VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MATEMATIKY
FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF MATHEMATICS

Aproximační plochy pro trojrozměrná data

APPROXIMATION SURFACES FOR 3D DATA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
ALŽBETA VALACHOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR
Mgr. JANA PROCHÁZKOVÁ, Ph.D.

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav matematiky
Studentka:	**Alžbeta Valachová**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Matematické inženýrství
Vedoucí práce:	**Mgr. Jana Procházková, Ph.D.**
Akademický rok:	2020/21

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Aproximační plochy pro trojrozměrná data

Stručná charakteristika problematiky úkolu:

Hlavní úkol práce bude studium a zpracování různých druhů aproximačních metod pro proložení prostorových dat. Student se seznámí se získáváním 3D dat (3D skenování) a různými algoritmy pro jejich zpracování a následnou vizualizaci. Některé algoritmy také implementuje a porovná výsledky.

Cíle bakalářské práce:

1. Seznámení se s principy 3D skenování a vytváření prostorových modelů.
2. Nastudování problematiky prokládání prostorových dat (metoda nejmenších čtverců, RANSAC, spline, apod.).
3. Programové zpracování vybraných algoritmů a jejich srovnání.

Seznam doporučené literatury:

FISCHLER, M. A. a R. C. BOLLES. Random sample consensus. Communications of the ACM. 1981, 24(6), 381-395. ISSN 0001-0782. Dostupné z: doi:10.1145/358669.358692

SCHNABEL, R., R. WAHL a R. KLEIN. Efficient RANSAC for Point-Cloud Shape Detection. Computer Graphics Forum. 2007, 26(2), 214-226. ISSN 0167-7055. Dostupné z: doi:10.1111/j.1467-8659.2007.01016.x

PIEGL, L. A. a W. TILLER. The NURBS book: with 334 figures in 578 parts. Berlin: Springer-Verlag, c1995. ISBN 3-540-55069-0.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2020/21

V Brně, dne

L. S.

prof. RNDr. Josef Šlapal, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Táto bakalárska práca sa zaoberá aproximáciou 3D dát plochami pomocou metódy RANSAC, metódy najmenších štvorcov a B-spline plochy. Jej cieľom je naštudovanie a spracovanie menovaných metód v programoch. Najprv sú metódy popísané a potom je vysvetlené a rozobraté ich programové spracovanie. V poslednej kapitole sú metódy porovnané na dátach získaných 3D skenovaním. Vďaka tomu dokážeme určiť ich výhody a nevýhody.

Abstract

This bachelor thesis is dealing with approximation surfaces for 3D data using methods such as RANSAC, least square method and B-spline surface. Its goal is to study and program these methods. First, the methods are described and then the actual programs are analysed. In the end of the thesis, we compare all three methods using data from 3D scanning. Through this effort we can assess their positives and negatives.

kľúčové slová

aproximácia, plochy, trojrozmerné dáta, 3D skenovanie, reverzné inžinierstvo, RANSAC, metóda najmenších štvorcov, B-spline krivky, B-spline plochy

keywords

approximation, surfaces, 3D data, 3D scan, reverse engineering, RANSAC, least square method, B-spline curves, B-spline surfaces

VALACHOVÁ, Alžbeta. *Aproximační plochy pro trojrozměrná data*, 45 s. Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/132356. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav matematiky. Vedoucí práce Jana Procházková.

Prehlasujem, že som bakalársku prácu *Aproximační plochy pro trojrozměrná data* vypracovala samostatne pod vedením Mgr. Jany Procházkovej, Ph.D. s použitím materiálov uvedených v zozname literatúry.

Alžbeta Valachová

Rada by som poďakovala svojej vedúcej bakalárskej práce Mgr. Jane Procházkovej, Ph.D. za odborné rady, pomocnú ruku a trpezlivosť.

Alžbeta Valachová

Obsah

Úvod

Slovo aproximácia pochádza z latinského *approximatus*, čo v preklade znamená *veľmi blízky, podobný*. Je významné v technických a vedeckých smeroch. A tak je to aj s aproximačnými plochami, ktorých použiteľnosť je rozsiahla. Táto práca je zameraná na konkrétne tri spôsoby aproximácie dát – metódu RANSAC, metódu najmenších štvorcov a B-spline plochu. V Teoretickej časti práce sú tieto metódy popísané a v časti Programové spracovanie a rozbor sa nachádza analýza vzniknutých programov a na záver práce metódy porovnáme na vstupoch z 3D skenu.

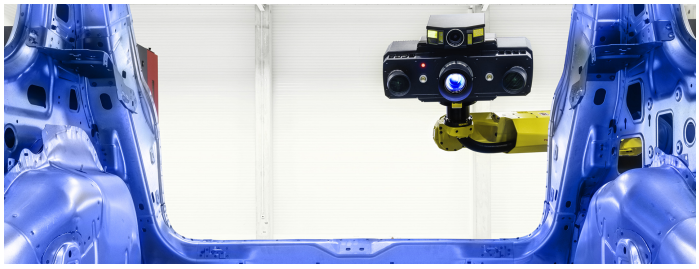
Najstaršia aproximácia zo spomínaných je metóda najmenších štvorcov, ktorej pôvod sa datuje do 19. storočia, kedy ju ako prvý popísal Francúz Adrien-Marie Legendre v roku 1805. Teraz je to významná a všeobecne známa metóda v štatistike. RANSAC sa stal veľmi dôležitým nástrojom v analýze obrazu už v osemdesiatych rokoch minulého storočia. Predtým, ako vznikol pojem počítačová grafika, okolo roku 1960 sa v automobilovom a leteckom priemysle používali B-spline krivky.

Jeden zo smerov, ktorý dokáže oceniť a využiť aproximačné plochy, je reverzné inžinierstvo. Reverzné inžinierstvo v strojárenstve je oblasť, ktorá sa zaoberá spätným rozstrojením modelu komponentu z nameraných dát. Namerať dáta môžeme rôznymi spôsobmi, napríklad 3D skenovaním. V tejto práci sme spomínané aproximačné metódy použili na reálne dáta, ktoré boli namerané 3D skenerom typu ATOS.

Reverzné inžinierstvo je uplatniteľné v mnohých oblastiach – skenujú a následne sa spätne modelujú sochy alebo historické budovy, v zubnom lekárstve sa vyrábajú ortézy alebo protézy pre pacientov, vytvárajú sa 3D modely pre novodobé videohry, kde hrajú skutoční herci. Pomocou ich pohybov sa následne vytvoria modely postáv v hre, ktoré sú jednoduchšie programovateľné a vyzerajú realisticky.

1 3D skenovanie

Nasledujúci text je čerpaný z [1, 2]. 3D skenovanie je optická metóda na získavanie dát. Skenovaním reálnych 3D objektov získame mračno bodov, s ktorým môžeme ďalej pracovať. ATOS je typ skenera, ktorý používa premietanie uzkopásmového modrého svetla na zmeranie mračna bodov objektu. Vpredu má dve kamery a projektor. Je vybavený technológiou Triple Scan, pri ktorej sa použijú kamery a projektor zároveň. Táto technológia vie účinne minimalozovať počet skenov a poskytovať kvalitnejšie dáta. Nasledujúci obr.1 pochádza zo zdroja [2].



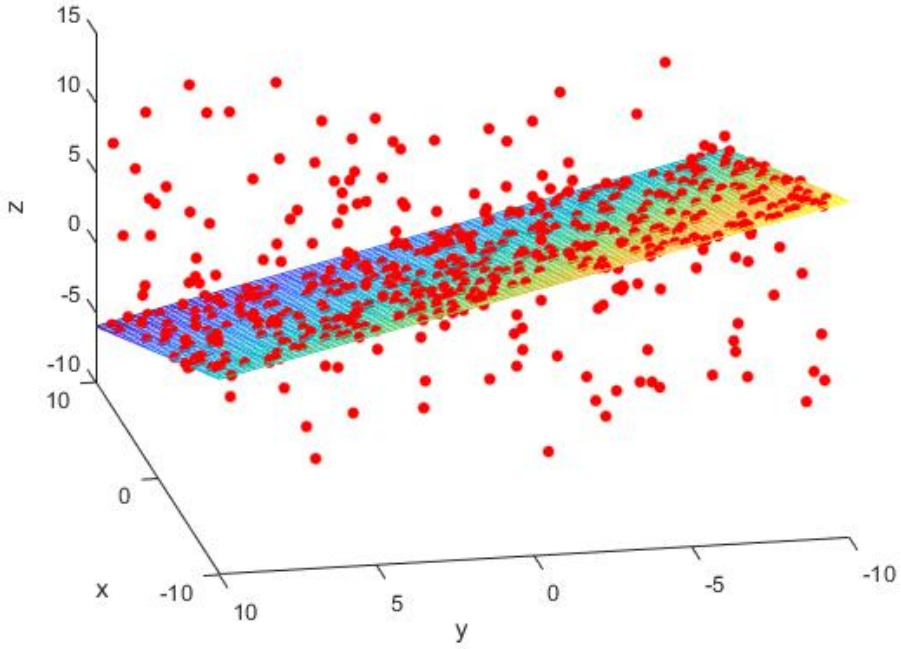
Obr. 1: ATOS

2 Teoretická časť

2.1 RANSAC

RANSAC je iteratívna metóda na odhadnutie matematického modelu podľa vstupných pozorovaných dát. Názov predstavuje skrátený výraz pre Random Sample Consesnus, v preklade zhoda náhodnej vzorky.

Túto metódu môžeme použiť na výpočet konkrétneho matematického modelu, ktorým budeme aproximovať dáta. Vyberieme zo všetkých dát minimálnu vzorku potrebnú na výpočet modelu a pomocou nej získame testovací model. V tejto práci sme sa zamerali na aproximačné plochy, tým pádom testovať optimálnosť modelu budeme na základe kritéria vzdialenosti. Body, ktoré sa nachádzajú nanajvýš v preddefinovanej hraničnej vzdialenosti, sa nazývajú *inliers* a body, ktoré sú za ňou, *outliers*. Výpočet modelu opakujeme, pokým počet *inliers* nedosiahne požadovanú hodnotu. [3, 4, 5]



Obr. 2: Metóda RANSAC

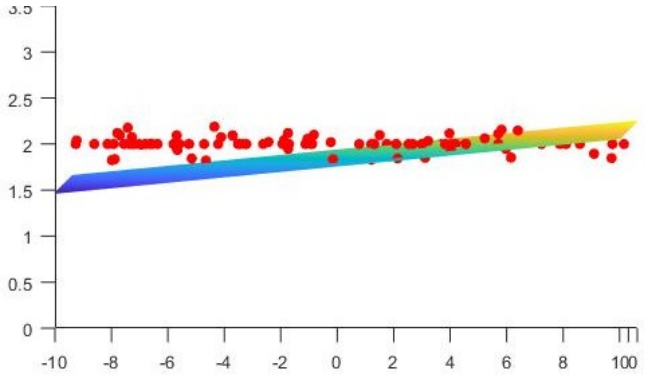
Body, ktoré sa na obr.2 nachádzajú mimo aproximačnú rovinu, sú *outliers*. Avšak môžu to byť aj presné hodnoty, ktoré sa nastavením preddefinovaných hodnôt nedostali do výberu. Môžu byť spôsobené napríklad šumom.

Je na nás, aké preddefinované hodnoty vzdialenosti a počtu *inliers* si zvolíme. Od ich zvolenia závisí presnosť aproximácie.

2.1.1 Iterácie

Pre úspešné nájdenie optimálneho riešenia musíme opakovane vyberať vzorku z počiatočných bodov. Dôležitá je voľba počtu opakovaní. Pokiaľ by sme napevno zadali veľký počet

iterácií, mohlo by sa stať, že program nájde vhodné riešenie, ale výpočet pokračuje. Pokiaľ by sme naopak zadali nízky počet, program nemusí nájsť vhodné riešenie.



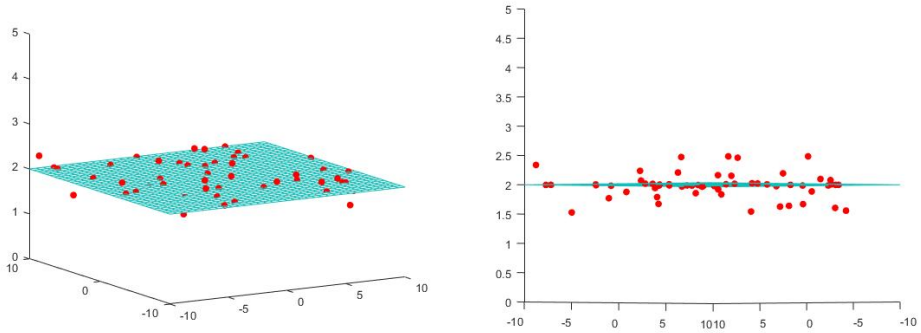
Obr. 3: Nesprávne vybratá rovina

Na obr.3 vybratá vzorka bodov nesprávne aproximuje zvyšné body. V tomto prípade by bolo treba zvýšiť počet iterácií. Aby sme sa zbavili tohto problému, na výpočet iterácií využijeme nasledujúci vzťah:

$$k = \frac{ln(1-p)}{ln(1-w^n)}, \tag{2.1}$$

kde k je počet iterácií, p je pravdepodobnosť úspešného riešenia – číslo z intervalu $< 0, 1 >$, ktorým nastavíme presnosť. Hodnota w predstavuje pravdepodobnosť vybratia *inlieru* zo vstupných bodov a n je minimálny počet dát na výpočet modelu.[3, 5]

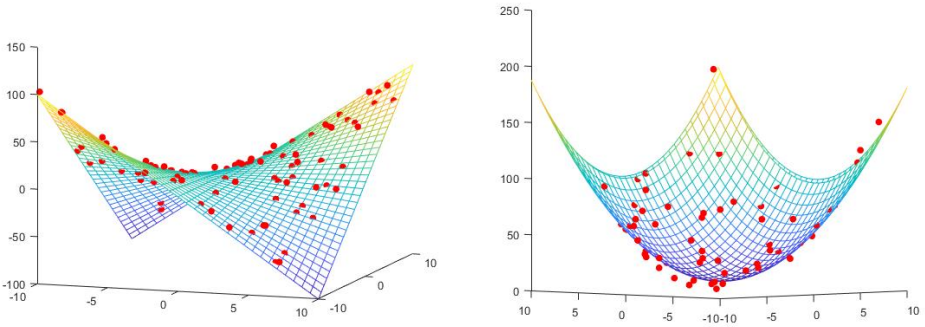
Na nasledujúcich obrázkoch je použitý RANSAC na aproximáciu 60 bodov rovinou, kde polovica bodov mala rovnakú súradnicu $z = 2$ a zvyšné boli náhodne posunuté o číslo z intervalu $< -0.5, 0.5 >$. Vybrali sme rovinu, ktorá spĺňala kritériá, a teda sa výsledná rovnica blížila k $z = 2$.



(a) (b)

Obr. 4: Rovina definovaná rovnicou $z = \frac{29.77}{14.89} \doteq 1.9993$

RANSAC môžeme použiť na akékoľvek geometrické plochy, ktoré vieme zadefinovať.



(a) hyperbolický paraboloid (b) eliptický paraboloid

Obr. 5: RANSAC použitý na nájdenie modelu kvadratickej plochy

Kritériá, ktoré sme použili na otestovanie vhodnosti riešenia boli založené na percentuálnom počte *inliers*. Pri rovine definovanej $ax + by + cz + d = 0$ je vzdialenosť bodu od roviny počítaná jednoduchou analytickou geometriou.

Veta 2.1. *Nech $X_0 = [x_0, y_0, z_0]$ je bod v euklidovskom priestore a nech α je rovina zadaná všeobecnou rovnicou $ax + by + cz + d = 0$. Potom pre vzdialenosť bodu od roviny platí:*

$$vzdial(X_0, \alpha) = \frac{|ax_0 + by_0 + cz_0 + d|}{\sqrt{a^2 + b^2 + c^2}} \tag{2.2}$$

Pri kvadratických plochách je výpočet vzdialenosti zložitejší. Na výpočet sme použili Lagrangeove multiplikátory.

2.1.2 Lagrangeove multiplikátory

Joseph-Louis Lagrange bol významný vedec, konkrétne v matematike hneď v niekoľkých odvetviach. Podaroval svoje meno rôznym vetám aj štruktúram. Zaviedol aj Lagrangeove multiplikátory, tiež známe pod pojmom Lagrangeove neurčité súčinitele. Slúžia na nájdenie viazaných lokálnych extrémov funkcie viacerých premenných. Nasledujúca teória je čerpaná zo zdrojov [6, 7].

Definícia 2.2. Nech $f = f(x_1, x_2, ..., x_n)$ je funkcia a $M \subseteq D(f)$ je nejaká neprázdna množina. Povieme, že f má v bode $x_0 \in M$ lokálne maximum, respektíve minimum, ak existuje také okolie $o(x_0)$, že pre všetky x platí: $f(x_0) \geq f(x)$, respektíve $f(x) \geq f(x_0)$.

Poznámka. Pokiaľ sú nerovnosti ostré, extrémy nazveme ostré.

Budeme uvažovať prípad, keď je množina M definovaná sústavou rovníc:

$$g_i(x_1, x_2, ..., x_n) = 0, \qquad i = 1, ..., m \tag{2.3}$$

V tomto prípade miesto lokálneho extrému vzhľadom k M budeme x_0 nazývať viazaným lokálnym extrémom.

Rovnice (2.3) nazývame väzbami. Úlohou Lagrangeových multiplikátorov je teda nájsť extrémy funkcie f s väzbami $g_i(x_1, x_2, ..., x_n) = 0$.

