UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ROZŠÍRENIE AUTOMATICKEJ KONŠTRUKCIE ZÁVISLÉHO ŠVA

Bakalárska práca

2020 **František Tomana**

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ROZŠÍRENIE AUTOMATICKEJ KONŠTRUKCIE ZÁVISLÉHO ŠVA

Bakalárska práca

Študijný program: Aplikovaná informatika
Študijný odbor: 2511 Aplikovaná informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: Doc. RNDr. Milan Ftáčnik, CSc.
Konzultant: Ing. Karol Žitňanský

Bratislava, 2020 **František Tomana**

75594803

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

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

Názov: Rozšírenie automatickej konštrukcie závislého šva
The extension of automatic construction of dependent seam

Anotácia: Firma assyst vyvinula prvú verziu softvérového modulu, ktorý dokáže zrekonštruovať pravidlá pre konštrukciu závislého šva zo vstupnej geometrie hlavnej a nezávislej švovej kontúry. Tento modul dokáže v súčasnosti zrekonštruovať 2/3 zákazníckych dát. Cieľom práce je zanalyzovať možné konštrukčné pravidlá závislého šva, kategorizovať chybové situácie ich implementácie v SW module firmy assyst, navrhnúť vlastné riešenie a implementovať ďalšie algoritmy automatickej konštrukcie závislého šva, ktoré rozšíria existujúci modul, aby pokryl čo najväčšie množstvo dát pri rekonštrukcii závislého šva.

Vedúci: doc. RNDr. Milan Ftáčnik, CSc.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 05.10.2019

Dátum schválenia: 07.10.2019 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

....
študent vedúci práce

Čestné vyhlásenie

Čestne vyhlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím citovaných zdrojov.

V Bratislave, dňa 14. 5. 2020 .....

František Tomana

Poďakovanie

Chcem sa poďakovať svojmu školiteľovi Doc. RNDr. Milanovi Ftáčnikovi, CSc. a odbornému konzultantovi Ing. Karolovi Žitňanskému za cennú pomoc, rady, konzultácie a čas, ktorý mi venovali počas písania bakalárskej práce. Ďakujem aj Ulrike Reng, ktorá má na starosti zabezpečenie kvality, že odborne otestovala implementované softvérové moduly a tým potvrdila ich kvalitu.

ABSTRAKT

V bakalárskej práci vychádzame z úvodnej analýzy dát pôvodného softvérového modulu firmy assyst. Na základe problémových skupín dát sme rozdelili prácu na dve kategórie pre konštrukčné pravidlá: voľný šev a zastrihnutý roh.

Pre konštrukčné pravidlo voľný šev sme navrhli modul, ktorý pomocou rozkopírovania odchýlkových bodov dokáže zachytiť miesta švovej kontúry, ktoré pôvodný softvérový modul odignoroval, teda vytvorí voľný šev na potrebných miestach. Po implementácii doplnenia odchýlkových bodov sme analyzovali ďalší problém, spojený s nekompletnými trajektóriami, zapríčinený stratou zlomového bodu v niektorej zo stupňovaných veľkostí a navrhli sme modul, ktorý dopočíta zlomový bod a vytvorí kompletnú trajektóriu s pôvodnými zlomovými bodmi.

Pre konštrukčné pravidlo zastrihnutý roh sme navrhli modul, ktorý dokáže pomocou pomeru vzdialeností odchýlkových bodov na elementoch hlavnej kontúry detegovať, či sa tieto body nachádzajú na mieste rohového elementu. Následne sme navrhli modul, ktorý pomocou vektorovej geometrie dokáže prednastaviť hodnoty pre zastrihnutý roh, ktoré následne globálny optimalizátor pôvodného softvéru dopočíta, aby dosiahla pracovná verzia závislého šva tvar geometrickej predlohy.

Kľúčové slová: šev, hlavná kontúra, švová kontúra, pracovná verzia závislého šva, voľný šev, zastrihnutý roh, odchýlkové body, konštrukčné pravidlo

ABSTRACT

The bachelor thesis is based on the initial data analysis of the original software module of the assyst company. Based on the problematic groups of data, we divided the thesis into two categories for design rules: free seam and cut corner.

For the design rule free seam, we have designed the module that, by copying the deviation points, can capture the seam contour locations that the original software module ignored, thus creating a free seam at the required locations. After implementing the addition of deviation points, we analyzed another problem with incomplete trajectories, causing the loss of a turn point in one of the graded sizes, and therefore we designed module that added turn point and creates complete trajectory with the original turn points.

