, že θ odpovídají dva různé úhly lišící se od sebe o 180° . Raghupathy toto řeší ve smyslu zavedení pravdivostní hodnoty pro $\theta(x_0, y_0)$, kterou $\theta(x_1, y_1)$ zdědí, pokud platí:

$$|\theta(x_0, y_0) - \theta(x_1, y_1)| < 90^\circ \quad (34)$$

Pokud podmínka (34) neplatí, $\theta(x_1, y_1)$ zdědí negaci té pravdivostní hodnoty, kterou nesla $\theta(x_0, y_0)$. Pro pravdivostní hodnotu 1 budeme rozumět orientaci mířící směrem do pravé poloviny a pravdivostní hodnotou 0 budeme rozumět orientaci mířící směrem do levé poloviny.

Takový postup v praxi znamená, že pokud bychom si křivku představili jako řeku, orientace všech jejich pixelů bude mít buď vždy po proudu, nebo vždy proti proudu.

Pokud se v okolí posledního nalezeného pixelu nevyskytuje žádný další, jehož rozdíl orientací $\Delta\theta$ by byl dostatečně malý, křivka je ukončena.

2.3.1 Zakomponování globálních vlastností

Raghupathy se zakomponováním globálních vlastností obrázku do algoritmu snaží vylepšit některé neduhy Stegerova řešení. Nejpříčivějším z nich je nesprávné následování křivek na křižovatkách. Za nesprávné považujeme takové, které se liší od způsobu, kterým by je vyhodnotil lidský pozorovatel.

Jelikož při replikaci fotografie Autopenem nemá pořadí kreslených křivek vliv na výslednou podobu repliky, budeme se zde zabývat jen myšlenkou, která za globálními vylepšeními stojí. Celé toto vylepšení přichází na scénu ve chvíli, kdy byla křivka ukončena kvůli nedostatku pixelů s podobnou orientací. Smyslem je se vydat směrem, kterým křivka mříila a hledat ve větší vzdálenosti od konečného pixelu. To nám umožní ukládání orientace posledních několika pixelů, ze kterých je buď obyčejným nebo váženým průměrem (nejbližší pixely mají větší váhu) vytvořena orientace reprezentující nedávný vývoj křivky. Podobným způsobem lze z těchto ukládaných hodnot dostat první derivaci směru křivky, díky které je možné lépe odhadnout, jak se vyvíjí křivky s velkým zakřivením.

V praktické části je těchto globálních vlastností využito jen v té nejúspornější formě a to tak, aby bylo možné spojit křivku, ve které vypadlo malé množství pixelů. Ale samotná křivka o kousek dál pokračuje.

V této oblasti je možné algoritmus nadále zlepšovat a vymýšlet nové způsoby, jak křivky spojovat tím nejfektivnějším způsobem, tak aby nakonec i pohyb pera Autopenu kopíroval pravděpodobný pohyb lidské ruky. To je ovšem nad rámec této práce.

2.3.2 Výběr počátečních bodů S_0

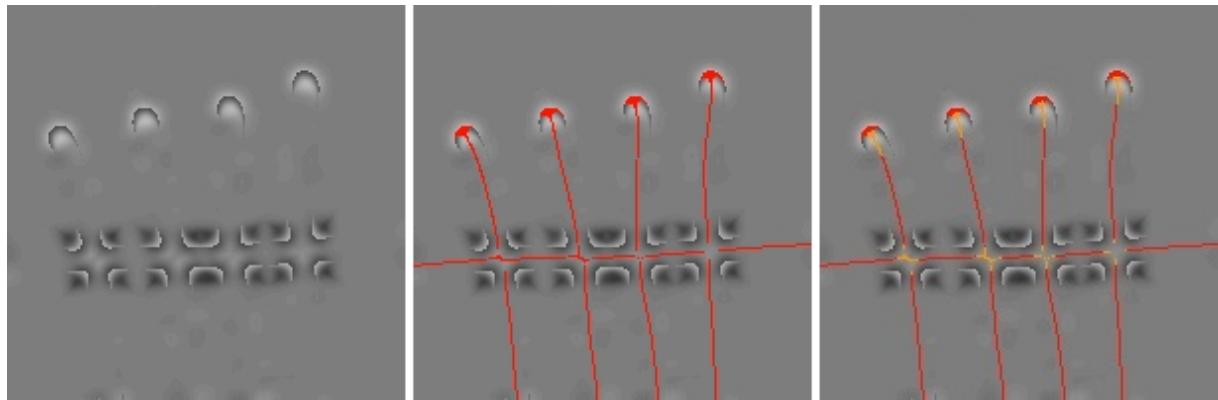
Steger i Raghupathy vybírali počáteční body manuálně. To ovšem není slučitelné s požadovanou automatizací, proto tato práce přichází s novým způsobem hledání vhodných počátečních bodů.