Lagrangeova metóda:

Nech funkcie $f(x_1, x_2, ..., x_n)$ a $g_i(x_1, x_2, ..., x_n), i = 1, ..., m$ majú spojité parciálne derivácie prvého rádu na nejakej otvorenej oblasti v $\mathbb{R}^n$. Nech f je viazaná podmienkou (2.3) pre všetky body množiny M a nech nadobúda svoj viazaný lokálny extrém na M v bode x_0. Potom existujú reálne čísla $\lambda_1, ..., \lambda_n$, pre ktoré platí:

$$\bigtriangledown f(x_0) = \lambda_i \bigtriangledown g_i(x_0), \qquad g_i(x_1, x_2, ..., x_n) = 0 \tag{2.4}$$

Čísla λ sú tzv. Lagrangeove multiplikátory.

Poznámka. Cieľom je zostrojiť Lagrangeovu funkciu:

$$L(x_1, ..., x_n, \lambda_1, ...\lambda_n) = f(x_1, ..., x_n) + \lambda_i g_i(x_1, ..., x_n), i = 1, ...m \tag{2.5}$$

a nájsť jej lokálne extrémy.

Definícia 2.3. Bod $x_0 \in M$, pre ktorý existujú Lagrangeove multiplikátory $\lambda_1, ..., \lambda_m$, tak, že platí (2.4), sa nazýva stacionárny bod funkcie f na M.

Poznámka. Nájdením lokálneho extrému funkcie L získame lokálny viazaný extrém funkcie f s podmienkou (2.3).

Lagrangeove multiplikátory použijeme na minimalizovanie vzdialenosti bodu $X = [x_0, y_0, z_0]$ od roviny spočítanej metódou RANSAC.

Máme funkciu kvadrátu vzdialenosti:

$$f(x, y, z) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 \tag{2.6}$$

a funkciu kvadratickej plochy, ktorá je jej väzbovou podmienkou:

$$ax^2 + by^2 + cx + dy + exy + f - z = 0 \tag{2.7}$$

Zostavíme Lagrangeovu funkciu:

$$L = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 + \lambda(ax^2 + by^2 + cx + dy + exy + f - z) \tag{2.8}$$

Spočítame jej parciálne derivácie podľa x, y, z, λ a položíme ich rovné 0:

$$\frac{\partial L}{\partial x} = 0, \quad \frac{\partial L}{\partial y} = 0, \quad \frac{\partial L}{\partial z} = 0, \quad \frac{\partial L}{\partial \lambda} = 0 \tag{2.9}$$

Riešením tohto systému rovníc získame lokálne extrémy funkcie L a tým pádom aj lokálne viazané extrémy rovnice (2.6) s podmienkou (2.7). Lokálny viazany extrém je teda miesto na kvadratickej ploche, ktoré je k danému bodu najbližšie alebo najďalej. A túto vlastnosť využijeme na výpočet vzdialenosti bodu od kvadratickej plochy.

2.2 Metóda najmenších štvorcov

Nasledujúce informácie sú čerpané zo zdroja [12]. Metóda najmenších štvorcov je aproximačná metóda, ktorá je významná v štatistike. Všeobecne sa používa na elimináciu chýb dát vzhľadom k nejakému kritériu.

Môže byť využitá napríklad na riešenie predurčených sústav rovníc.

Na začiatku máme n meraní pre m nezávislých premenných a závislej veličiny z. Pre vstupné merania hľadáme optimálny regresný model, ktorý ich bude aproximovať. Matematický model musíme mať pred použitím tejto metódy zvolený. V špeciálnom prípade metódy najmenších štvorcov môžeme pridať aj váhu týchto meraní, aby sme jednotlivým merianiam pridali príslušnú spoľahlivosť. Model, s ktorým v metóde pracujeme, predstavuje množinu podobných funkcií F, ktoré sú závislé na m vstupných premenných a líšia sa v parametroch označených $\bar{\beta}_i = (\beta_1, ...\beta_g)$. Tieto funkcie označíme nasledovne:

$$F = F(\bar{\beta}, x_1, ..., x_m) \tag{2.10}$$

Dáta aproximujeme funkciou, takže okrem ideálneho prípadu funkcia nebude prechádzať všetkými vstupnými dátami. Vždy budeme mať dáta, ktoré budú vzdialené od nej o odchýlku e_i. Pre optimálnosť riešenia požadujeme, aby odchýlky dát od modelovej funkcie boli minimálne. Preto musíme vybrať takú funkciu F, od ktorej majú vstupné dáta minimálnu odchýlku.

Veličinu z_i vyjadríme ako funkčnú hodnotu F posunutú o odchýlku e_i:

$$z_i = F(\bar{\beta}_i, x_1, ..., x_m) + e_i \tag{2.11}$$

Hľadáme minimálny kvadrát odchýlky, a to pre všetky dáta:

$$min(S(\bar{\beta}_i)^2) = min(\sum_{i=1}^{n} e_i^2) = min(\sum_{i=1}^{n} (z_i - F(\bar{\beta}_i, x_1, ..., x_m))^2) \tag{2.12}$$

Na minimalizáciu funkcie $S(\bar{\beta}_i)^2$ použijeme parciálne derivácie podľa $\beta_1, ...\beta_g$ a položíme ich rovné 0:

$$\begin{aligned} \frac{\partial S(\bar{\beta}_i)^2}{\partial \beta_1} &= 0 \\ \frac{\partial S(\bar{\beta}_i)^2}{\partial \beta_2} &= 0 \\ &\vdots \\ \frac{\partial S(\bar{\beta}_i)^2}{\partial \beta_g} &= 0 \end{aligned}$$

Riešením sústavy g rovníc o g neznámych získame hľadané koeficienty optimálneho riešenia.

V tejto práci sme sa zamerali na použitie metódy najmenších štvorcov pre rovinu definovanú všeobecnou rovnicou roviny a pre kvadratickú plochu. Konkrétny postup nájdeme v kapitole Programové spracovanie a rozbor na strane 24.

2.3 Bézierové krivky

Nasledujúca kapitola vychádza zo zdrojov [14]. Bézierove krivky získali svoj názov po francúzskom inžinierovi Pierre Bézierovi, ktorý ich ako prvý spomenul v publikácii *How*

Renault Uses Numerical Control for Car Body Design and Tooling v roku 1968. Bézierova krivka je aproximačná parametrická krivka určená riadiacim polynómom. Parameter je použitý na vymedzenie hodnoty premenných.

Definícia 2.4. Majme $p+1$ riadiacich bodov P_i v euklidovskom priestore alebo rovine. Definujeme bázovú funkciu $B_{i,p}(u)$ p-tého stupňa:

$$B_{i,p}(u) = \frac{p!}{i!(p-i)!}u^i(1-u)^{p-1}, \quad u \in <0,1> \tag{2.13}$$

Definícia 2.5. Majme $p+1$ riadiacich bodov P_i a nech $B_{i,p}(u)$ sú jednotlivé bázové funkcie. Potom Bézierovu krivku p-tého stupňa definujeme ako lineárnu kombináciu jednotlivých bázových funkcií a riadiacich bodov:

$$C(u) = \sum_{i=0}^{p} B_{i,p}(u)P_i, \quad u \in <0,1> \tag{2.14}$$

Poznámka. Bázová funkcia je tiež známa pod pojmom Bernsteinov polynóm alebo riadiaci polygón. V skutočnosti predstavuje klasický polynóm p-tého stupňa. Stupeň Bézierovej krivky bude vždy o 1 menší ako počet riadiacich bodov.

2.3.1 Lineárna Bézierova krivka

Lineárna Bézierova krivka je krivka prvého stupňa. Predstavuje úsečku, kde riadiace body P_0 a P_1 splynú s krajnými. Dosadením do (2.14) získame nasledujúcu parametrizáciu:

$$C(u) = (1-u)P_0 + uP_1, \quad u \in <0,1> \tag{2.15}$$

Obr. 6: Lineárna Bézierova krivka

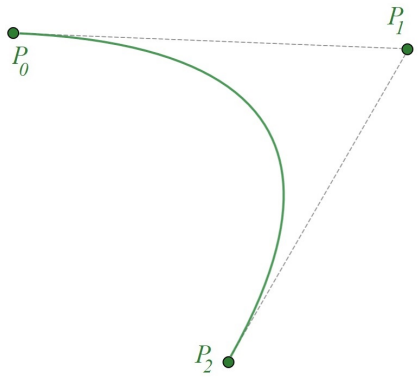
Podľa zvolenej hodnoty parametru u sa dostaneme na príslušné miesto na lineárnej funkcii. Dosadením do rovnice (2.13) získame 2 riadiace polygóny:

$$\begin{aligned} B_{0,1}(u) &= 1-u \\ B_{1,1}(u) &= u \end{aligned} \tag{2.16}$$

2.3.2 Kvadratická Bézierova krivka

Kvadratická Bézierova krivka je krivka druhého stupňa. Na vykreslenie potrebuje 3 riadiace body P_0, P_1 a P_2, ktoré predstavujú trojuholník, v ktorom Bézierova krivka vytvorí parabolu.
Krivku opäť získame dosadením do rovnice (2.14):

$$C(u) = (1-u)^2P_0 + 2u(1-u)P_1 + u^2P_2, \quad u \in <0,1> \tag{2.17}$$



Obr. 7: Kvadratická Bézierova krivka

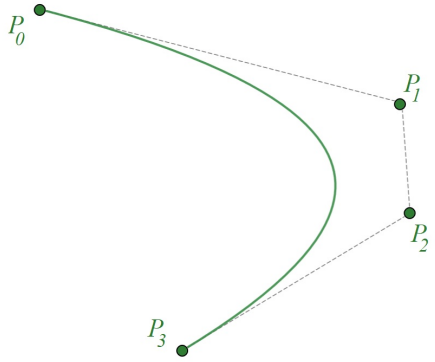
Riadiace polygóny dostaneme dosadením do (2.13):

$$\begin{aligned} B_{0,2}(u) &= (1-u)^2 \\ B_{1,2}(u) &= 2(1-u) \\ B_{2,2}(u) &= u^2 \end{aligned} \tag{2.18}$$

2.3.3 Kubická Bézierova krivka

Kubická Bézierova krivka je krivka tretieho stupňa. Začína v počiatočnom bode $P_0 = C(0)$ a ďalej pokračuje smerom k bodom P_1 a P_2, ktoré určujú jej tvar, ale nenachádzajú sa na nej. Končí na poslednom riadiacom bode $P_3 = C(3)$. Krivku opäť získame dosadením do rovnice (2.14):

$$C(u) = (1-u)^3 P_0 + 3u(1-u)^2 P_1 + 3u^2(1-u)P_2 + u^3 P_3, \quad u \in <0,1> \tag{2.19}$$



Obr. 8: Kubická Bézierova krivka

Riadiace polygóny dostaneme dosadením do (2.13):

$$\begin{aligned} B_{0,3}(u) &= (1-u)^3 \\ B_{1,3}(u) &= 3u(1-u)^2 \\ B_{2,3}(u) &= 3u^2(1-u) \\ B_{3,3}(u) &= u^3 \end{aligned} \tag{2.20}$$

2.3.4 Racionálne Bézierové krivky

Racionálne Bézierove krivky navyše uvažujú váhu vrcholov. Každý z vrcholov je definovaný v priestore štyrmi číslami, kde posledná hodnota predstavuje váhu. Racionálne Bézierove krivky sú vhodné na vykreslenie kuželosečiek.

Definícia 2.6. Nech p je stupeň krivky, $B_{i,p}(u)$ sú riadiace polygóny, P_i riadiace body a w_i nech je nezáporná váha vrcholu. Potom definujeme racionálne Bézierove krivky nasledovne:

$$C(u) = \frac{\sum_{i=1}^{p} B_{i,p}(u) w_i P_i}{\sum_{i=1}^{p} B_{i,p}(u) w_i}, \quad u \in <0,1> \tag{2.21}$$

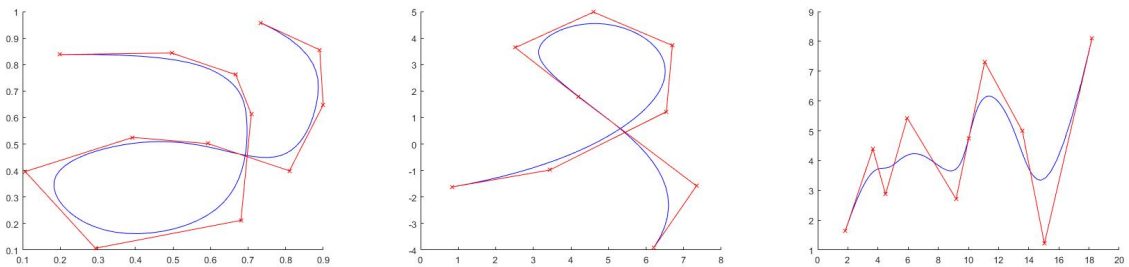
2.3.5 Geometrické vlastnosti Bézierovych kriviek

- **Konvexný obal:** Bézierova krivka sa nachádza v konvexnom obale tvorenom z jej riadiacich bodov P_0 až P_{p-1}.
- **Afinná invariantnosť Bézierovej krivky:** Afinné zobrazenie stačí použiť len na riadiace body Bézierovek krivky. Patrí sem napríklad posunutie a otočenie krivky.
- **Počiatočné a koncové body:** Bézierova krivka interpoluje koncové body.

2.4 B-spline krivky

Nasledujúca kapitola vychádza zo zdrojov [8, 10, 13, 14]. Na Bézierove krivky nadväzuje teória B-splinov. Názov spline predstavuje prúžok dreva alebo kovu, ktorý sme schopný ohýbať. B predstavuje slovo bázový, a teda B-spliny sú vlastne bázové spliny. Nevýhoda kriviek, ktoré sú definované jedným polynómom je, že prispôsobenie niektorým komplexným tvarom vyžaduje vysoký stupeň polynómu. Na druhú stranu, príliš vysoký stupeň negatívne ovplyvní stabilitu numerického výpočtu. To je jeden z dôvodov zavedenia B-splinu. B-spline krivka je zložená z viacerých Bézierov v jednotlivých úsekoch.

Existujú ďalšie krivky zložené z Bézierových, napríklad kubický Hermitov spline a taktiež Catmull-Romov spline. Podobne ako pri Bézierových krivkách, aj B-spliny môžeme rozdeliť na racionálne a neracionálne. Výhoda B-splinov je, že ich algoritmus je pomerne jednoduchý, rýchly a prípadne sa dajú ľahko lokálne upraviť. Sú teda výhodnejšie v praxi.



(a) (b) (c)

Obr. 9: B-spline krivky

2.4.1 Uzlový vektor

Ak by sme chceli napriamo rozdeliť B-spline krivku, bolo by to náročné. Je preto výhodnejšie, aby sme rozdelili jej oblasti. Ak je oblasť napríklad interval $< 0, 1 >$, rozdelíme ju na takzvané uzly $0 \leq u_0 \leq u_1 \leq ... \leq u_m \leq 1$. Táto neklesajúca postupnosť $U = \{u_0, u_1, ..., u_m\}$ sa nazýva uzlový vektor, jednotlivé u_i sú uzly a je to potrebný vstupný parameter na vytvorenie B-splinu. Keďže je to neklesajúca postupnosť, uzly u_i sa môžu opakovať. Maximálna násobnosť pre vnútorné uzly je p a pre koncové uzly $p + 1$, kde p je stupeň krivky.

2.4.2 Bázové funkcie

Definícia 2.7. Nech $U = \{u_0, u_1, ..., u_m\}$ je uzlový vektor a $u \in < u_0, u_m >$. Potom bázové funkcie stupňa p definujeme takto:

$$N_{i,0}(u) = \begin{cases} 1 & u \in < u_i, u_{i+1}) \\ 0 & inak \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{2.22}$$

Poznámka. Na definícii 2.7 si môžeme všimnúť, že $N_{i,p}(u)$ využíva rekurziu. $N_{i,p}(u)$ je nezáporná po častiach spojitá polynomiálna funkcia. Pri viacnásobných uzloch vznikajú zlomky $\frac{0}{0}$. Tento problém odstránime, ak ich položíme rovné nule.

Definícia 2.8. Majme $N_{i,p}$ bázovú funkciu stupňa p z definície 2.7, $n+1$ navzájom rôznych riadiacich bodov P_i, uzlový vektor $U = \{u_0, u_1, ..., u_m\}$ a číslo $u \in < u_0, u_m >$. Definujme B-spline krivku stupňa p ako lineárnu kombináciu bázových funkcii a riadiacich bodov:

$$C(u) = \sum_{i=0}^{p} N_{i,p}(u) P_i, \quad u \in < u_0, u_m > \tag{2.23}$$

Poznámka. Pre B-spline stupňa p s $n + 1$ riadiacimi bodmi bude mať uzlový vektor dĺžku $m + 1 = n + p + 2$ a bude mať tento tvar:

$$U = \{a_0, ..., a_p, u_{p+1}, ..., u_{m-p-1}, b_0, ..., b_p\} \tag{2.24}$$

Pomocou uzlového vektoru sa posúvame po jednotlivých sekvenciách B-splinu, ktoré sú definované dĺžkou uzlu – dĺžkou intervalu $< u_i, u_{i+1})$.

Poznámka. Pokiaľ platí, že $n = p$ a $U = \{0, ..., 0, 1, ..., 1\}$, potom bázové funkcie majú rovnaký tvar ako pri Bézierovej krivke a $C(u)$ je vlastne Bézier.

Definícia 2.9. Uzlový vektor sa nazýva uniformný, ak platí $u_{i+1} - u_i = konst$ pre $i \in < p + 1, m - p - 1 >$.

2.4.3 Násobné uzly

Pokiaľ parameter u nie je uzol v uzlovom vektore U, $C(u)$ v strede segmentu stupňa p je nekonečne diferencovateľná. Pokiaľ toto neplatí a u je uzol s násobnosťou k pri nenulovej bázovej funkcii, $C(u)$ je iba C^{p-k} spojitá na danom segmente.

Zmena spojitosti pri násobných uzloch nie je ich jediný dopad na B-spline. Ďalší dôsledok, ktorý je dôležitý pri výpočtových algoritmoch B-spline kriviek, je, že každý uzol s multiplicitou k zníži počet nenulových bázových funkcií v tomto uzle. Uzol s multiplicitou k má najviac $p - k + 1$ nenulových bázových funkcií. Pre uzol s násobnosť $k = p$ je iba jedna nenulová bázová funkcia.

2.4.4 Typy B-spline kriviek

B-spline krivky môžeme rozdeliť na viacero druhov. Nech $C(u)$ je B-spline krivka z definície (2.8).

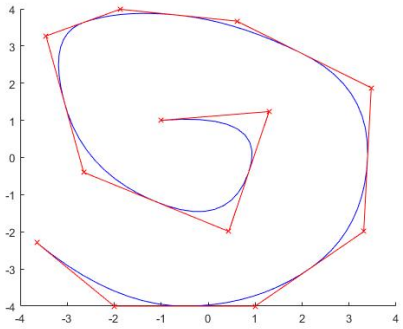
1. **Uzavreté:** Ak by sme chceli z $C(u)$ skonštruovať uzavretú B-spline krivku p-tého stupňa, jedným zo spôsobov je vziať prvých p bodov a zaradiť ich za posledný bod P_n. Uzlový vektor musí byť uniformný. Krivka bude potom začínať v bode P_0 a končiť v P_n.
2. **Vnorené**[1]**:** Na vytvorenie vnorenej B-spline krivky potrebujeme, aby prvý a posledný uzol v uzlovom vektore mali multiplicitu $p + 1$.
3. **Otvorené:** Otvorená B-spline krivka je taká, ktorej uzlový vektor nemá štruktúru násobnosti prvého a posledného uzlu ako pri vnorenej. Uniformnosť nie je nutná podmienka.

2.4.5 Vlastnosti B-spline kriviek

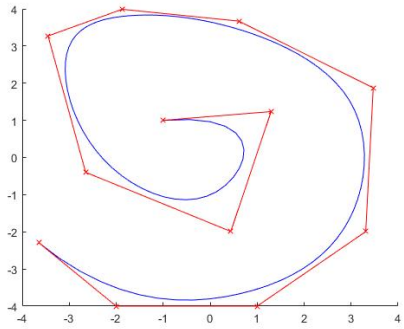
Najprv sa pozrieme na rozdiely medzi Bézierovými a B-spline krivkami. Zmenou aspoň jedného riadiaceho bodu v Bézierovej krivke dostaneme iný tvar krivky. Toto však neplatí pri B-spline – zmena i-tého riadiaceho bodu zmení $C(u)$ iba na intervale $< u_i, u_{i+p+1})$, kde p predstavuje stupeň krivky. Ďalším rozdielom je, že pri B-spline je stupeň krivky vstupným parametrom. Toto nám umožňuje, aby sme zostrojili B-splne krivku komplexného tvaru pri nízkom stupni, čo je pri Bézierovej krivke samotnej nemožné. Pokiaľ má uzlový vektor tvar $U = \{0, ..., 0, 1, ..., 1\}$ a platí $n = p$, tak sa B-spline zredukuje na Bézierovu krivku.

B-spline krivka teda zdedila vlastnosti Bézierovej, spomínané v predošlej kapitole, medzi nimi afinnú invariantnosť. Vďaka nej nám stačí použiť afinnú transformáciu na riadiace body a zostrojiť pre ne B-spline. Nemusíme transformovať samotnú krivku. Použitie B-spline kriviek v počítačovej grafike má mnoho výhod, avšak sú to polynomiálne krivky, a tie sa nedajú použiť na vykreslenie kruhov, elíps a podobných tvarov.
Na nasledujúcich obrázkoch sa nachádzajú B-spline krivky rôznych stupňov pri rovnakých riadiacich bodoch.

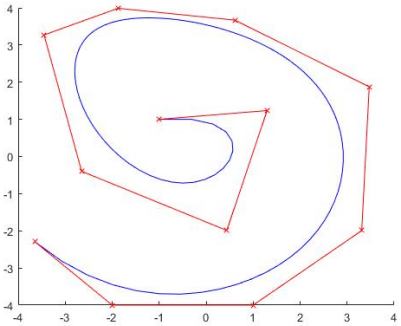
[1]Preložené z anglického *clamped* – upnutý,vnorený. V českom jazyku sa používa výraz *plovoucí*.



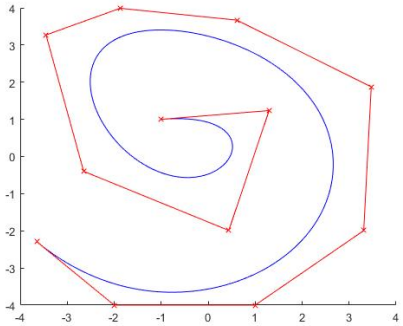
(a) stupeň 2



(b) stupeň 3



(a) stupeň 5



(b) stupeň 7

Obr. 11: B-spline krivky rôznych stupňov

2.5 B-spline plochy

Literatúra použitá v tejto kapitole je rovnaká ako v predošlej.

Definícia 2.10. Nech p a q sú stupne kriviek. Majme $m+1$ riadkov a v každom riadku $n+1$ riadiacich bodov $P_{i,j}$, kde $0 \leq i \leq m$ a $0 \leq j \leq n$ a uzlové vektory $U = \{u_0, ..., u_r\}$ $V = \{v_0, ..., v_s\}$ v nasledujúcom tvare:

$$\begin{aligned} U &= \{0, ..., 0, u_{p+1}, ..., u_{r-p-1}, 1, ..., 1\} \\ V &= \{0, ..., 0, v_{q+1}, ..., u_{s-q-1}, 1, ..., 1\}, \end{aligned} \tag{2.25}$$

(krajné uzly majú násobnosť $p+1$ v U a $q+1$ vo V).

Nech $\{u, v\}$ je dvojica čísel, pre ktoré platí: $u \in < u_0, u_r >$ a $v \in < v_0, v_s >$ a nech $N_{i,n}(u)$ a $N_{j,q}(v)$ sú bázové funkcie. Potom definujeme B-spline plochu takto:

$$S(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) P_{i,j} \tag{2.26}$$

U má $r+1$ uzlov a V má $s+1$ uzlov. Indexy teda musia spĺňať nasledovné rovnice:

$$\begin{aligned} r &= n + p + 1 \\ s &= m + q + 1 \end{aligned} \tag{2.27}$$

V časti Programové spracovanie a rozbor sa dozvieme, ako zostrojiť B-spline krivku. Zameriame sa konkrétne na De Boorov algoritmus, ktorý vychádza z riadiacich bodov, z nich spočíta ďalšie, až sa prepracuje k bodom, ktoré sa nachádzajú na hľadanej krivke.

2.5.1 Vlastnosti B-spline plôch

1. **Nezápornosť bázových funkcií**
   $\forall u, v, i, j, p, q : N_{i,p}(u)N_{j,q}(v) \geq 0$
2. **Bézierova plocha je špeciálny prípad B-spline plochy**
   pokiaľ $n = p$, $m = q$, $U = \{0, ..., 0, 1, ..., 1\}$ a $V = \{0, ..., 0, 1, ..., 1\}$, bázové funkcie sú Bernsteinove polynómy a výsledná plocha je Bézierova plocha.
3. **Nulový prípad súčinu bázových funkcií**
   Pokiaľ sa $\{u, v\}$ nachádza mimo štvorca $< u_i, u_{i+p+1} >$x$< v_j, v_{j+q+1} >$, súčin $N_{i,p}(u)N_{j,q}(v)$ je nulový.
4. **Afínna invariantnosť**
   Platí ako aj pri krivkách. Aplikovaním afinného zobrazenia na riadiace body aplikujeme afinné zobrazenie na celú plochu.
5. **Maximum plochy**
   Pri kladných stupňoch p,q B-spline plocha dosiahne práve jedno maximum.
6. **Konvexný obal**
   Pri dvojici $\{u, v\}$ nachádzajúcej sa vo vnútri štvorca $< u_{i^*}, u_{i^*+p+1} >$x$< v_{j^*}, v_{j^*+q+1} >$ platí, že plocha $S(u, v)$ je v konvexnom obale tvorenom riadiacimi bodmi $P_{i,j}$, kde $(i^* - p) \leq i \leq i^*$ a $(j^* - q) \leq j \leq j^*$.
7. **Rohové body**
   B-spline plocha interpoluje štyri rohové body: $S(0, 0) = P_{0,0}$, $S(1, 1) = P_{1,1}$, $S(1, 0) = P_{n,0}$, $S(0, 1) = P_{0,m}$.

3 Programové spracovanie a rozbor

3.1 RANSAC s modelom všeobecnej rovnice roviny

Nasledujúci algoritmus hľadá aproximačnú rovinu pre m bodov definovanú všeobecnou rovnicou. Máme náhodnú vzorku trojice bodov A_1, A_2, A_3 v priestore a hľadáme rovnicu v tvare:

$$ax + by + cz + d = 0 \tag{3.1}$$

Platí, že $\vec{w} = (a, b, c)$ je normálový vektor roviny. Najprv si spočítame zo vzorky bodov dva smerové vektory $\vec{u}, \vec{v}$.

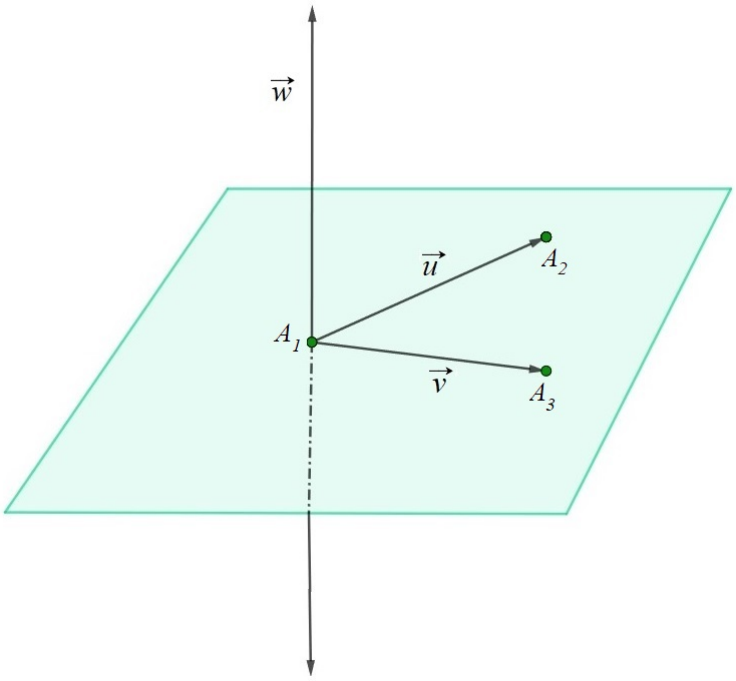
$$\begin{aligned} \vec{u} &= \overrightarrow{A_1A_2} = A_2 - A_1 \\ \vec{v} &= \overrightarrow{A_1A_3} = A_3 - A_1 \end{aligned} \tag{3.2}$$

Normálový vektor $\vec{w}$ získame aplikovaním vektorového súčinu na smerové vektory $\vec{u}$ a $\vec{v}$.

$$\vec{w} = \vec{u} \times \vec{v} \tag{3.3}$$

Koeficienty a, b, c dostaneme ako prvky normálového vektoru $\vec{w}$:

$$a = w(1), \quad b = w(2), \quad c = w(3) \tag{3.4}$$



Obr. 12: Znázornenie vektorov u, v, w testovacej roviny

Zostáva dopočítať štvrtý koeficient d. Do rovnice (3.1) dosadíme koeficienty a, b, c a bod $A_1 = [x_1, y_1, z_1]$. Potom si z nej vyjadríme d:

$$d = -ax_1 + by_1 - cz_1 \tag{3.5}$$

Teraz už máme všeobecnú rovnicu roviny vypočítanú zo vzorky dát. Overíme jej vhodnosť dosadením každého bodu $A_i = [x_i, y_i, z_i]$ do rovnice (2.2):

$$vzdial(A_i, \alpha) = \frac{|ax_i + by_i + cz_i + d|}{\sqrt{a^2 + b^2 + c^2}}, i = 1, ..., m \tag{3.6}$$

Spočítame percento *inliers*. Pokiaľ je toto percento väčšie ako hraničná hodnota percenta, rovina je vyhovujúca a výpočet môžeme ukončiť. Ak nie, zopakujeme výpočet pre inú vzorku náhodných bodov.

Vstupné hodnoty do funkcie:

A	matica, ktorej jednotlivé riadky predstavujú vstupné body
hran_v	hraničná vzdialenosť bodov od roviny, pre ktorú považujeme body za *inliers*
hran_p	minimálne percento *inliers*
p	pravdepodobosť nájdenia úspešného riešenia
w	pomer *inliers* k celkovému počtu bodov

Funkcia v MATLABE:

```
function [H]=ransac2(A,hran_v,hran_p,p,w)
  % nastaveníme parametre na začiatku 0
  it=0; a=0; b=0; c=0; d=0; perc=0;
  m=size(A);   % velkost matice bodov
  points=zeros(3,m(2));

  k=log(1-p)/log(1-w^3); % výpočet potrebných iterácií

  % cyklus sa zastaví pri presiahnutom počte iterácií alebo pri nájdení
    optimálneho riešenia

  while (perc<hran_p) && (it<k)
    % v každej iterácií vynulujeme premenné
    vzdial=0; pocetbodov=0; perc=0;

    t=randperm(m(1),m(2)); % náhodné indexy, podla ktorých vyberieme
    vstupnú vzorku
    for i=1:3
      points(i,:)=A(t(i),:);  % do každého riadku matice points vložíme
    postupne náhodné body
    end

    % výpočet koeficientov pomocou normálového vektoru roviny
    u=points((2),:)-points((1),:); % smerovy vektor u
    v=points((3),:)-points((1),:); % smerovy vektor v
    w=cross(u,v); % vektorový súčin, w je normálový vektor roviny

    % koeficienty roviny a,b,c sú zložky jej normálového vektoru
    a=w(1); b=w(2); c=w(3);
    % d vypočítame dosadením prvého bodu a koeficientov a,b,c do
    všeobecnej rovnice
    d=-a*points((1),1)-b*points((1),2)-c*points((1),3);

    % overíme optimálnost roviny na základe vzdialenosti
    for i=1:m(1)
      bod=A(i,:); % do premennej si uložíme vždy i-ty bod
      % výpočet vzdialenosti bodu od roviny
      vzdial=abs(a*bod(1)+b*bod(2)+c*bod(3)+d)/sqrt(a^2+b^2+c^2);

      % overím, či sa bod nachádza pod hraničnou vzdialenostou
      if vzdial<=hran_v
        pocetbodov=pocetbodov+1; % pripočítam bod
      end
    end

```

```
42      perc=pocetbodov/m(1)*100; % počet percent bodov, ktoré sú pod
        hraničnou vzdialenostou
43      it=it+1; % počítam iterácie
44    end
45    % koeficienty optimálnej roviny uložíme do matice H ako výstup funkcie
46    H=[a,b,c,d];
47  end
```

Majme 100 bodov, z ktorých 60 má z-tovú súradnicu 2.0 a zostávajúce ju majú posunutú náhodne o číslo z intervalu $< -0.2, 0.2 >$ a ďalšie vstupy:

Počet bodov	w	p	Hraničná vzdialenosť	Hraničné percento
100	0.6	0.85	0.1	80

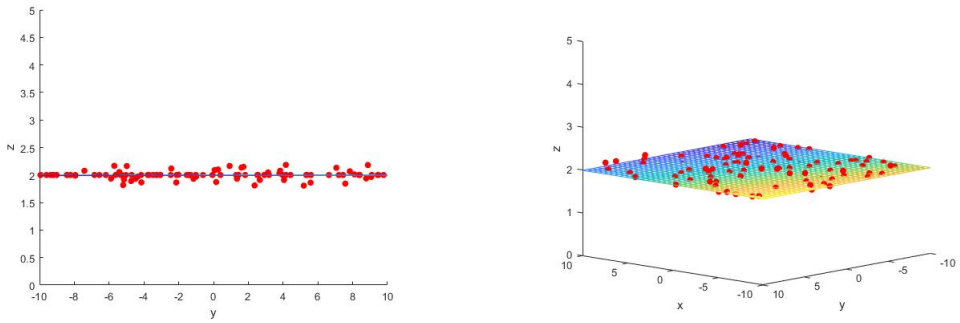
Program našiel rovinu, od ktorej je 81% bodov vzdialených nanajvýš 0.1. Označme túto rovinu α. Rovina α je na základe vstupných kritérií vyhovujúca a má rovnicu:

$$0.12x - 0.04y + 150.85z - 301.25 = 0 \tag{3.7}$$

Po úpravách dostávame rovnicu, ktorá sa podobá na $z = 2.0$.

$$7.96 \cdot 10^{-4}x - 2.65 \cdot 10^{-4}y + z = 2.001 \tag{3.8}$$

Rovina α:



(a) Pohľad na α v rovine yz (b) α v 3D pohľade

Obr. 13: Metóda RANSAC použitá na výpočet roviny

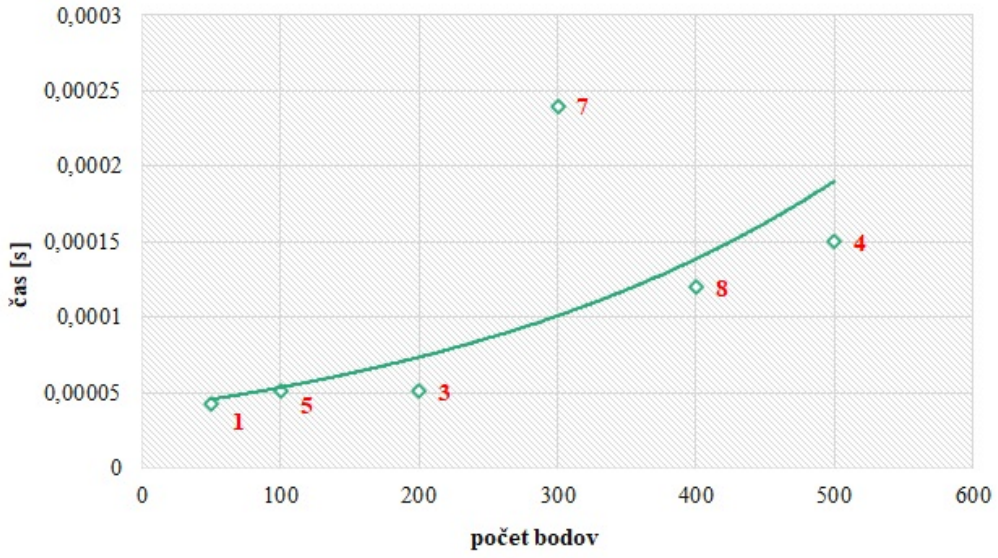
Ďalej sme pokračovali testovaním rýchlosti programu pri zmenách vstupných hodnôt a presností. Testy boli vykonávané v programe MATLAB s verziou R2020a na počítači s procesorom Intel Core i7 7700HQ s frekvenciou 2.8GHz a ôsmimi jadrami. Operačná pamäť tohto počítača bola 8GB.