Po bodech S_0 bychom chtěli, aby se na každé části křivky mezi dvěma křižovatkami, nebo křižovatkou a přirozeným koncem křivky, nacházel alespoň jeden takový bod. Není žádoucí, aby těchto bodů bylo příliš mnoho. Mohlo by se stát, že po nalezení jedné křivky se ze zbylých bodů vedle ní spojí další paralelní křivka.

Zaved'me jednotkový vektor ve směru křivky $\bar{u} = (n_y, -n_x)$. Druhá derivace ve směru \bar{u} bude pak vypočtena obdobně jako v rovnici (31). Jelikož nás zajímá jen velikost této hodnoty, celý výraz vezmeme v absolutní hodnotě:

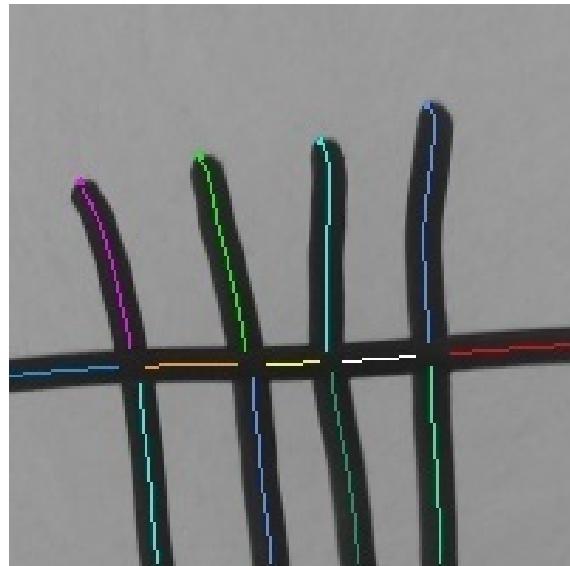
$$I_{uu} = |n_y^2 I_{xx} - 2n_x n_y I_{xy} + n_x^2 I_{yy}| \quad (35)$$

Pokud si představíme křivku po konvoluci s Gaussovou funkcí, intuitivní je, že druhá derivace ve směru křivky bude po celé její délce, s výjimkou konce a křižovatek, blízká nule. Na koncích a křižovatkách se druhá derivace bude od nuly zásadně lišit (viz obrázek č. 13 vlevo). Obrázek č. 13 uprostřed obsahuje pro orientaci pixely splňující podmínu (29). Pixely, které splňují podmínu (29) a zároveň se jejich I_{uu} dostatečně liší od nuly, označme počátečními body (viz pixely vybarvené oranžově na obrázku č. 13 vpravo).



Obrázek č. 13: Graf I_{uu} vlevo ; pixely splňující (29) uprostřed ; pixely S_0 vpravo

Výsledkem spojovacího algoritmu je pak obrázek č. 14, kde jsou jednotlivé pospojované křivky barevně odlišeny.



Obrázek č. 14: Výsledek spojovacího algoritmu

2.3.3 Pořadí křivek

Pořadí křivek záleží na pořadí bodů S_0 ve kterých tyto křivky začínají. Z praktického hlediska to znamená, že první nalezneme křivky s počátečními body u horního okraje fotografie, jelikož se tam při iteraci přes všechny pixely začínalo, kvůli standardům pro reprezentaci obrázků v OpenCV. Tento způsob je nevhodný, jelikož neoptimalizuje vzdálenost nutnou k přesunu pera mezi jednotlivými křivkami. K lepšímu seřazení by mohlo vést aplikování jedné z variací Problému obchodního cestujícího. V demonstrační aplikaci však není tohoto postupu užito. Mohl by se mu věnovat navazující výzkum.

2.4 Stanovení a provázání konstant

Díky tomu, že do jisté míry víme, jak bude vypadat fotografie, se kterou algoritmus bude pracovat, můžeme si dovolit napevno stanovit většinu konstant. Steger řešil obecný problém nalezení křivek, v této práci tomu tak není. Můžeme si totiž dovolit předpokládat, že křivky jsou oproti jejich pozadí výrazně tmavší, že jsou v rámci jedné fotografie stejně tlusté a stejně tmavé a že nevadí spojovat křivky tak, jak nebyly nakresleny.

Jediné dvě hodnoty, které musí být zadány manuálně, jsou σ a počet pixelů zkoumané fotografie. σ proto, že si nemůžeme být jisti, jak tlusté budou křivky v obrázku. Počet pixelů proto, aby bylo možné upravovat do jakého detailu má algoritmus křivky hledat. Fotografie je pak zmenšena na obdélník se stejným poměrem stran a obsahem odpovídajícím počtu zadaných pixelů.