3.1.1 Časové závislosti

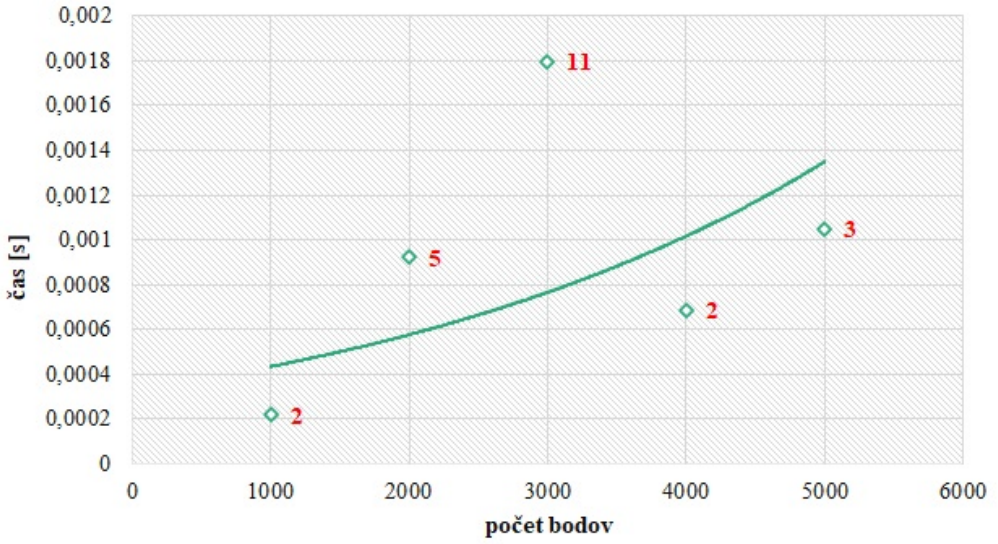
Na každom grafe sú červenou vyznačené počty iterácií pri jednotlivých bodoch grafu. Podľa toho môžeme lepšie pozorovať náročnosť výpočtu.

Počet vstupných bodov

Nasledujúce grafy znázorňujú priebeh časovej závislosti na počte vstupných bodov. Červené čísla pri bodoch sú počty iterácií, ktoré musel program vykonať, aby našiel riešenie.



Obr. 14: Graf závislosti času na počte vstupných bodov v stovkách

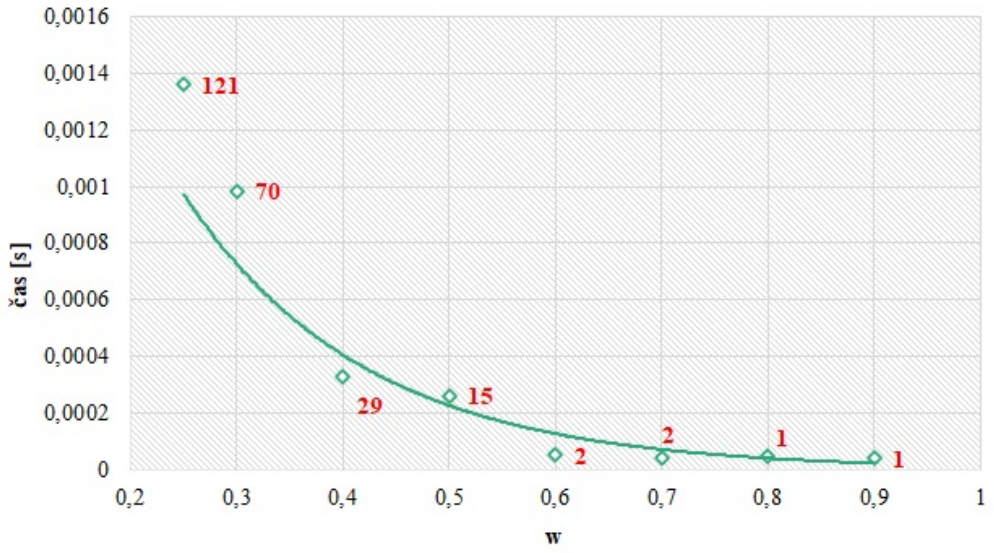


Obr. 15: Graf závislosti času na počte vstupných bodov v tisícoch

Príliš veľa bodov dokáže zahltiť procesor a oddialiť nájdenie výsledku. Avšak, čím viac bodov máme, tým je väčšia šanca na to, aby sme sa trafili do tých neposunutých. Výpočet je jednoduchý, preto ho počítač zvládne za krátku dobu aj pri väčšom množstve bodov.

w – pomer *inliers* k celkovému počtu bodov

Ako bolo zmienené v teoretickej časti RANSAC, w predstavuje pomer *inliers* k celkovému počtu bodov. Je to číslo z intervalu $< 0, 1 >$.

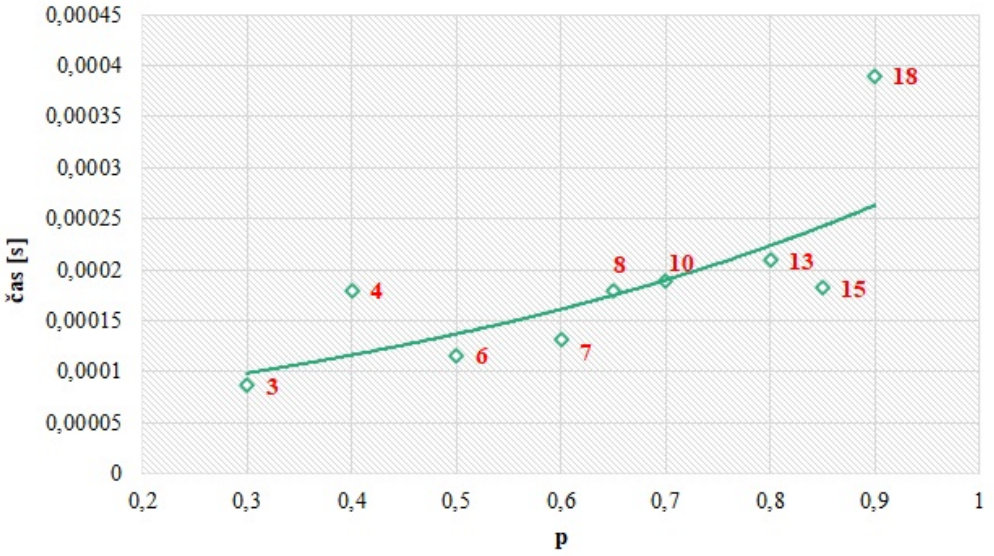


Obr. 16: Graf závislosti času na w – pomer *inliers* k celkovému počtu bodov

Čím bude počet *inliers* vyšší vzhľadom k celkovému počtu bodov, tým bude hľadanie vhodnej roviny zaberať menej času, a teda bude stačiť nižší počet iterácií.

p – pravdepodobnosť nájdenia úspešného riešenia

Vstupná hodnota $p \in < 0, 1 >$ predstavuje pravdepodobnosť nájdenia úspešného riešenia. Počet iterácií k z rovnice (2.1) sa so stúpajúcou hodnotou p zvyšuje.



Obr. 17: Graf závislosti času na p – pravdepodobnosť nájdenia úspešného riešenia

Pri zvyšovaní p sa zvýši počet iterácií, aby bol program účinnejší v hľadaní vyhovujúcej roviny. Pri príliš nízkom p sa môže stať, že počet iterácií na predpokladané nájdenie riešenia je nedostatočný, výpočet sa ukončí, ale riešenie sa nenájde. V tomto prípade program počíta iba pár iterácií, ale bez výsledku. Je dobré voliť vyššiu hodnotu p, aby bolo zaručené nájdenie optimálneho riešenia.

3.2 RANSAC s modelom kvadratickej plochy

S využitím znalostí Lagrangeových multiplikátorov sa môžeme posunúť na hľadanie optimálnej kvadratickej plochy.
Lagrangeove multiplikátory použijeme na minimalizovanie vzdialenosti bodu $X = [x_0, y_0, z_0]$ od plochy spočítanej metódou RANSAC. Zostavili sme Lagrangeovu funkciu a spravili parciálne derivácie podľa jednotlivých premenných, vrátane Lagrangeovho multiplikátoru λ.
Máme funkciu kvadrátu vzdialenosti:

$$f(x, y, z) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 \tag{3.9}$$

a funkciu kvadratickej plochy, ktorá je jej väzbovou podmienkou:

$$ax^2 + by^2 + cx + dy + exy + f - z = 0 \tag{3.10}$$

Zostavíme Lagrangeovu funkciu:

$$L = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 + \lambda(ax^2 + by^2 + cx + dy + exy + f - z) \tag{3.11}$$

Spočítame jej parciálne derivácie podľa x, y, z, λ a položíme ich rovné 0:

$$\frac{\partial L}{\partial x} = 0, \quad \frac{\partial L}{\partial y} = 0, \quad \frac{\partial L}{\partial z} = 0, \quad \frac{\partial L}{\partial \lambda} = 0 \tag{3.12}$$

Riešením tohto systému rovníc získame lokálne extrémy funkcie L a tým pádom aj lokálne viazané extrémy rovnice (3.9) s podmienkou (3.10).
Po vyriešení rovníc vyberieme z lokálnych viazaných extrémov minimum, a to je výstup našej funkcie.

Vstupné hodnoty do funkcie:

$[x_0, y_0, z_0]$	súradnice bodu, ktorého vzdialenosť počítame
G	vektor koeficientov plochy

Funkcia naprogramovaná v MATLABE vyzerá nasledovne:

```
function [dmin]=lagrangemult(x0,y0,z0,G)

  syms x y z lambda
  a=G(1); b=G(2); c=G(3); d=G(4); e=G(5); f=G(6); % koeficienty z matice
    G uložíme do premenných

  L = (x-x0)^2 + (y-y0)^2 + (z-z0)^2 + lambda*(a*x^2 + b*y^2 + c*x + d*y
    + e*x*y + f - z); %lagrangeova funkcia
```

```
7
8   % parciálne derivácie L podla premenných
9   dL_dx       = diff(L,x)          == 0;
10  dL_dy       = diff(L,y)          == 0;
11  dL_dz       = diff(L,z)          == 0;
12  dL_dlambda  = diff(L,lambda)     == 0;
13
14  % systém 4 rovnic
15  system = [dL_dx, dL_dy, dL_dz, dL_dlambda];
16
17  % výpočet cez vpasolve
18  [xs, ys, zs, ls] = vpasolve(system, [x, y, z, lambda], [-Inf Inf;-Inf
     Inf;-Inf Inf;-Inf Inf;]);
19
20  % počet výsledkov
21  numberOfSolutions = size(xs, 1);
22
23  % hladanie minimálneho kvadrátu vzdialenosti
24  dmin=(xs(1)-x0)^2 + (ys(1)-y0)^2 + (zs(1)-z0)^2;
25
26  for  i = 2:numberOfSolutions % vyberáme minimálny kvadrát zo všetkých
     riešení systému rovníc
27    d(i)=(xs(i)-x0)^2 + (ys(i)-y0)^2 + (zs(i)-z0)^2;
28    if d(i)<dmin
29      dmin=d(i);
30    end
31  end
32 end
```

Minimálnu vzdialenosť bodu od kvadratickej plochy už vypočítať dokážeme, môžeme prejsť na samotný RANSAC. V tomto prípade hľadáme plochu charakterizovanú rovnicou $ax^2 + by^2 + cx + dy + exy + f - z = 0$. Na výpočet koeficientov a až f potrebujeme 6 bodov $A_i = [x_i, y_i, z_i]$, kde $i = 1, ..., 6$. Maticový prepis bude:

$$\begin{bmatrix} x_1^2 & y_1^2 & x_1 & y_1 & x_1y_1 & 1 \\ x_2^2 & y_2^2 & x_2 & y_2 & x_2y_2 & 1 \\ x_3^2 & y_3^2 & x_3 & y_3 & x_3y_3 & 1 \\ x_4^2 & y_4^2 & x_4 & y_4 & x_4y_4 & 1 \\ x_5^2 & y_5^2 & x_5 & y_5 & x_5y_5 & 1 \\ x_6^2 & y_6^2 & x_6 & y_6 & x_6y_6 & 1 \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \end{bmatrix} \tag{3.13}$$

Túto rovnicu prepíšeme symbolicky:

$$Q \times G = R \tag{3.14}$$

Koeficienty vypočítame upravenou rovnicou:

$$G = Q^{-1} \times R \tag{3.15}$$

Po získaní koeficientov nastáva opäť overenie optimálnosti plochy. Vezmeme každý bod $A_i = [x_i, y_i, z_i]$ zo vstupných a vypočítame jeho vzdialenosť od plochy funkciou `lagrangemult`. Výstup z nej bude minimálna kvadratická vzdialenosť každého bodu od testovacej plochy.

Opakujeme predošlý postup. Spočítame percento *inliers*. Túto hodnotu porovnáme so

vstupným parametrom hraničné percento. Pokiaľ je vyššia, plocha je optimálna, výpočet je dokončený. Pokiaľ nie je, celý výpočet sa zopakuje pre ďalšiu náhodnú vzorku dát.

Vstupné hodnoty do funkcie:

A	matica, ktorej jednotlivé riadky predstavujú vstupné body
hran_v	hraničná kvadratická vzdialenosť bodov od plochy, pre ktorú považujeme body za *inliers*
hran_p	minimálne percento *inliers*
p	pravdepodobosť nájdenia úspešného riešenia
w	pomer *inliers* k celkovému počtu bodov

Funkcia metódy RANSAC pre nájdenie kvadratickej plochy:

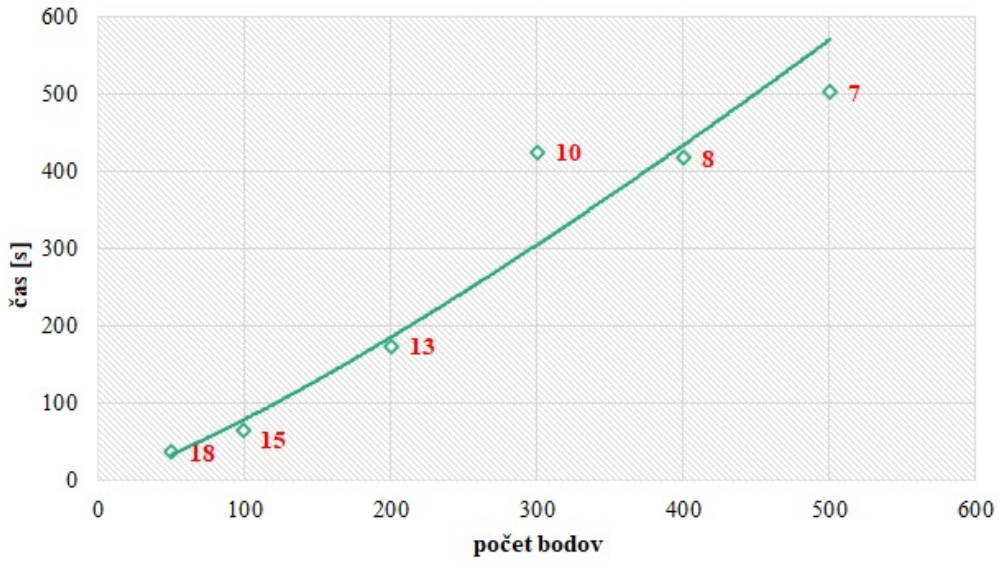
```
function [H]=ransac1(A,hran_v,hran_p,p,w)
  % nastaveníme parametre na začiatku 0
  it=0;
  a=0; b=0; c=0; d=0; e=0; f=0;
  perc=0;
  m=size(A); % velkost matice bodov
  points=zeros(6,m(2));

  k=log(1-p)/log(1-w^6); % výpočet potrebných iterácií

  % cyklus sa zastaví pri presiahnutom počte iterácií alebo pri nájdení
    optimálneho riešenia

  while (perc<hran_p) && (it<k)
    % v každej iterácií vynulujeme premenné
    vzdial=0;
    pocetbodov=0;
    perc=0;

    % náhodné indexy, podla ktorých vyberieme vstupnú vzorku
    t=randperm(m(1),6);
    for i=1:6
      % do každého riadku matice data vložíme postupne náhodné body
      data(i,:)=A(t(i),:);
    end

    %vypočet a,b,c,d,e,f pomocou sústavy rovníc

    %rovnice sú tvaru: G*Q=R, G=[a; b; c; d; e; f]... matica neznámych

    Q=zeros(6,6);
    Q(:,1)=data(:,1).^2; Q(:,2)=data(:,2).^2;          Q(:,3)=data(:,1);
    Q(:,4)=data(:,2);    Q(:,5)=data(:,1).*data(:,2); Q(:,6)=ones(1,6)';

    R=data(:,3);

    G=Q\R; % matica výsledkov

    % vypočítane koeficienty sú prvky G
    a=G(1); b=G(2); c=G(3); d=G(4); e=G(5); f=G(6);

    % overíme optimálnost plochy na základe vzdialenosti
```

```
41      for i=1:m(1)
42        bod=A(i,:); % do premennej si uložíme vždy i-ty bod
43
44        % výpočet vzdialenosti bodu od kvadratickej plochy
45        [vzdial]=lagrangemult(bod(1),bod(2),bod(3),G);
46
47        % overíme, či sa bod nachádza pod hraničnou vzdialenostou
48        if vzdial<hran_v
49          pocetbodov=pocetbodov+1; % pripočítame bod
50        end
51      end
52
53      perc=pocetbodov/m(1)*100; % počet percent bodov, ktoré sú pod
        hraničnou vzdialenostou
54
55      it=it+1; %iterácie
56    end
57    H=[a,b,c,d,e,f]; % výstup pre funkciu
58  end
59
```

3.2.1 Časové závislosti

V prípade kvadratickej plochy je výpočet vzdialenosti náročnejší. Vzdialenosť sa musí spočítať pre každý bod vždy ku každej testovacej ploche. Funkcia `vpasolve` použitá v programe hľadá všetky reálne riešenia sústavy rovníc parciálnych derivácií.
Červené čísla opäť predstavujú počty iterácií, ktoré prebehli, kým sa výpočet ukončil.

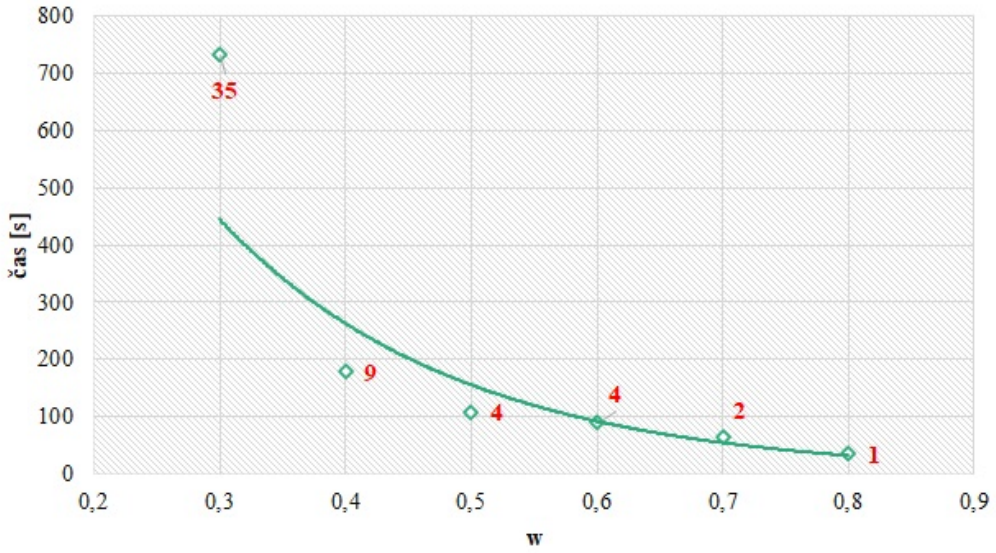
Počet vstupných bodov



Obr. 18: Graf závislosti času na počte vstupných bodov v stovkách

Závislosť vyzerá podobne ako pri predošlom programe. S rastúcim počtom bodov je väčšia pravdepodobnosť, že sa trafíme do *inliers*, lenže pribúda počet bodov, ktorých vzdialenosť musíme spočítať, preto výpočet trvá dlhšie.

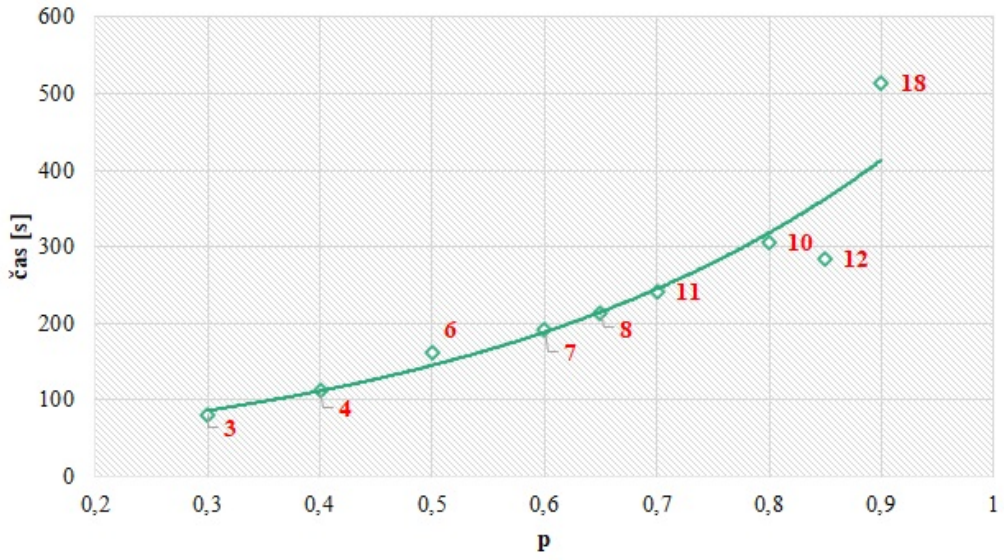
w – pomer *inliers* k celkovému počtu bodov



Obr. 19: Graf závislosti času na w – pomer *inliers* k celkovému počtu bodov

Krivka má klesajúcu tendenciu. Opäť si môžeme všimnúť, že pri väčšom množstve *inliers* program nájde riešenie rýchlejšie.

p – pravdepodobnosť nájdenia úspešného riešenia



Obr. 20: Graf závislosti času na p – pravdepodobnosť nájdenia úspešného riešenia

Pri vyššom p výpočet trvá dlhšie, ale je väčšia pravdepodobnosť, že program nájde optimálne riešenie. Pri nižšom p sa môže stať, že výpočet sa ukončí skôr, ale riešenie sa nenájde.

3.3 Metóda najmenších štvorcov

Ďalšia aproximačná metóda je metóda najmenších štvorcov.

3.3.1 Všeobecná rovnica roviny

Opäť máme rovinu α zadanú všeobecnou rovnicou:

$$\dot{a}x + \dot{b}y + \dot{c}z + \dot{d} = 0 \tag{3.16}$$

Túto rovnicu upravíme na tvar:

$$ax + by + c - z = 0 \tag{3.17}$$

Každý bod $A_i = [x_i, y_i, z_i]$, ktorý sa nachádza na α, má nulovú odchýlku od tejto roviny a spĺňa rovnicu (3.17). Pri aplikovaní metódy najmenších štvorcov na n bodov teda chceme minimalizovať súčet všetkých $ax_i + by_i + c - z_i$ v kvadráte:

$$min(\sum_{i=1}^{n}(ax_i + by_i + c - z_i)^2) \tag{3.18}$$

Označne si sumačný výraz z (3.18):

$$F_1 = \sum_{i=1}^{n}(ax_i + by_i + c - z_i)^2 \tag{3.19}$$

Hľadáme minimum F_1. Body podozrivé z extrému dostaneme, keď parciálne derivácie prvého rádu podľa koeficientov a, b, c položíme rovné nule.

$$\begin{aligned}
\frac{\partial F_1}{\partial a} &= \sum_{i=1}^{n} 2(ax_i + by_i + c - z_i)x_i = 0 \\
\frac{\partial F_1}{\partial b} &= \sum_{i=1}^{n} 2(ax_i + by_i + c - z_i)y_i = 0 \\
\frac{\partial F_1}{\partial c} &= \sum_{i=1}^{n} 2(ax_i + by_i + c - z_i) = 0
\end{aligned} \tag{3.20}$$

Sústavu rovníc upravíme a prepíšeme pomocou matíc:

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} y_i^2 & \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} y_i & \sum_{i=1}^{n} 1 \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_i z_i \\ \sum_{i=1}^{n} y_i z_i \\ \sum_{i=1}^{n} z_i \end{bmatrix} \tag{3.21}$$

Túto rovnicu môžeme symbolicky prepísať:

$$S_1 \times K_1 = P_1 \tag{3.22}$$

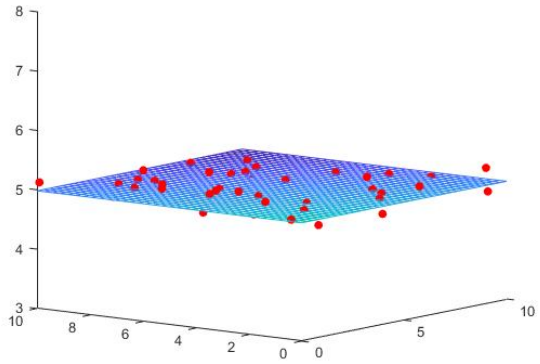
Koeficienty a, b, c vypočítame ako prvky matice K_1:

$$K_1 = S_1^{-1} \times P_1 \tag{3.23}$$

Vstupný parameter do funkcie je matica, ktorej jednotlivé riadky predstavujú vstupné body. Túto maticu označíme A.

Funkcia metódy najmenších štvorcov pre nájdenie všeobecnej rovnice roviny:

```
function [K1]=MNC(A)
% výpočet súm do rovnice:
sumx=sum(A(:,1));
sumy=sum(A(:,2));
sumz=sum(A(:,3));

sumxx=sum(A(:,1).^2);
sumyy=sum(A(:,2).^2);

sumxy=sum((A(:,1)).*(A(:,2)));
sumxz=sum((A(:,1)).*(A(:,3)));
sumyz=sum((A(:,2)).*(A(:,3)));

S1=[sumxx sumxy sumx;
  sumxy sumyy sumy;
  sumx sumy size(A,1)];

P1=[sumxz; sumyz; sumz;];

K1=S1\P1;
end

```



Obr. 21: Metóda najmenších štvorcov použitá na všeobecnú rovnicu roviny

3.3.2 Kvadratická plocha

Máme zadanú kvadratickú rovinu rovnicou:

$$ax^2 + by^2 + cx + dy + exy + f - z = 0 \tag{3.24}$$

Opäť potrebujeme nájsť minimum $(ax_i^2 + by_i^2 + cx_i + dy_i + ex_iy_i + f - z_i)^2$ pre všetky body $A_i = [x_i, y_i, z_i]$, kde $i = 1, ..., n$:

$$min(\sum_{i=1}^{n}(ax_i^2 + by_i^2 + cx_i + dy_i + ex_iy_i + f - z_i)^2) \tag{3.25}$$

Označne si sumačný výraz z (3.25):

$$F_2 = \sum_{i=1}^{n} (ax_i^2 + by_i^2 + cx_i + dy_i + ex_iy_i + f - z_i)^2 \tag{3.26}$$

Vykonajme parciálne derivácie podľa koeficientov a, b, c, d, e, f a položme ich rovné 0:

$$\begin{aligned}
\frac{\partial F_2}{\partial a} &= \sum_{i=1}^{n} 2(ax_i^2 + by_i^2 + cx_i + dy_i + ex_iy_i + f - z_i)x_i^2 = 0 \\
\frac{\partial F_2}{\partial b} &= \sum_{i=1}^{n} 2(ax_i^2 + by_i^2 + cx_i + dy_i + ex_iy_i + f - z_i)y_i^2 = 0 \\
\frac{\partial F_2}{\partial c} &= \sum_{i=1}^{n} 2(ax_i^2 + by_i^2 + cx_i + dy_i + ex_iy_i + f - z_i)x_i = 0 \\
\frac{\partial F_2}{\partial d} &= \sum_{i=1}^{n} 2(ax_i^2 + by_i^2 + cx_i + dy_i + ex_iy_i + f - z_i)y_i = 0 \\
\frac{\partial F_2}{\partial e} &= \sum_{i=1}^{n} 2(ax_i^2 + by_i^2 + cx_i + dy_i + ex_iy_i + f - z_i)x_iy_i = 0 \\
\frac{\partial F_2}{\partial f} &= \sum_{i=1}^{n} 2(ax_i^2 + by_i^2 + cx_i + dy_i + ex_iy_i + f - z_i) = 0
\end{aligned} \tag{3.27}$$

Sústavu prepíšeme do maticového tvaru:

$$\begin{bmatrix}
\sum_{i=1}^n x_i^4 & \sum_{i=1}^n x_i^2y_i^2 & \sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^2y_i^2 & \sum_{i=1}^n x_i^3y_i & \sum_{i=1}^n x_i^2 \\
\sum_{i=1}^n x_i^2y_i^2 & \sum_{i=1}^n y_i^4 & \sum_{i=1}^n x_iy_i^2 & \sum_{i=1}^n y_i^3 & \sum_{i=1}^n x_iy_i^3 & \sum_{i=1}^n y_i^2 \\
\sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_iy_i^2 & \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_iy_i & \sum_{i=1}^n x_i^2y_i & \sum_{i=1}^n x_i \\
\sum_{i=1}^n x_i^2y_i & \sum_{i=1}^n y_i^3 & \sum_{i=1}^n x_iy_i & \sum_{i=1}^n y_i^2 & \sum_{i=1}^n x_iy_i^2 & \sum_{i=1}^n y_i \\
\sum_{i=1}^n x_i^3y_i & \sum_{i=1}^n x_iy_i^3 & \sum_{i=1}^n x_i^2y_i & \sum_{i=1}^n x_iy_i^2 & \sum_{i=1}^n x_i^2y_i^2 & \sum_{i=1}^n x_iy_i \\
\sum_{i=1}^n x_i^2 & \sum_{i=1}^n y_i^2 & \sum_{i=1}^n x_i & \sum_{i=1}^n y_i & \sum_{i=1}^n x_iy_i & \sum_{i=1}^n 1
\end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} =$$

$$= \begin{bmatrix}
\sum_{i=1}^n x_i^2z_i \\
\sum_{i=1}^n y_i^2z_i \\
\sum_{i=1}^n x_iz_i \\
\sum_{i=1}^n y_iz_i \\
\sum_{i=1}^n x_iy_iz_i \\
\sum_{i=1}^n z_i
\end{bmatrix} \tag{3.28}$$

Túto rovnicu si prepíšeme symbolicky:

$$S_2 \times K_2 = P_2 \tag{3.29}$$

Koeficienty a, b, c, d, e, f spočítame ako prvky matice K_2:

$$K_2 = S_2^{-1} \times P_2 \tag{3.30}$$

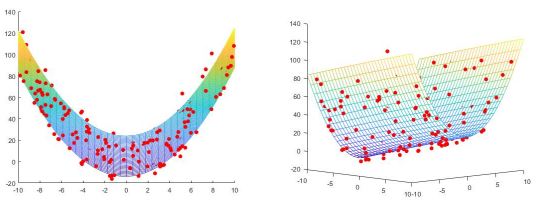
Vstupný parameter do funkcie je matica, ktorej jednotlivé riadky predstavujú vstupné body. Túto maticu označíme A.

Funkcia metódy najmenších štvorcov pre nájdenie kvadratickej plochy:

```
function [K2]=KVMNC(A)
  % výpočet súm do rovnice
  sumx=sum(A(:,1));
  sumy=sum(A(:,2));
  sumz=sum(A(:,3));

  sumxx=sum(A(:,1).^2);
  sumyy=sum(A(:,2).^2);

  sumxy=sum((A(:,1)).*(A(:,2)));
  sumxz=sum((A(:,1)).*(A(:,3)));
  sumyz=sum((A(:,2)).*(A(:,3)));

  sumxxy=sum(A(:,1).^2.*A(:,2));
  sumxxz=sum(A(:,1).^2.*A(:,3));
  sumyyz=sum(A(:,2).^2.*A(:,3));
  sumxyy=sum(A(:,1).*A(:,2).^2);

  sumxxx=sum(A(:,1).^3);
  sumyyy=sum(A(:,2).^3);
  sumxyz=sum(A(:,1).*A(:,2).*A(:,3));

  sumxxxy=sum(A(:,1).^3.*A(:,2));
  sumxyyy=sum(A(:,1).*A(:,2).^3);
  sumxxyy=sum(A(:,1).^2.*A(:,2).^2);

  sumxxxx=sum(A(:,1).^4);
  sumyyyy=sum(A(:,2).^4);

  S2=[sumxxxx sumxxyy sumxxx sumxxy sumxxxy  sumxx;
    sumxxyy sumyyyy sumxyy sumyyy sumxyyy  sumyy;
    sumxxx  sumxyy  sumxx  sumxy  sumxxy   sumx;
    sumxxy  sumyyy  sumxy  sumyy  sumxyy   sumy;
    sumxxxy sumxyyy sumxxy sumxyy sumxxyy  sumxy;
    sumxx   sumyy   sumx   sumy   sumxy    size(A,1)];

  P2=[sumxxz; sumyyz; sumxz; sumyz; sumxyz; sumz;];

  K2=S2\P2;
end

```

Na nasledujúcich obrázkoch sa nachádza kvadratická plocha definovaná rovnicou:

$$z = -0.005x^2 + 0.997y^2 + 1.88x + 0.16y - 0.002xy + 5.51 \tag{3.31}$$



(a) (b)

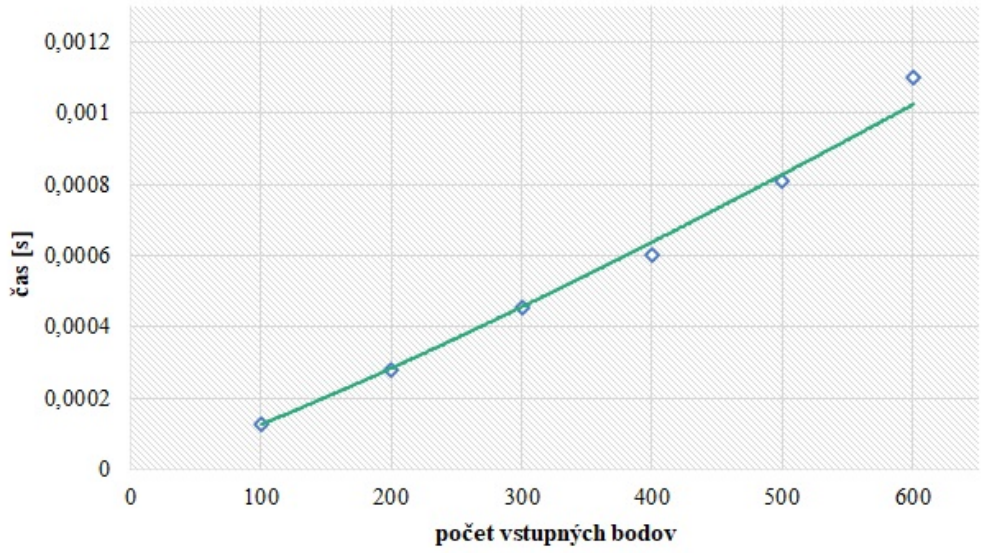
Obr. 22: Metóda najmenších štvorcov použitá na model kvadratickej roviny