Následující hodnoty byly určeny manuálním laděním:

Minimální hodnota I_{nn} pro pixel křivky ≈ 4.1

Minimální hodnota I_{uu} pro počáteční bod ≈ 0.3

Minimální délka jednoho úseku křivky $\approx 10 + \frac{\sqrt{\text{počet pixelů}}}{25}$

Maximální velikost úhlu $\Delta\theta \approx 40^\circ$

3 Android aplikace

Pro demonstraci metod představených v prvních dvou kapitolách je využito mobilní aplikace, což umožňuje rychlejší manipulaci s pořízenými fotografiemi, než by umožňoval program na PC. Výpočetní možnosti v tomto případě netrpí příliš, jelikož většina moderních mobilních zařízení disponuje 4GB RAM a procesory s frekvencí kolem 1.8GHz.

Samotná aplikace je napsaná v programovacím jazyku Java a využívá C++ knihovny. Výpočetní čas t závisí převážně na počtu prozkoumávaných pixelů:

$$t \approx 4 \cdot 10^{-5} \cdot \text{počet pixelů} \quad (36)$$

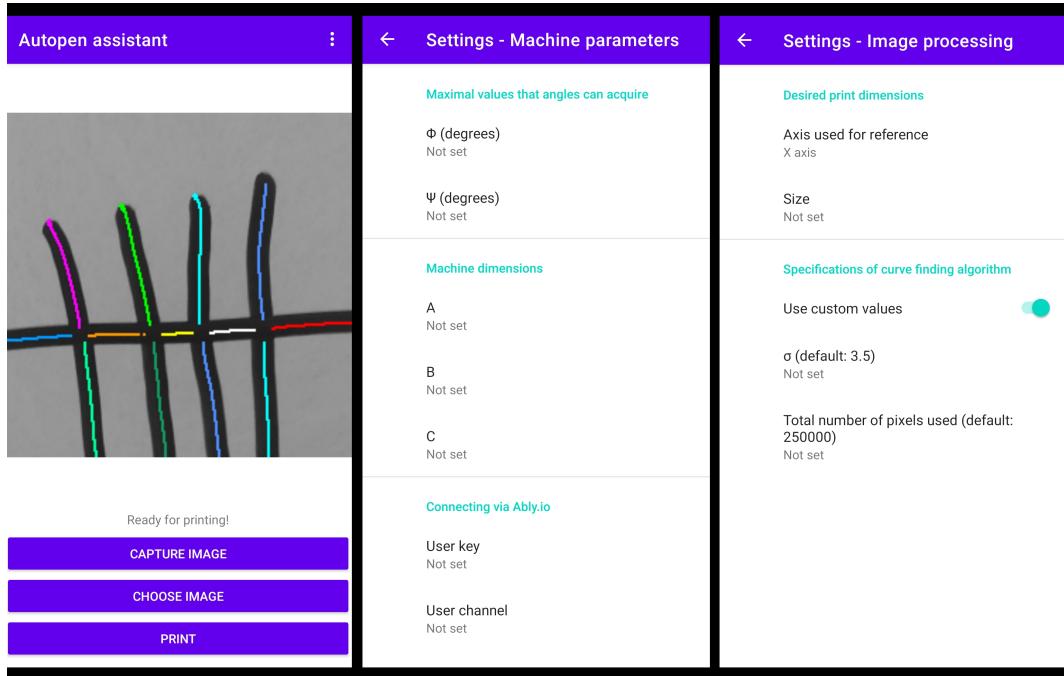
Tedy výpočet pro fotografii o rozměrech 500×500 pixelů zabere zhruba 10 vteřin. Pro urychlení by bylo možné napsat podstatnou většinu programu v nativním C++, nicméně aplikace slouží jen k demonstračním účelům, proto se nebudeme v této práci zabývat její optimalizací.

Vstupem aplikace jsou parametry Autopenu (a, b, c, ϕ, ψ), parametry Stegerova algoritmu (počet pixelů, σ) a požadovaná velikost kresby. Výstupem jsou příkazy pro Autopen (obdoba G-code, jehož je využíváno při 3D tisku) posílané přes peer to peer službu Ably do samotného zařízení.

3.1 GUI - Uživatelské prostředí

Skládá se z hlavní strany a dvou stran nastavení, jak je vidět na obrázku č. 15. Aktuální verze je navržena minimalisticky, následující verze by se mohly dočkat uživatelsky přívětivých vylepšení - například nástroje pro manipulaci s pořízenou fotografií, její oříznutí nebo manuální odstranění nechtěných křivek. (Např. pokud by podpis ležel na čtverečkovém papíře.)

Data z nastavení jsou ukládána pomocí tzv. Shared Preferences. Díky tomu není nutné při opětovném spuštění aplikace informace znova vyplňovat. Stejným způsobem by bylo možné v budoucích verzích implementovat ukládání již jednou zpracovaných fotografií, ke kterým by se dalo později vracet. Současná verze získaná data po odeslání maže.



Obrázek č. 15: Uživatelské prostředí

3.2 Odesílání dat přes Ably

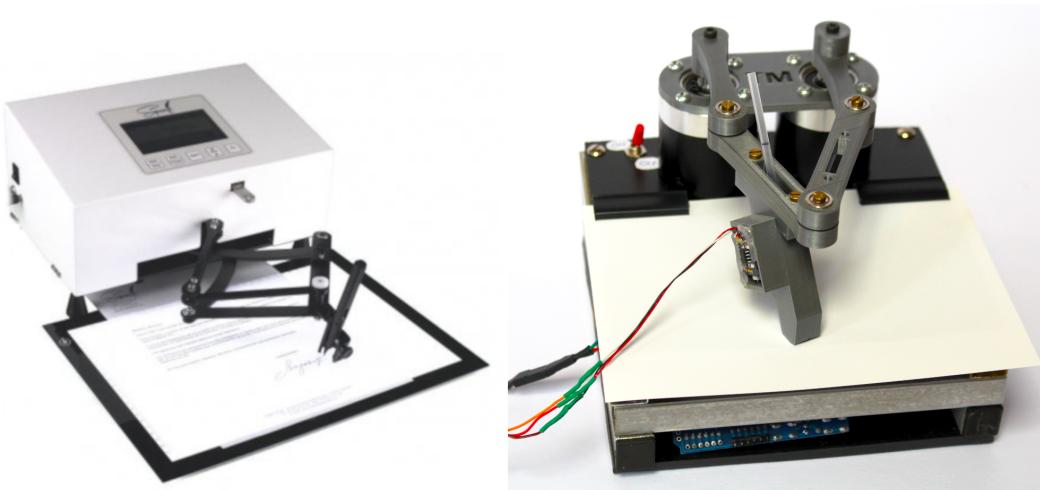
Ably je real time API, která zprostředkovává peer to peer spojení. Její hlavní předností je, že je takzvanou „cross platform“, tedy dokáže propojit programy napsané v rozličných programovacích jazycích. Této vlastnosti je využito v kapitole 4, kde komunikuje program napsaný v Pythonu s programem napsaným v Javě. Sekvence kódu pro odesílání a přijímání zpráv naleznete v přílohách pod číslem 3.

Zpráva je typu String a skládá se z příkazů: `up\n` pro zvednutí pera, `down\n` pro položení pera, `go\n126870\n117456\n` pro nastavení úhlu α na 126.87° a nastavení úhlu β na 117.456° . `\n` umožňuje zprávu číst po řádcích.

Maximální velikost zprávy je 16 kB, proto jsou větší soubory rozděleny do více zpráv, které jsou následně posílány v obráceném pořadí. To jest počínaje poslední, jelikož se při čtení zpráv jako první zobrazí zpráva, která dorazila naposledy.

4 Konstrukce demonstračního Autopenu

Na obrázku č. 16 vlevo je jeden z konvenčních přístrojů Autopen. Nejlevnější takový přístroj vás přijde na přibližně 45000 Kč, zatímco za kvalitnější je možné zaplatit částku až ve výši několika stovek tisíc. Autopen, na němž jsou v této části demonstrované výsledky prvních dvou kapitol, byl sestrojen ze součástek v hodnotě přibližně 3500 Kč (obrázek č. 16 vpravo). Je tedy třeba myslet na to, že nedokonalosti kopíí jsou převážně způsobeny nižší kvalitou konstrukce, nikoliv softwarem jako takovým.



Obrázek č. 16: Vlevo Autopen společnosti Signascript. Obrázek převzat z: www.signascript.com [Online]. 31.1.2021 ; Vpravo Autopen zkonstruovaný autorem

Pro jednoduchost jsou úhly α a β nastavovány motory, jejichž osy jsou umístěny přímo do bodů A a B . V úvahu by mohly připadat i motory umístěné mimo body A a B , ke kterým by poté byly připojeny pomocí rozvodových řemenů nebo podobných součástek umožňujících změnu převodu. Namísto toho, je zde k získání vyššího převodu využito planetových převodovek s poměrem 1:30.

Mozkem přístroje je Raspberry Pi Zero W, na kterém běží program v jazyku Python, jež naslouchá příkazům z android aplikace. Raspberry Pi je v tomto ohledu výrazně lepší volbou než většina desek Arduino, které postrádají výpočetní sílu a je k nim nutné doplnit spoustu dodatečných modulů, aby se mohly připojit na internet nebo operovat s větším množstvím dat.