3.3.3 Časové závislosti

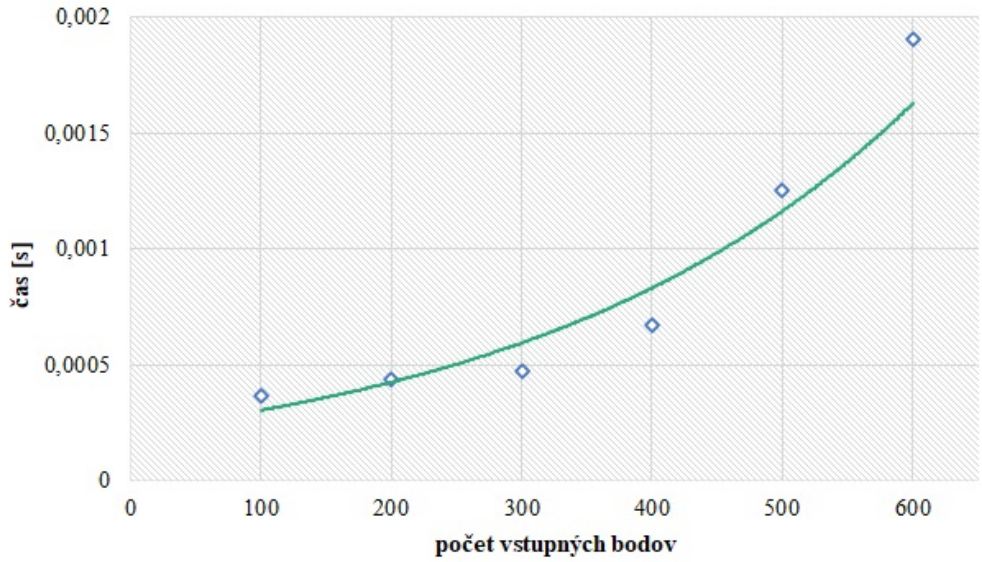
Teraz sa pozrieme na závislosť času na počte vstupných bodov pre metódu najmenších štvorcov.

Počet riadiach bodov

Na nasledujúcom obrázku sa nachádzajú grafy závislosti času na počte vstupných bodov pre metódu najmenších štvorcov.



Obr. 23: Graf závislosti času na počte riadiacich bodov pre rovinu



Obr. 24: Graf závislosti času na počte riadiacich bodov pre kvadratickú plochu

V oboch prípadoch majú grafy obdobný priebeh. V prípade kvadratickej plochy je výpočet o niečo náročnejší, preto sú namerané hodnoty času väčšie ako pri všeobecnej

rovnici roviny. Čím viac bodov vstúpi do funkcie, tým bude výpočet časovo náročnejší. Je to spôsobené jednotlivými sumami, ktoré sa nachádzajú v maticových rovniciach.

3.4 B-spline

V tejto časti práce sa pozrieme na vytvorenie B-spline plochy. Aby sme ju vytvorili, musíme najprv vedieť vytvoriť B-spline krivku z riadiacich bodov. Zopakujeme si jej definíciu:

Definícia 3.1. Nech p je stupeň krivky, P_i sú riadiace body s počtom $n+1$ a $U = \{u_0, ..., u_p, u_{p+1}, ..., u_{m-p-1}, u_{m-p}, ..., u_m\}$ je uniformný uzlový vektor, kde prvých $p+1$ uzlov sú nuly a posledných $p+1$ uzlov sú jednotky. Ďalej nech $N_{i,p}(u)$ je bázová funkcia. B-spline krivka má potom tento tvar:

$$C(u) = \sum_{i=0}^{p} N_{i,p}(u)P_i, \quad u \in < u_0, u_m > \tag{3.32}$$

Na jej výpočet môžeme použiť viacero algoritmov. Jedným z nich je de Casteljau algoritmus, ktorý využíva rekurzívnu vlastnosť B-spline bázových funkcií $N_{i,p}(u)$. Ďalším z nich, De Boorov algoritmus, je zovšeobecnením de Casteljau algoritmu, a naň sa v tejto práci zameriame.

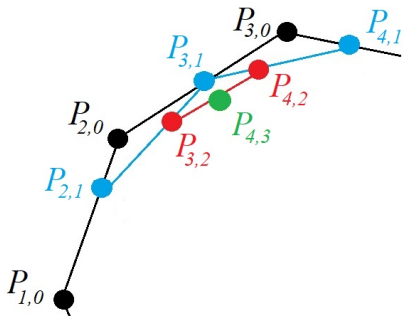
3.4.1 De Boorov algoritmus

De Boorov algoritmus je rýchly a numericky stabilný algoritmus, ktorý slúži na nájdenie bodu B-spline krivky. Tento bod vypočíta pomocou riadiacich bodov.

vstupné hodnoty:	parameter $u \in < u_0, u_m >$, uzlový vektor U, riadiace body P_i
výstupná hodnota:	bod na krivke B-spline $C(u)$

Algorithm 1: De Boorov algoritmus

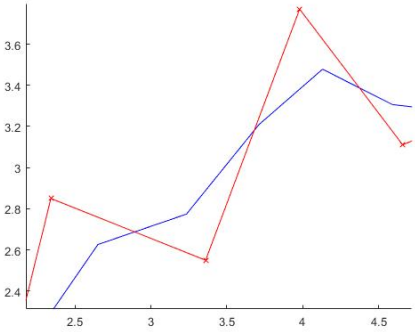
Ak $u \in < u_k, u_{k+1}) \wedge u \neq u_k$, potom $h = p$ a $s = 0$
Ak $u = u_k$ a u_k je uzol multiplicity s, potom $h = p - s$
Afektované riadiace body $P_{k-s}, P_{k-s-1}, ..., P_{k-p}$ skopírujeme do nového poľa a premenujeme na $P_{k-s,0}, P_{k-s-1,0}, ..., P_{k-p,0}$.
for *r=1 to h-s* **do**
  for *i=k-p+r to k-s* **do**
    $a_{i,r} = (u - u_i)/(u_{i+p-r+1} - u_i)$
    $P_{i,r} = (1 - a_{i,r})P_{i-1,r-1} + a_{i,r}P_{i,r-1}$
  end
end
$P_{k-s,p-s}$ je bod $C(u)$



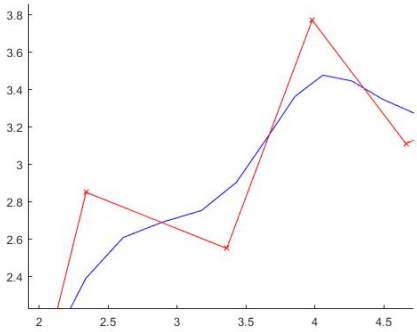
Obr. 25: Výpočet bodov De Boorovým algoritmom

Takto získame jeden bod B-spline krivky pre konkrétnu hodnotu parametra u. Na získanie celej krivky vypočítame jednotlivé $P_{k-s,p-s}$ pre delenie intervalu $< u_0, u_m >$, ktoré si sami zvolíme. Čím jemnejšie delenie zvolíme, tým budú body krivky hustejšie a krivka môže nadobudnúť takmer spojitý tvar pre ľudské oko.

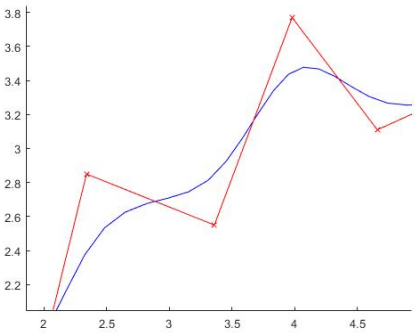
Na nasledujúcich obrázkoch vidíme B-spline krivky stupňa 2 s rovnakými riadiacimi bodmi pri rôznom delení intervalu $< u_0, u_m >$. Pohľad je zväčšený, aby sme mohli lepšie vidieť rozdiely.



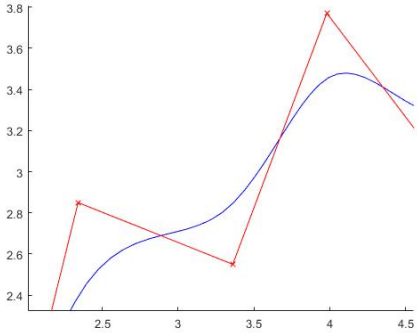
(a) 15 deliacich sekvencií



(b) 30 deliacich sekvencií



(a) 50 deliacich sekvencií



(b) 100 deliacich sekvencií

Obr. 27: B-spline krivky s rôznym počtom deliacich sekvencií

Teraz si vysvetlíme funkcie naprogramované v MATLABE.

Treba podotknúť, že zatiaľ sme pri B-spline funkciách indexovali všetko od 0. MATLAB však indexuje od 1. Výpočty tomu boli prispôsobené.

Prvá funkcia spočíta uzlový vektor, multiplicitu parametru u a tiež index intervalu alebo uzlu, kde sa u nachádza v uzlovom vektore.

Uniformný uzlový vektor sa počítal jednoducho: najprv sme do prvých p+1 prvkov vektoru uložili nuly a do posledných $p+1$ prvkov jednotky. Prostredné hodnoty $u_{p+1}, ..., u_{m-p-1}$ sme spočítali tak, aby boli ekvidistantné spolu s prvkami u_p a u_{m-p}.

Hodnotu, o ktorú sa v prostredných hodnotách uzlového vektoru posúvame, spočítame takto:

$$krok = \frac{1}{n-p+1} \tag{3.33}$$

Príklad 3.2. Máme B-spline krivku druhého stupňa s ôsmimi riadiacimi bodmi. Chceme vypočítať jej uzlový vektor U:

$$\begin{aligned} p &= 2 \\ n &= 8 \end{aligned} \tag{3.34}$$

Dĺžku uzlového vektoru vypočítame:

$$m+1 = p+n+1 = 11 \tag{3.35}$$

Teraz zistíme krok, o ktorý sa budeme v prostredných hodnotách vektoru posúvať:

$$krok = \frac{1}{n-p+1} = \frac{1}{6} \tag{3.36}$$

Krajných $p+1$ uzlov naplníme na začiatku nulami a na konci jednotkami. Prostredné hodnoty budeme posúvať po $\frac{1}{6}$. Uniformný uzlový vektor v našom prípade vyzerá takto:

$$U = \{0, 0, 0, \frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}, 1, 1, 1\} \tag{3.37}$$

Ďalší zo vstupných parametrov De Boorovej funkcie je multiplicita a index parametra. Pre každý prvok delenia intervalu u_i si uložíme jeho pozíciu a násobnosť:

Ak $u_i \geq U_j \wedge u_i \leq U_{j+1}$ potom $k_i = j$, $s_i = 0$
Ak $u_i = U_j$ potom $k_i = j$ a s_i získame v cykle

Vstupné hodnoty do MATLAB funkcie:

n	počet riadiacich bodov
p	stupeň krivky
sekv	počet sekvencií delenia intervalu

Funkcia na výpočet uzlového vektoru, multiplicity a indexu:

```
function [U,s,k]=knotvektor(n,p,sekv)

  m=2;  % dimenzia riadiacich bodov
  uk=0;

  % naplnenie vektoru nulami na začiatku a jednotkami na konci
  U(1:(p+1))=0;
  U((n+1):(p+n+1))=1;

  % prostredné hodnoty u dopočítame ekvidistantne
  krok=1/(n-p);

  for i=1:(n-p-1)
    uk=uk+krok;
    U((p+1+i):n)=uk;
  end

  % vytvoríme delenie intervalu podla počtu sekvencií
  u = linspace(0, 1, sekv);

  k=zeros(size(u));
  s=zeros(size(u));

  % algoritmus
  % pre každú sekvenciu vypočítame jej multiplicitu v uzlovom vektore a
    index, kde sa v ňom nachádza

  for i=1:numel(u)

    h=0;

    for j=1:(numel(U))

      % pokial sa nachádza sekvencia u(i) medzi U(j) a U(j+1) uzlami,
    jej multiplicita je nulová a uložíme si jej index

      if (U(j)<u(i)) && (U(j+1)>u(i))
        k(i)=j;
        s(i)=0;
        break; %našli sme interval, v ktorom sa nachádza u(i)

        % pokial sa sekvencia rovná uzlu, spočítame jej multiplicitu a
    uložíme index prvého zhodného uzlu

      elseif  u(i)==U(j)
        k(i)=j;
        s(i)=s(i)+1;
      end
    end
  end
end
```

Vstupné hodnoty do funkcie De Boor:

P	matica, ktorej jednotlivé riadky predstavujú riadiace body
p	stupeň krivky
U	uzlový vektor
k	index sekvencie
s	multiplicita sekvencie v uzlovom vektore
u	sekvencia

Nasledujúcu funkciu zavoláme pre každú sekvenciu delenia intervalu u_i, jej násobnosť s_i v uzlovom vektore, index k_i a potom parametre riadiace body P, uzlový vektor U a stupeň krivky p, ktoré sú rovnaké pre všetky u_i. B-spline bude potom aproximovať všetky riadiace body.

```
function [C]=deboor(P,p,U,k,s,u)

  m=size(P,1); % dimenzia  riadiacich bodov
  n=size(P,2); % pocet riadiacich bodov

  Pk = zeros(m,n,p);

  % algoritmus

  h=p-s;

  if h>0

    c= k-p;
    b= k-s;
    Pk(:, c:b, 1) = P(:, c:b); % vyberieme afektované body zo vstupných

    for r=2:(h+1)
      for j=(k-1-p+r):(k-s)

        a(j,r)=(u-U(j))/(U(j+p-r+1+1)-U(j));
        Pk(:,j,r)=(1-a(j,r))*Pk(:,j-1,r-1)+a(j,r)*Pk(:,j,r-1);

      end
    end

    C = Pk(:,k-s,p-s+1);  % získaný bod na krivke

  elseif k == numel(U)

    C = P(:,end);   % posledný bod

  else

    C = P(:,1);   % prvý bod

  end
end
```

3.4.2 B-spline plocha

Výpočet jedného bodu na B-spline ploche vyžaduje dvojnásobne použitie De Boorovho algoritmu. V tejto časti si to priblížime. Zopakujme si definíciu B-spline plochy:

Definícia 3.3. Nech p a q sú stupne krivky. Majme $m+1$ riadkov a v každom riadku $n+1$ riadiacich bodov $P_{i,j}$, kde $0 \leq i \leq m$ a $0 \leq j \leq n$ a uzlové vektory $U = \{u_0, ..., u_r\}$ $V = \{v_0, ..., v_s\}$ v nasledujúcom tvare:

$$U = \{0, ..., 0, u_{p+1}, ..., u_{r-p-1}, 1, ..., 1\}$$
$$V = \{0, ..., 0, v_{q+1}, ..., u_{s-q-1}, 1, ..., 1\},$$

(krajné uzly majú násobnosť $p+1$ v U a $q+1$ vo V).

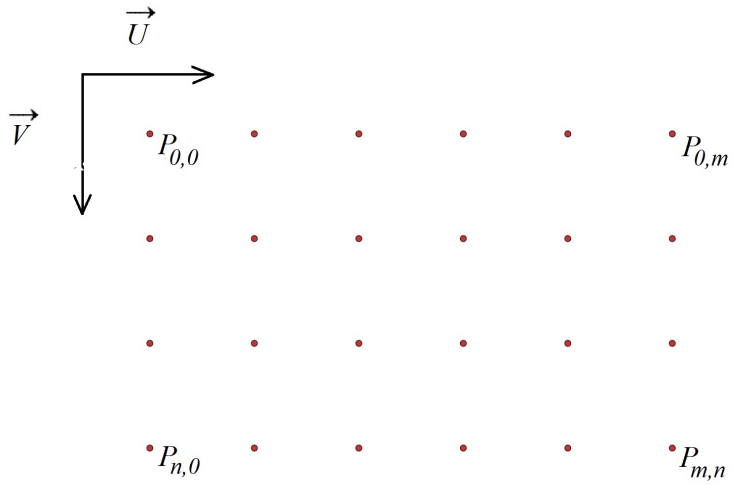
Nech $\{u, v\}$ je dvojica čísel, pre ktoré platí: $u \in < u_0, u_r >$ a $v \in < v_0, v_s >$ a nech $N_{i,n}(u)$ a $N_{j,q}(v)$ sú bázové funkcie. Potom definujeme B-spline plochu takto:

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) P_{i,j} \tag{3.38}$$

U má $r+1$ uzlov a V má $s+1$ uzlov. Indexy teda musia spĺňať nasledovné rovnice:

$$\begin{aligned} r &= n+p+1 \\ s &= m+q+1 \end{aligned} \tag{3.39}$$

Poznámka. Vstupné riadiace body $P_{i,j}$ musia mať pri kolmom pohľade nasledujúcu štruktúru:



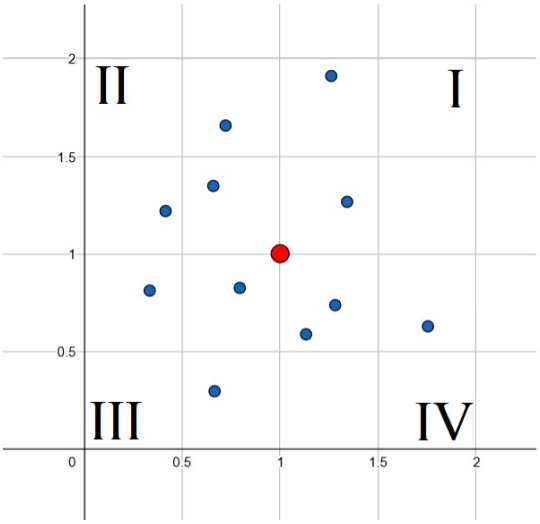
Obr. 28: Sieť bodov $P_{i,j}$

3.4.3 Mriežková štruktúra

Riadiace body, ktoré vstupujú do výpočtu B-spline plochy, musia mať požadovanú mriežkovú štruktúru. Namerané body sú väčšinou nepravidelne usporiadané, a preto ich potrebujeme upraviť do mriežky ako na obrázku obr.28. Výšku bodov mriežky spočítame štatistickou metódou najbližieho suseda.