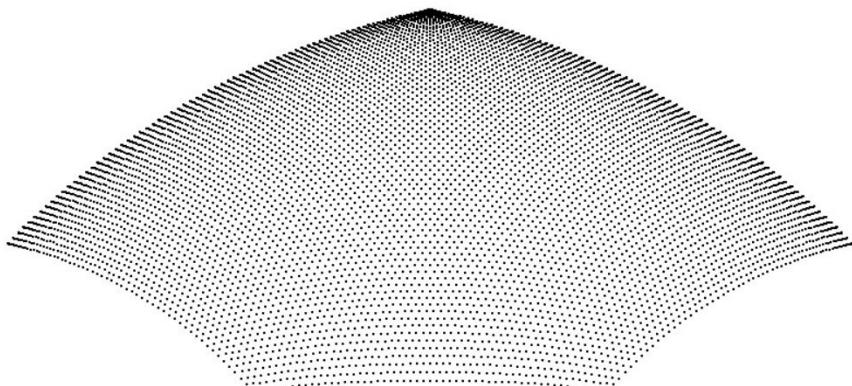
Pero je vysunováno lineárním servomotorkem, díky pružnosti v rámennou Autopenu je možné tímto servem určovat přítlač (této funkce není zatím využito). Pro autentičtější replikaci by bylo možné zvětšovat přítlač v oblastech většího zakřivení a naopak zmenšovat tam, kde křivka příliš nemění směr a kde se blíží ke svým koncům. Schéma zapojení je k nalezení v přílohách pod číslem

4.

4.1 Rozlišení kopie

Rozlišení výsledné kopie záleží na přesnosti nastavení úhlů α a β . Ta by pro motor Nema 17 s převodem 1:30 měla být 0.06° . Ve skutečnosti tomu tak není, planetové převodovky mají totiž vůli přibližně 0.20° . Přesnost by bylo možné zlepšit i o celý jeden řád použitím vyššího převodu a kvalitnější převodovky s menší nebo nulovou vůlí.

Přesnost nastavení úhlů není tou nejobjektivnější metodou pro určení rozlišení výsledné kopie, lepším kvantifikátorem by mohla být hustota bodů, po kterých se Autopen může pohybovat. Jak je vidět na obrázku č. 17, hustota těchto bodů se různí napříč pracovní obálkou. Zdá se, že je větší při horním okraji.

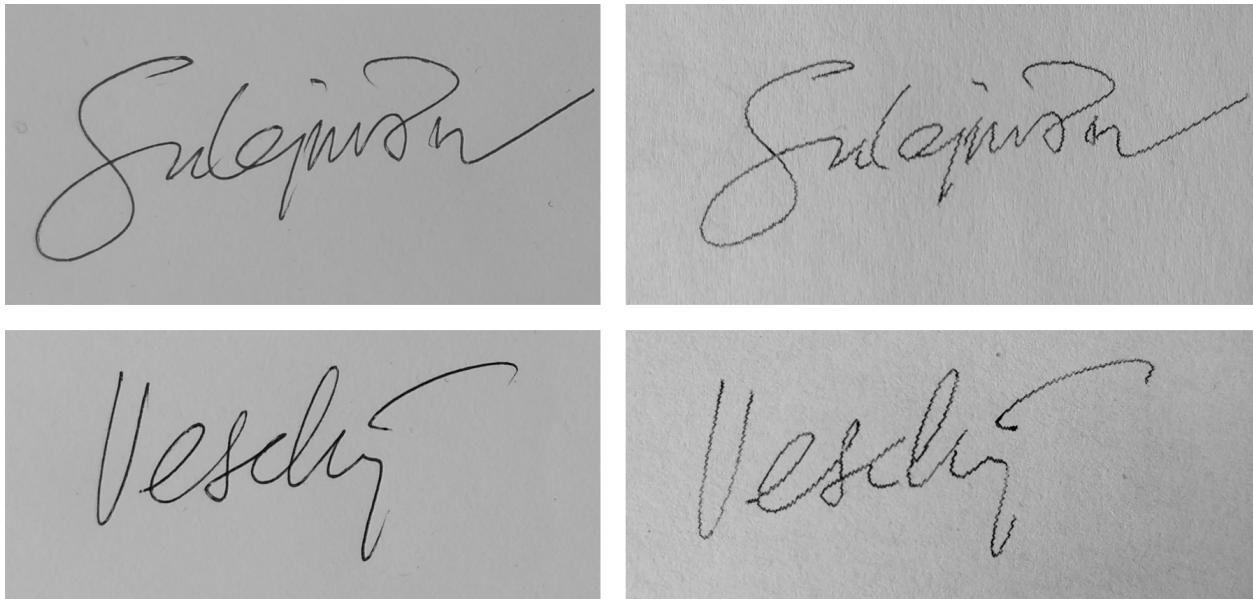


Obrázek č. 17: Body dosažitelné perem

Této problematice by se mohl věnovat další výzkum, má potenciál změnit strategii, kterou používáme při vybírání ideální oblasti pro kreslení.

4.2 Srovnání originálů a kopií

Na 5 centimetrů širokých kopiích jsou jasně zřetelné zoubky. Zvýšení přesnosti Autopenu v kombinaci s využitím pera zanechávajícího tlustší stopu by mohlo vést k ošálení zraku a spletení s originálem, nicméně pod mikroskopem by nejspíš stále byly rozeznat jemné variace v hustotě inkoustu.



Obrázek č. 18: Originál vlevo ; kopie vpravo

5 Padělání podpisů

Každý užitečný nástroj se v rukou nepřejícího člověka lehce mění ve zbraň. Proto je třeba zvážit možné implikace v oblasti práva, kde podpis slouží jako identifikátor.

Díky univerzálnosti této techniky je možné rychle kopírovat různé podpisy pořízené třeba z veřejných listin, žákovských knížek nebo jednoduše nalezených na internetu. Tak tomu doposud u společností vyrábějících Autopeny nebylo, zákazníci museli zpravidla zaslat své podpisy samotnému výrobci, který následně manuálně označil křivky ve fotografii a získaná data vrátil zákazníkovi na čipové kartě. Takový proces je zdlouhavý a neumožňuje replikovat mnoho různých podpisů naráz. Navíc by bylo možné při odhalení padělku propojit padělaný podpis s pachatelem skrze databázi společnosti, která Autopeny provozuje.

S rozvojem moderních výpočetních technologií může kdokoliv obejít tato úskalí pomocí aplikace podobné té z kapitoly 3, kterou by bylo možné upravit

cíleně tak, aby za sebou zahrazovala stopy. Například každý podpis unikátně deformaovala, aby neexistovaly dvě stejné kopie, nebo měnila přítlač během psaní, možností je nespočet. Nicméně není třeba podpisy vytvářet tak bezchybné, aby je nerozeznali odborníci, v mnohých případech by pachateli postačilo získat důvěru protistrany (např. Plná moc) pro to, aby mohl provést trestný čin dřív, než je možné podpis přezkoumat.

Pachatel by ovšem nebyl limitován jen na využití nákladného Autopenu, aplikace by se dala bez přílišné námahy modifikovat tak, aby produkovala G-code, tedy sekvenci příkazů pro 3D tiskárny. Zde by nebyl třeba přepočet na úhly a zvedání pera by se nahradilo pohybem na ose z . 3D tiskárny disponují dostatečnou přesností a jsou dostupné doslova každému. Pomocí izolepy, propisky a mobilní aplikace by bylo možné bez úprav softwaru nebo hardwaru přeměnit 3D tiskárnu na přístroj padělající podpisy.

S těmito podklady nelze jinak než dojít k závěru, že schopnost podpisu sloužit jako spolehlivá identifikační metoda je značně kompromitována rozvojem moderních technologií. Bylo by tedy záhadno přijít s alternativou k podpisu nebo ho opatřit dodatečným povinným identifikátorem, bez kterého by postrádal jakoukoliv kredibilitu. Navrhnut takovou dodatečnou ochranu ovšem není v kompetencích této práce, proto by bylo vhodné, kdyby se problematikou zabezpečení podpisu před moderními padělacími metodami zabývaly navazující výzkumy.

Závěr

Tato práce představila obecný geometrický model Autopenu a z něj vyplývající geometrické vlastnosti pracovní obálky, na jejichž základě lze vybrat rozměry ramen vhodné pro jednotlivé přístroje. Tím tato práce vyplnila vakuum, jež bylo způsobeno tajnústkářstvím komerčních výrobců Autopenů. Výpočetně úsporný numerický způsob byl následně využit k nalezení vhodného obdélníku uvnitř obálky, do nějž bude vložen kreslený útvar.

V druhé kapitole byl úspěšně automatizován Stegerův algoritmus tak, aby bylo nutné na vstupu poskytovat jen minimální množství informací. Kvůli automatizaci bylo třeba přijít se způsobem, jak nalézt vhodné počáteční body užívané ve spojovací části algoritmu. Poskytnuté řešení je poměrně efektivní a ve většině případů umožňuje umístit alespoň jeden počáteční bod na každý segment křivky.

Dosažené a nashromázděné poznatky byly demonstrovány sestrojením Autopenu a naprogramováním mobilní aplikace schopné ho ovládat. Autopenem vytvořené kopie byly velmi vyvedené v tom smyslu, že zachovávaly veškeré tvary a osobité křivky jednotlivých podpisů (ujištění o funkčnosti softwarové stránky). Ovšem jejich kvalita nebyla dostatečně přesvědčivá, jelikož se stroji nepodařilo dosáhnout kýzené přesnosti a zanechával tak „zubaté“ linie.