Metóda najbližšieho suseda

Nasledujúci text je čerpaný zo zdroja [11]. Metóda najbližšieho suseda je jednoduchá interpolačná metóda, ktorá odhaduje neznámu hodnotu na základe hodnôt najbližšieho miesta, v ktorom údaj poznáme. Za hľadaný údaj teda budeme považovať známy údaj z najbližšieho miesta. Metódu si rozšírime tak, že budeme uvažovať najbližší údaj z každého kvadrantu.
Červený bod je bod mriežky a modré body sú namerané údaje. V každom kvadrante vyberieme modrý bod, ktorý je najbližšie k červenému a vypočítame ich vzdialenosť.



Obr. 29: Metóda najbližšieho suseda

Čím je modrý bod bližie k červenému, tým je podstatnejší pre výpočet. Jednotlivým n minimálnym vzdialenostiam d_i pridáme ďalšiu hodnotu, a to váhu λ_i, pričom platí:

$$\sum_{i=1}^{n} \lambda_i = 1 \tag{3.40}$$

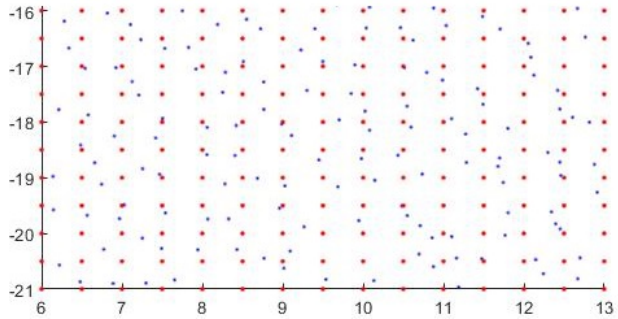
Váhu λ_i minimálnej vzdialenosti d_i vypočítame nasledovne:

$$\lambda_i = \frac{\frac{\sum_{k=1}^{n} d_k}{d_i}}{\sum_{j=1}^{n} \frac{\sum_{k=1}^{n} d_k}{d_j}} \tag{3.41}$$

Nakoniec hľadanú výšku v vypočítame ako lineárnu kombináciu váh λ_i jednotlivých vstupných bodov a ich výšok v_i:

$$v = \sum_{i=1}^{n} \lambda_i v_i \tag{3.42}$$

Upravená štruktúra vstupných bodov má tento tvar:



Obr. 30: Sieť upravených červených bodov $P_{i,j}$

Pred vypočítaním mriežky pre vstupné body je nutné určiť, v ktorom smere ju budeme počítať. V tejto práci sme mriežku počítali v rovine xz, tým pádom výšky riadiacich bodov v boli ich y-ové súradnice.

Pokiaľ by sa mal výpočet mriežky zovšeobecniť, program by bol veľmi náročný, preto sme ho upravili podľa zvolených vstupov.

Prejdime na výpočet samotných bodov B-spline plochy. Vezmeme prvý riadok a vypočítame body B-spline krivky De Boorovým algoritmom pre zvolený počet deliacich sekvencií v tomto smere. Následne tento výpočet zopakujeme pre všetky vzniknuté stĺpce, tiež pre zvolený počet sekvencií v stĺpcovom smere. Takto získame body $S(u, v)$ pre všetky dvojice $\{u, v\}$, kde $u \in < u_0, u_r >$ a $v \in < v_0, v_s >$.

Vstupné hodnoty do funkcie:

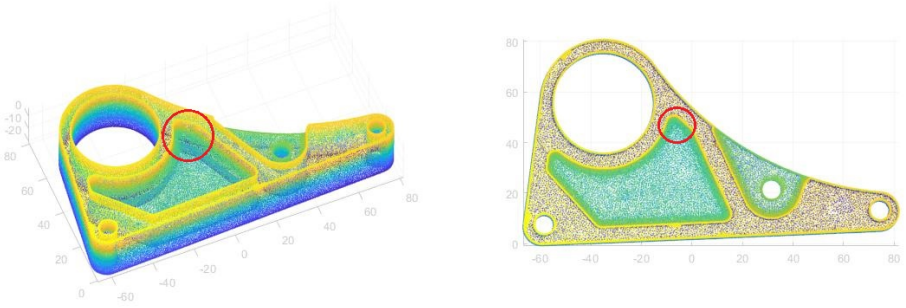
P	body mriežky $P_{i,j}$
p	stupeň krivky v smere riadkov
q	stupeň krivky v smere stĺpcov
sekv1	počet deliacich sekvencií v smere riadkov
sekv2	počet deliacich sekvencií v smere stĺpcov

Funkcia B-spline plochy:

```
function [deb_col]=bsplineplocha(P,p,q,sekv1,sekv2)

  ps=size(P,2); % počet stlpcov
  pr=size(P,3); % počet riadkov

  [U,s,c]=knotvektor(pr,p,sekv2); % uzlový vektor a vektor multiplicity
   pre stĺpce
  [V,t,d]=knotvektor(ps,q,sekv1); % uzlový vektor a vektor multiplicity
   pre riadky

  u=linspace(0,1,sekv2); % parameter u
  v=linspace(0,1,sekv1); % parameter v

  deb_row=zeros(3,sekv2,pr);
```

```
13
14  P_z=P(3,1,1:pr); % uložíme si súradnice z, pre jednotlivé riadky sú
     rovnaké, do De Boorovho algoritmu nevstupujú
15
16  % DEBOOROV ALGORITMUS PO RIADKOCH
17  for k=1:sekv1 %indexovanie podla sekvencií
18
19    for i=1:pr   % riadky
20      Pdyn=P(1:2,:,i); % do Pdyn si uložíme i-ty riadok
21      deb_row(1:2,k,i)=deboor(P(1:2,:,i),p,V,d(k),t(k),v(k)); % na i-ty
     riadok aplikujeme De Boorov algoritmus
22      deb_row(3,k,i)=P_z(i); % dopíšeme chýbajúcu súradnicu z
23    end
24
25  end
26
27  P_x=deb_row(1,1:sekv1,1);  % uložíme si súradnice x, pre jednotlivé
     stĺpce sú rovnaké, do De Boorovho algoritmu nevstupujú
28
29  % DEBOOROV ALGORITMUS PO STĹPCOCH
30  for k=1:sekv1
31
32    for i=1:pr
33      Pdyn2(:,i)=deb_row(:,k,i);  % do Pdyn2 si uložíme i-ty stĺpec
34    end
35
36    for i=1:sekv2 %indexovanie podla sekvencii
37      deb2(2:3,i)=deboor(Pdyn2(2:3,:),q,U,c(i),s(i),u(i)); % na i-ty
     stĺpec aplikujeme De Boorov algoritmus
38    end
39
40    deb2(1,1:i)=P_x(k); % dopíšeme chýbajúcu súradnicu x
41    deb_col(:,:,k)=deb2; % v deb_col sú uložené hladané body B-spline
     plochy
42  end
```

Vstupné body, ktoré boli použité na výpočet mriežky a potom B-spline plochy sme získali orezaním 3D skenu. Na nasledujúcich obrázkoch sa nachádza 3D sken.

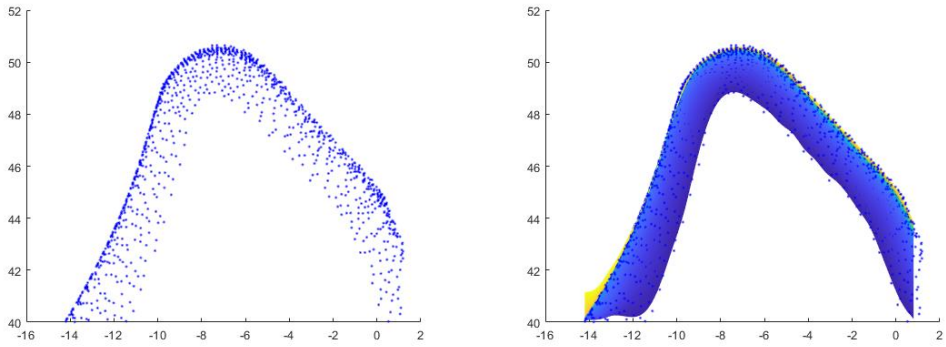


(a) (b)

Obr. 31: 3D sken

Červeným krúžkom je z neho označená zvolená oblasť, ktorá bola použitá na aproximáciu B-spline plochou.

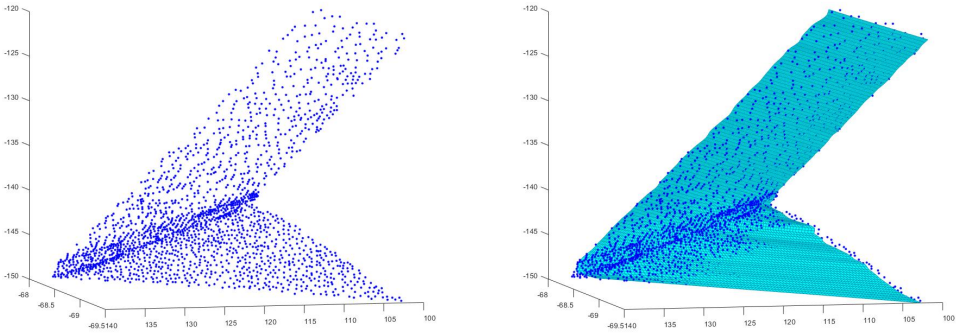
Aproximovali sme B-spline plochu medzi bodmi z 3D skenu:



(a) Vstupné body (b) Výstupná B-spline plocha

Obr. 32: Aproximácia vstupných bodov pomocou B-spline plochy

Funkcia sa dá použiť aj na iné vstupné body, pokiaľ sú vhodne otočené. Výšky bodov musia byť ich ypsilonové súradnice. Na nasledujúcich obrázkoch aproximujeme iné vstupné body B-spline plochou:



(a) Vstupné body (b) Výstupná B-spline plocha

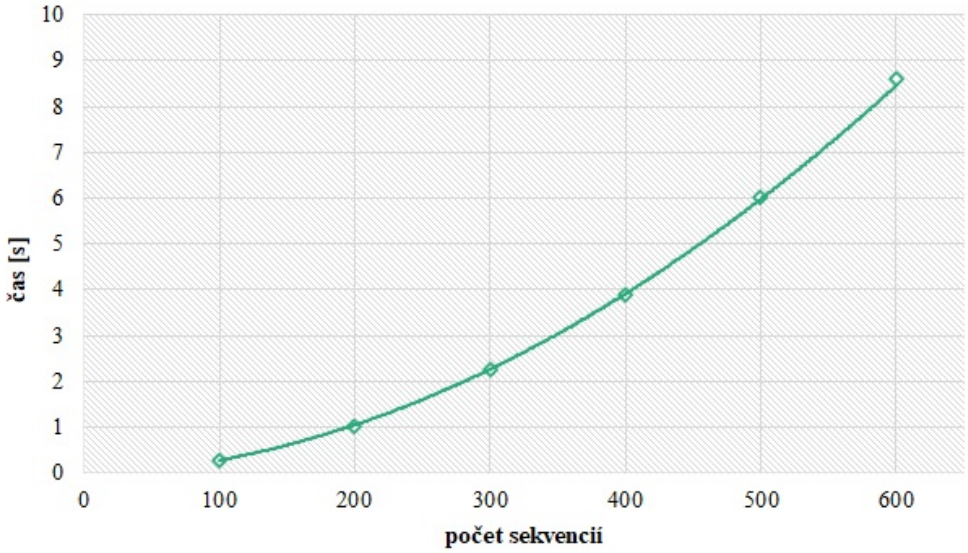
Obr. 33: Aproximácia vstupných bodov pomocou B-spline plochy

3.4.4 Časové závislosti

Narozdiel od iteratívnej metódy RANSAC, De Boorov algoritmus neberie do úvahy iterácie.

Počet sekvencií

Nasledujúci graf ukazuje závislosť času na počte sekvencií delenia intervalu. Hodnoty na osi x predstavujú počet sekvencií pre každý smer. Uvažujeme teda rovnaký počet pre smer riadkov aj pre smer stĺpcov.

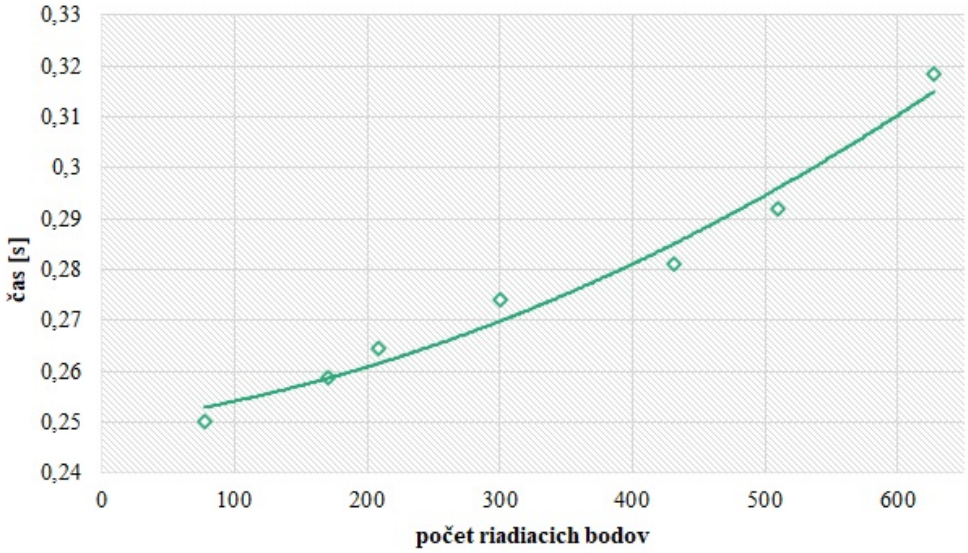


Obr. 34: Graf závislosti času na počte sekvencií

Program vypočíta bod na ploche B-spline pre každú dvojicu $\{u, v\}$, kde $u \in < u_0, u_m >$ a $v \in < v_0, v_n >$. Pri jemnejšom delení intervalov nám teda algoritmus spočíta viac bodov B-spline plochy ako pri hrubšom. To znamená, že výpočet zaberie viac času, avšak plocha je hladšia a my ju vidíme spojitejšie. Preto je vhodné použiť viac sekvencií. Problém by nastal, pokiaľ by sme zadali počet sekvencií delenia menší ako počet prvkov uzlového vektoru, potom by sa neaproximovala krivka, a teda ani plocha, medzi niektorými riadiacimi bodmi.

Počet riadiach bodov

Teraz sa pozrime na vplyv počtu riadiacich bodov na trvanie výpočtu.



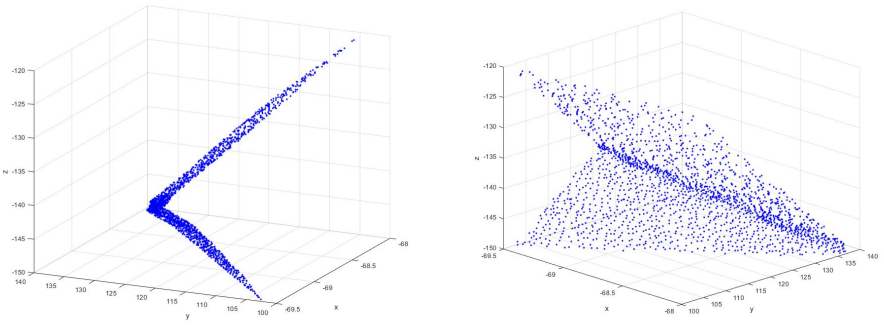
Obr. 35: Graf závislosti času na počte riadiacich bodov

Závislosť sa merala pri nemennom počte sekvencií delenia intervalov, tým pádom je počet bodov plochy stále rovnaký. Avšak dĺžka uzlového vektoru je závislá od počtu riadiacich

bodov. Čím je ich viac, tým má uzlový vektor viac prvkov a do výpočtu potom vstupuje aj viac riadiacich bodov, preto zaberá viac času.

4 Porovnanie aproximačných metód

Na porovnanie metód sme použili 1948 vstupných bodov vyrezaných zo skenu. Pre tieto vstupné body sme hľadali optimálnu aproximačnú rovinu definovanú všeobecnou rovnicou alebo B-spline plochou.



(a) (b)

Obr. 36: Vstupné body z 3D skenu

RANSAC

Body si najprv rozdelíme na hornú a dolnú časť. Na obe časti oddelene použijeme RANSAC. Takto sme volili preddefinované hodnoty pre oba prípady:

Počet bodov	w	p	Hraničná vzdialenosť	Hraničné percento
1948	0.7	0.75	0.2	80

Výpočet trval 0.0032 sekúnd a pre oba prípady prebehli 2 iterácie. Výsledné roviny boli definované rovnicami:

$$z = 2.082x - 0.898y + 115.191 \tag{4.1}$$
$$z = -11.389x + 0.580y - 1001.040 \tag{4.2}$$

Priemerná odchýlka bodov od roviny bola 0.16484 pre hornú časť a 0.15377 pre časť dolnú.