Aplikace by již v této formě mohla pomoci mnohým kutilům a bastlířům, kteří jsou schopni zkonztruovat Autopen, ale chybí jim schopnosti vytvořit potřebný software. Zároveň by mohla pomoci komerčním výrobcům Autopenů k tomu, aby si jejich zákazníci mohly extrahovat informaci z podpisů sami, a nemuseli to za ně zdlouhavě dělat oni výrobci, jak je tomu dnes.

Rozvoj takovýchto a podobných moderních technologií ovšem značně kompropituje bezpečnost podpisu jakožto identifikátoru, jelikož umožňují rychlé a nekontrolované množení podpisů. Proto jsem se rozhodl aplikaci neudělat veřejně přístupnou. Nicméně tím hrozba nemizí, jen se oddaluje. Budoucí výzkum by se tak mohl zabývat mimo jiné digitálními způsoby rozpoznávání Autopenem padělaných podpisů.

Bibliografie

- [1] AKRAM, Saba; ANN, Quarrat Ul. Newton raphson method. International Journal of Scientific & Engineering Research, 2015, 6.7: 1748-1752.
- [2] OVERTON, Michael. Quadratic Convergence of Newton's Method [online]. 2017 [cit. 13.1.2021]. Dostupné z New York University: <https://cs.nyu.edu/overton/NumericalComputing/newton.pdf>.
- [3] STEGER, Carsten. An unbiased detector of curvilinear structures, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998, vol. 20, no. 2, 113-125.
- [4] RAGHUPATHY, Karthik, 2004. Curve Tracing and Curve Detection in Images. New York. Diplomová práce. Cornell University.
- [5] VAIRALKAR, Manoj K.; NIMBHORKAR, S. U. Edge detection of images using Sobel operator. International Journal of Emerging Technology and Advanced Engineering, 2012, 2.1: 291-293.
- [6] PRESS, TEUKOLSKY, VETTERLING, FLANNERY, Numerical Recipes in C: The Art of Scientific Computing, Cambridge, Anglie: Cambridge University Press, 1992, 463-465.

Seznam obrázků

1	Geometrický model platformy Autopen - Vlastní tvorba	2
2	Náčrt oblasti nedostupné kvůli omezení maxima γ - Vlastní tvorba	4
3	Oblast vyhovující podmínkám (2)(3)(4) - Vlastní tvorba	4
4	Oblast vyhovující podmínkám (2)(3)(4) a (8)(9) - Vlastní tvorba	5
5	Závislost tvaru obálky na hodnotě c - Vlastní tvorba	6
6	Závislost tvaru obálky na hodnotě b - Vlastní tvorba	6
7	Ilustrace rovnice (12) - Vlastní tvorba	8
8	Graf funkce $h(x)$ s význačnými body M a N - Vlastní tvorba .	9
9	Ilustrace vstupu a výstupu Stegerova algoritmu - Foto autor & Vlastní tvorba	13
10	Grafy derivací Gaussovy funkce - Vlastní tvorba	14
11	Pixely splňující podmínu (28) - Vlastní tvorba	17
12	Graf I_{nn} vlevo ; žádané pixely vpravo - Vlastní tvorba	18
13	Graf I_{uu} vlevo ; pixely splňující (29) uprostřed ; pixely S_0 vpravo - Vlastní tvorba	20
14	Výsledek spojovacího algoritmu - Vlastní tvorba	20
15	Uživatelské prostředí - Vlastní tvorba	23
16	Vlevo Autopen společnosti Signascript. Obrázek převzat z: www.signascript.com (Online). 31.1.2021 ; Vpravo Autopen zkonstruovaný autorem - Foto autor	24
17	Body dosažitelné perem - Vlastní tvorba	25
18	Originál vlevo ; kopie vpravo - Foto autor	26

Přílohy

Příloha 1

$$L = \sqrt{a^2 + b^2 - 2ab \cos(\psi) - (x_n + \frac{c}{2})^2}$$

$$V = \sqrt{a^2 - (x_n - b \cos(\phi - 180^\circ) - \frac{c}{2})^2}$$

Pro $f_1(x_n) = p$

$$x_{n+1} = x_n + L \cdot \frac{L - p - 2kx_n}{x_n + \frac{c}{2} + 2kL}$$

Pro $f_1(x_n) = k_2$

$$x_{n+1} = x_n - \frac{L - V - 2kx_n - b \sin(\phi)}{\frac{x_n b \cos(\phi) - \frac{c}{2}}{V} - \frac{x_n + \frac{c}{2}}{L} - 2k}$$

Pro $f_1(x_n) = q$

$$x_{n+1} = x_n + L \cdot \frac{L - q - 2kx_n}{x_n + \frac{c}{2} + 2kL}$$

Příloha 2

I_x filtr

$$\begin{aligned} \iint_D f_2 dx dy &= \int_{b-\frac{1}{2}}^{b+\frac{1}{2}} \int_{a-\frac{1}{2}}^{a+\frac{1}{2}} f_2 dx dy = \int_{b-\frac{1}{2}}^{b+\frac{1}{2}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(a+0.5)^2+y^2}{2\sigma^2}} (1 - e^{\frac{a}{\sigma^2}}) dy = \\ &= \frac{1 - e^{\frac{a}{\sigma^2}}}{\sigma\sqrt{2\pi}} \int_{b-\frac{1}{2}}^{b+\frac{1}{2}} e^{-\frac{(a+0.5)^2+y^2}{2\sigma^2}} dy = \frac{1}{2} (1 - e^{\frac{a}{\sigma^2}}) e^{-\frac{(a+0.5)^2}{2\sigma^2}} \left(\operatorname{erf} \left(\frac{b+0.5}{\sigma\sqrt{2}} \right) - \operatorname{erf} \left(\frac{b-0.5}{\sigma\sqrt{2}} \right) \right) \end{aligned}$$

I_{xx} filtr

$$\begin{aligned} \iint_D f_3 dx dy &= \int_{b-\frac{1}{2}}^{b+\frac{1}{2}} \int_{a-\frac{1}{2}}^{a+\frac{1}{2}} f_3 dx dy = \int_{b-\frac{1}{2}}^{b+\frac{1}{2}} \frac{1}{\sigma^3\sqrt{2\pi}} e^{-\frac{(a+0.5)^2+y^2}{2\sigma^2}} (e^{\frac{a}{\sigma^2}}(a-0.5) - (a+0.5)) dy = \\ &= \frac{(e^{\frac{a}{\sigma^2}}(a-0.5) - (a+0.5))}{\sigma^3\sqrt{2\pi}} \int_{b-\frac{1}{2}}^{b+\frac{1}{2}} e^{-\frac{(a+0.5)^2+y^2}{2\sigma^2}} dy = \\ &= \frac{(e^{\frac{a}{\sigma^2}}(a-0.5) - (a+0.5))}{2\sigma^2} e^{-\frac{(a+0.5)^2}{2\sigma^2}} \left(\operatorname{erf} \left(\frac{b+0.5}{\sigma\sqrt{2}} \right) - \operatorname{erf} \left(\frac{b-0.5}{\sigma\sqrt{2}} \right) \right) \end{aligned}$$

I_{xy} filtr

$$\begin{aligned} \iint_D f_4 dx dy &= \int_{b-\frac{1}{2}}^{b+\frac{1}{2}} \int_{a-\frac{1}{2}}^{a+\frac{1}{2}} f_4 dx dy = \int_{b-\frac{1}{2}}^{b+\frac{1}{2}} y \frac{e^{\frac{a}{\sigma^2}} - 1}{\sigma^3\sqrt{2\pi}} e^{-\frac{(a+0.5)^2+y^2}{2\sigma^2}} dy = \\ &= \frac{(e^{\frac{a}{\sigma^2}} - 1)(e^{\frac{b}{\sigma^2}} - 1)}{\sigma\sqrt{2\pi}} e^{-\frac{(a+0.5)^2+(b+0.5)^2}{2\sigma^2}} \end{aligned}$$

Příloha 3

Publikování zpráv v jazyce Java

```
import io.ably.lib.realtime.AblyRealtime;
import io.ably.lib.realtime.Channel;
import io.ably.lib.types.AblyException;
import io.ably.lib.types.ClientOptions;
import io.ably.lib.types.Message;

String key;
String channel;
String messageData = "down\nngo\n90000\n90000\nup\n"

try {
    options = new ClientOptions(key);
    AblyRealtime ably = new AblyRealtime(options);
    Channel channel = ably.channels.get(channel);
    Message message = new Message("PrintingData", messageData);
    channel.publish(new Message[] {message});
    System.out.println("PUBLISHED");
} catch (AblyException e) {
    e.printStackTrace();
}
```

Čtení zpráv v jazyce Python

```
from io import StringIO as StringIO
from ably import AblyRest

key = ""
channel = ""
client = AblyRest(key)
channel = client.channels.get(channel)
paginatedResult = channel.history()

if len(paginatedResult.items) > 0:
    message = StringIO(paginatedResult.items[0].data)
    read = message.readline()
    if read == "up\n":
        pass
    elif read == "down\n":
        pass
    elif read == "go\n":
        alpha = message.readline()
        beta = message.readline()
```

Příloha 4

Schéma zapojení Autopenu