Metóda najmenších štvorcov

Pokračovali sme na metódu najmenších štvorcov. Opäť sme rozdelili body na dve časti. Výpočet trval 0.0037 sekúnd. Výsledné optimálne roviny boli definované rovnicami:

$$z = 2.065x - 0.898y + 114.343 \tag{4.3}$$
$$z = -9.038x + 0.417y - 818.809 \tag{4.4}$$

Priemerná odchýlka bodov od roviny bola 0.06997 pre hornú časť a 0.15377 pre časť dolnú.

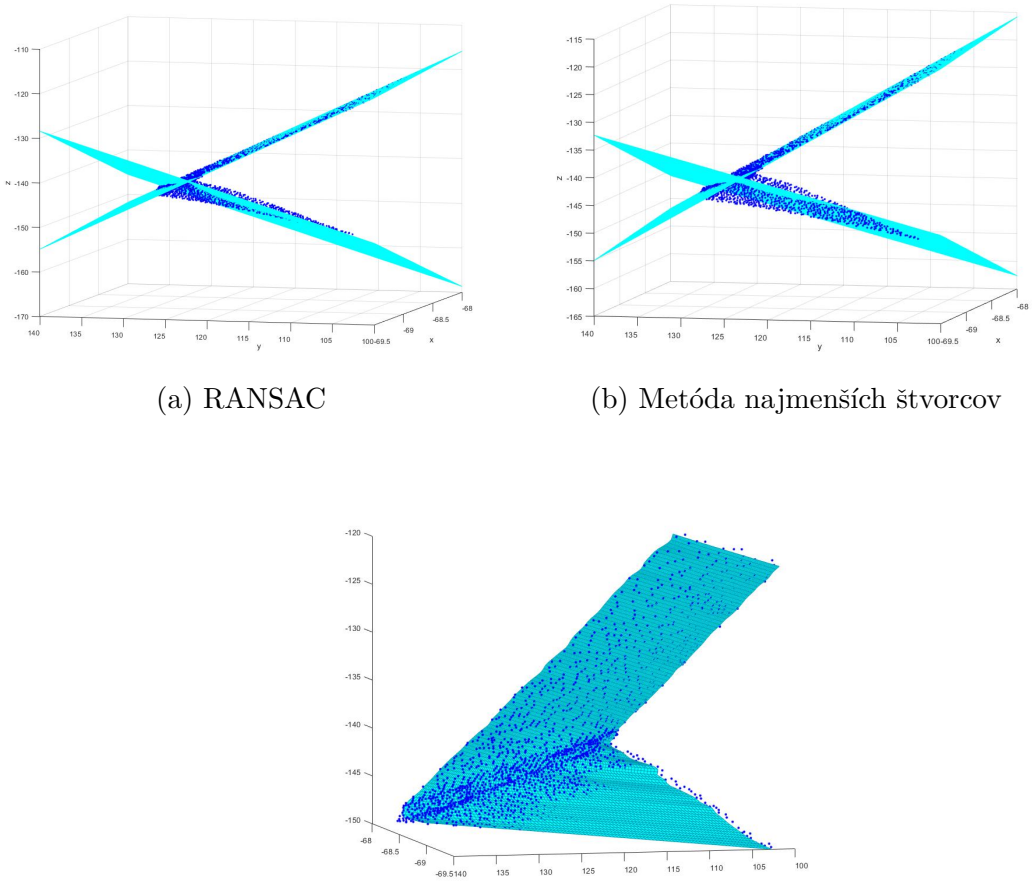
B-spline plocha

Na vstupné body z obr.36 sme použili aj aproximáciu B-spline plochou s parametrami:

Počet bodov	sekv1	sekv2	p	q
1948	300	300	3	3

Výpočet trval 3.067 sekúnd, priemerná odchýlka bodov od plochy bola 0.0038.

Vykreslenie riešení všetkých metód



(a) RANSAC

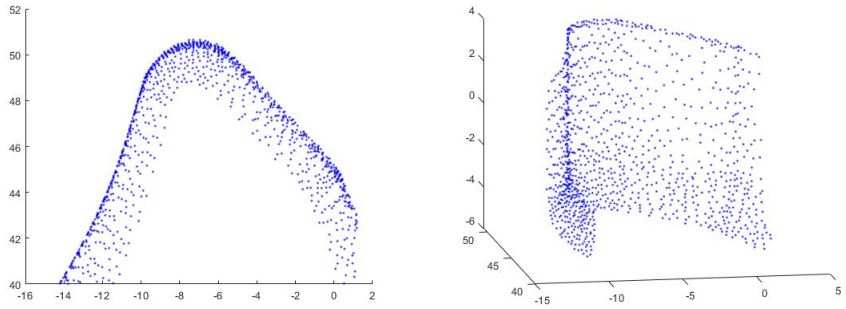
(b) Metóda najmenších štvorcov

Obr. 38: B-spline plocha

Výpočet v prípade metódy RANSAC a metódy najmenších štvorcov vyžadoval podobný čas. Výsledky sú tiež veľmi podobné. Obe metódy teda zvládli vypočítať koeficienty rovnice veľmi rýchlo a sú použiteľné. Metóda RANSAC funguje na princípe náhodného výberu. Optimálny výsledok nie je vždy zaručený, môže sa meniť pomocou vstupných parametrov, ktoré najprv musíme odhadovať, prípadne počítať. Metóda najmenších štvorcov nám vždy zaručí rovnaký výsledok, pretože počíta vždy rovnakú sústavu rovníc, ktoré sme získali použitím parciálnych derivácií.

B-spline plocha však aproximuje body iným spôsobom. Kopíruje ich vzhľad neporovnateľne presnejšie, pretože nie je obmedzená žiadnym konkrétnym matematickým modelom. Počíta jednotlivé B-spline krivky podľa toho, kde sa body nachádzajú. Výpočet je oproti prvým dvom metódam časovo náročnejší aj pracnejší, ale zato oveľa presnejší.

Teraz otestujeme metódy na iných vstupných bodoch:



(a) (b)

Obr. 39: Vstupné body z 3D skenu

Pre tieto vstupné body sme hľadali optimálnu kvadratickú plochu pomocou aproximačných metód.

RANSAC

Takto sme volili preddefinované hodnoty:

Počet bodov	w	p	Hraničná vzdialenosť	Hraničné percento
1208	0.7	0.75	1.25	80

Výpočet trval 964.243 sekúnd. 92% *inliers* bolo od plochy v maximálne kvadratickej hraničnej vzdialenosti. Výsledná optimálna plocha bola definovaná rovnicou:

$$z = 1.16x^2 + 0.09y^2 - 0.35x + 0.29y + 0.31xy - 199.68 \tag{4.5}$$

Priemerná odchýlka bodov od plochy bola 0.4633.

Metóda najmenších štvorcov

Výpočet trval 0.0006 sekúnd. Výsledná optimálna plocha bola definovaná rovnicou:

$$z = 0.25x^2 - 0.016y^2 - 0.035x + 3.459y + 0.068xy - 123.045 \tag{4.6}$$

Priemerná odchýlka bodov od plochy bola 0.7245.

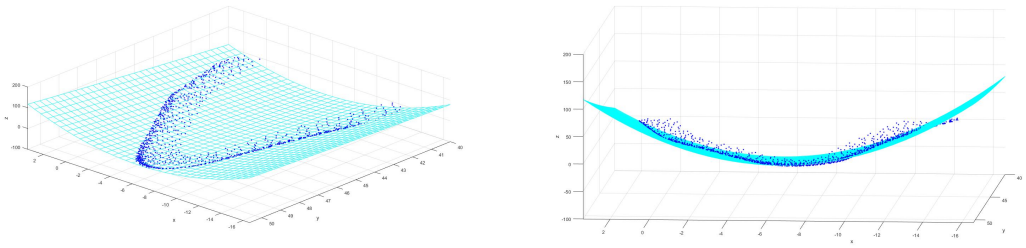
B-spline plocha

Na vstupné body z obr.39 použijeme aj aproximáciu B-spline plochou s parametrami:

Počet bodov	sekv1	sekv2	p	q
1208	300	300	3	3

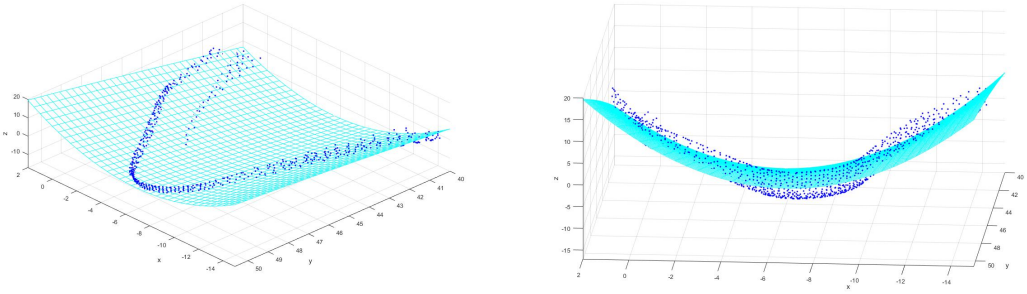
Výpočet trval 2.431 sekúnd, priemerná odchýlka bodov od plochy bola 0.0081.

Vykreslenie riešení všetkých metód



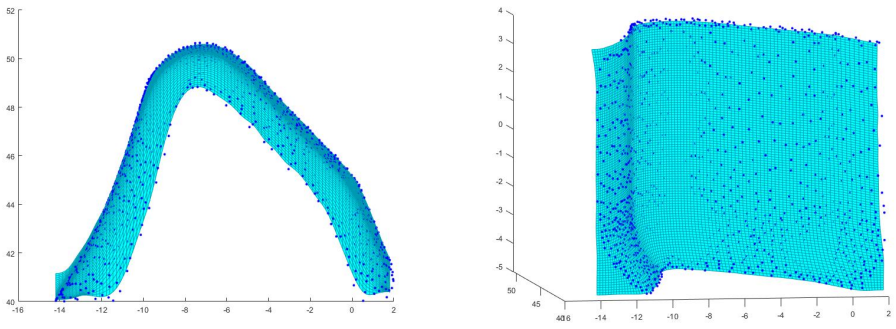
(a) (b)

Obr. 40: RANSAC



(a) (b)

Obr. 41: Metóda najmenších štvorcov



(a) (b)

Obr. 42: Aproximácia B-spline plochou

Optimálne plochy, ktoré boli vypočítané metódami RANSAC a metódou najmenších štvorcov sa opäť podobajú ako v predošlom prípade. Metóde RANSAC výpočet trval oveľa

dlhšie vzhľadom na náročnejší výpočet vzdialenosti bodov od plochy. Aproximácia B-spline plochou opäť kopírovala vzhľad bodov pomocou parametrických kriviek. Výsledok preto vyzerá najpresnejšie.

Záver

Aproximačné metódy sa používajú v oblasti reverzného inžinierstva, pri ktorom sa z hotového komponentu spätne vytvára model. Jeden zo spôsobov získania dát z kompomentu je 3D skenovanie.

Cieľom tejto práce bolo naštudovanie aproximačných metód pre trojrozmerné dáta, ich spracovanie v programovacom jazyku MATLAB a použitie na konkrétne dáta získané z 3D skenu. Práca začína teoretickou časťou, ktorá sa zaoberá vysvetlením princípov metód a ďalej pokračuje na spracované a popísané programy. Potom bolo dôležité otestovať programy pre rôzne vstupy a parametre. V poslednej kapitole sme metódy porovnali pre rozdielne dáta z 3D skenu.

Všetky metódy sú použiteľné, každá má svoje výhody. RANSAC je robustná metóda, ktorá hľadá optimálne riešenie pomocou minimálnej vzorky dát. Avšak optimálne riešenie je v tomto prípade subjektívne. Odvíja sa od preddefinovaných testovacích kritérií. Metóda najmenších štvorcov nevyberá náhodne vzorku dát, ale použije všetky dáta. Tým pádom je výsledok pre dané body vždy rovnaký. Aproximácia B-spline plochou pracuje inak ako predošlé dve metódy. Počíta v jednotlivých stĺpcoch a riadkoch bodov B-spline krivky a z nich vytvorí celú plochu. Avšak riadiace body vstupujúce do výpočtu B-spline plochy musia mať mriežkovú štruktúru. Namerané dáta sú väčšinou neusporiadané, preto musíme použiť zvolenú interpolačnú, aproximačnú alebo štatistickú metódu na úpravu bodov do požadového tvaru.

Pred konkrétnou aproximáciou je výhodné dáta najprv analyzovať a na základe toho si vybrať vhodnú metódu. Pokiaľ dokážeme dopredu určiť správny matematický model, môžeme použiť RANSAC alebo metódu najmenších štvorcov. Pokiaľ je tvar zložitejší, B-spline plocha ho bude aproximovať presnejšie, ale nevyhneme sa úprave dát.

Literatúra

[1] Přesná průmyslová 3D metrologie [online]. [cit. 2021-4-28]. Dostupné z: https://www.atos-core.com/cz/index.php

[2] ATOS Triple Scan Technology — Blue Structured Light 3D Scanner for Accuracy [online]. [cit. 2021-04-17]. Dostupné z: https://www.capture3d.com/3d-metrology-solutions/3d-scanners/atos-triple-scan

[3] KRBA, Martin. *Identifikace počítače na základě časových značek paketů.* Brno, 2012. Bakalářská práce. VUT. Vedoucí práce ING. Jan Kaštil.

[4] MANZUR, Angel. *Got outliers? RANSAC them!* [online]. In: . 2019 [cit. 2021-04-11]. Dostupné z: https://medium.com/@angel.manzur/got-outliers-ransac-them-f12b6b5f606e

[5] FISCHLER, M. A. a R. C. BOLLES. *Random sample consensus. Communications of the ACM.* 1981, 24(6), 381-395. ISSN 0001-0782. Dostupné z: doi:10.1145/358669.358692

[6] BARNOVSKÁ, M. Extrémy funkcie n premenných.*Matematická analýza III: Zbierka príkladov na cvičenia pre 2.ročník* [online]. s. 107-112 [cit. 2021-04-11]. Dostupné z: http://www.iam.fmph.uniba.sk/skripta/maiii/iii6.pdf

[7] MATEMATIKA online[online]. Brno: Ústav matematiky FSI VUT Brno,2005 [cit. 2020-06-06]. Dostupné z:https://mathonline.fme.vutbr.cz/Matematicka-analyzanbspI/sc-1225-sr-1-a-265/default.aspx

[8] SOBOTA, Branislav. *NURBS -NeUniformné Racionálne B-Spline* [online]. In: . 2015 [cit. 2021-04-11]. Dostupné z: https://hornad.fei.tuke.sk/predmety/svr/doc/slides2015/SVR0715_NURBS.pdf

[9] SHENE, C.-K. *Introduction to Computing with Geometry Notes* [online]. [cit. 2021-04-11]. Dostupné z: https://pages.mtu.edu/ shene/COURSES/cs3621/NOTES/

[10] THORNE, Tom. *Computer Graphics 17 - Curves and Surfaces 2* [online]. In: . [cit. 2021-04-11]. Dostupné z: http://www.inf.ed.ac.uk/teaching/courses/cg/lectures/slides17.pdf

[11] KAŇUK, Ján. Priestorové analýzy a MODELOVANIE. Košice: Univerzita Pavla Jozefa Šafárika v Košiciach, 2015. ISBN 978-80-8152-290-1.

[12] MIKULÁŠEK, Zdeněk. *Metoda nejmenších čtvercu* [online]. In: . 2003 [cit. 2021-04-11]. Dostupné z: https://astro.physics.muni.cz/download/documents/skripta/F7581mnc_ver1.pdf?fbclid=IwAR3gSRYeBxy-eMb3hFMa55781jwaldex11opjLj-xSUpxvhL2CuTrkR0K-E

[13] MARKOVÁ, Hana. *B-spline křivky.* Praha, 2007. Bakalářská práce. Univerzita Karlova v Praze. Vedoucí práce Doc. RNDr. Karel Najzar, CSc.

[14] PIEGL, Les a Wayne TILLER. *The NURBS book.* 2nd ed. Berlin: Springer-Verlag, 1997. ISBN 35-406-1545-8.

Použité symboly a skratky

$\times$	Obyčajné násobenie matíc
$\cdot$	Obyčajné násobenie skalárov
$\bigtriangledown$	Gradient
$\in$	Patrí
$\forall$	Pre všetky
$\mathbb{R}$	Množina reálnych čísel
$\bar{\beta}$	Vektor β
$\vec{u}$	Vektor u
$\vec{u} \times \vec{v}$	Vektorový súčin dvoch vektorov

Zoznam príloh

Zdrojové kódy v jazyku MATLAB.
Nachádzajú sa v priečinku 2021_Programy_Valachova_Alzbeta_200982.

RANSAC
Funkcia *ransac2* počíta metódou RANSAC optimálnu všeobecnú rovnicu roviny
Funkcia *ransac1* počíta metódou RANSAC optimálnu kvadratickú plochu
Funkcia *lagrangemult* počíta vzdialenosť bodu od kvadratickej plochy
Súbor *vstupy_ransac* otestuje *ransac1, ransac2* na vstupoch

Metóda najmenších štvorcov
Funkcia *MNC* počíta metódou najmenších štvorcov optimálnu všeobecnú rovnicu roviny
Funkcia *KVMNC* počíta metódou najmenších štvorcov optimálnu kvadratickú plochu
Súbor *vstupy_MNC* otestuje *MNC, KVMNC* na vstupoch

B-spline plocha
Funkcia *knotvektor* počíta uzlový vektor, multiplicitu a index sekvencie
Funkcia *deboor* počíta bod nachádzajúci sa na B-spline krivke
Funkcia *makegrid* vyrobí zo vstupných bodov pravidelnú mriežku
Funkcia *bsplineplocha* vyrobí z mriežky B-spline plochu
Súbor *vstupy_bsplineplocha* otestuje funkciu *bsplineplocha* na vstupoch