For the design rule cut corner, we have designed the module that can use the ratio of the distances of the deviation points on the main contour elements to detect whether these points are in place of the corner element. Subsequently, we designed the module that can use vector geometry to present values for the cut corner, which is then calculated by global optimizer to achieve a working version of the dependent weld shape of the geometric template.

Keywords: seam, main contour, seam contour, working version of the dependent seam, free seam, cut corner, deviation points, design rule

Obsah

ÚVOD **1**

1. VÝCHODISKÁ PRÁCE **2**

1.1. CAD softvér **2**

1.2. Vysvetlenie pojmov **3**
1.2.1. Konštrukčné pravidlá 5
1.2.2. Trajektórie 8

1.3. Formát AAMA/ASTM **9**

1.4. Stupňovanie **9**

1.5. Metrika hodnotenia presnosti kontúry **10**
1.5.1. Minkowského suma 10
1.5.2. Odchýlkové plochy 12

1.6. Proces rekonštrukcie **13**

1.7. Globálna optimalizácia **15**

1.8. Súčasný stav rekonštrukcie **16**
1.8.1. Sekvenčný diagram 17
1.8.2. Analýza problematiky 18

2. NÁVRH **19**

2.1. Voľný šev **20**
2.1.1. Doplnenie odchýlkových bodov 20
2.1.2. Doplnenie trajektórií 23

2.2. Zastrihnutý roh **27**

3. IMPLEMENTÁCIA **30**

3.1. Implementácia konštrukčného pravidla voľný šev **30**
3.1.1. Proces doplnenia odchýlkových bodov 30
3.1.2. Proces doplnenia trajektórií 32

3.2. Implementácia konštrukčného pravidla zastrihnutý roh **34**

4. TESTOVANIE **36**

4.1. Testovanie konštrukčného pravidla voľný šev.....36

4.2. Testovanie konštrukčného pravidla zastrihnutý roh 43

4.3. Možnosti rozšírenia automatickej rekonštrukcie 47

4.4. Hodnotenie softvérového modulu.....47

ZÁVER 50

LITERATÚRA.....52

Úvod

Predkladaná bakalárska práca vznikla v spolupráci s nemeckou firmou Assyst GmbH, ktorá sa zaoberá vývojom aplikácií pre odevný priemysel. Hlavným dôvodom výberu tejto témy bola možnosť nahliadnuť priamo do praxe tvorby softvéru.

Odborníci pôsobiaci v odevnom priemysle používajú rôzne CAD systémy pri tvorbe a návrhu jednotlivých dielov odevu. Tieto diely vytvárajú pomocou konštrukčných pravidiel daného CAD softvéru. Avšak počas prechodu medzi systémami sa stratia dôležité informácie z odevárskeho hľadiska napriek tomu, že majú medzi sebou dohodnutý všeobecný výmenný formát. Preto sa firma Assyst rozhodla vytvoriť softvérový modul, ktorý dokáže aplikovať konštrukčné pravidlá a automaticky zrekonštruovať jednotlivé diely odevu z cudzích CAD softvérov ako keby boli konštruované softvérom firmy Assyst.

V súčasnosti tento softvérový modul na automatickú rekonštrukciu obsahuje nedostatky. Cieľom tejto práce je zanalyzovať ich, problémové dáta zoradiť do skupín podľa dostupných konštrukčných pravidiel. Následne navrhnúť riešenie a implementovať ďalšie algoritmy s cieľom pokryť čo najväčšie množstvo zákazníckych dát počas automatickej rekonštrukcie.

Práca je rozdelená na 4 kapitoly. Prvá sa venuje opisu súčasného stavu softvéru, zdôvodňuje potrebu rozšírenia automatickej konštrukcie a opisuje postupy, ktoré sa používajú pri samotnej rekonštrukcii. Zároveň vysvetľuje dôležité pojmy na pochopenie danej tematiky.

Druhá kapitola popisuje problematiku jednotlivých chybových situácii a opisuje dôkladný návrh riešenia problémov. Na konci tejto kapitoly budeme pripravení na implementáciu.

Tretia kapitola zobrazuje implementačnú časť softvéru, akým spôsobom boli riešené skupiny problémov. Štruktúru softvérových modulov popisujeme triedami a vzťahmi medzi nimi pomocou UML triednych diagramov.

Záverečná kapitola zobrazuje výsledky analýzy zákazníckych dát pred a po implementácii softvérových modulov. Na záver opisuje nedostatky, ktoré sa nepodarilo vyriešiť.

1. Východiská práce

Prvá kapitola s názvom *Východiská práce* vysvetľuje základné pojmy potrebné na pochopenie danej problematiky. Popisuje konštrukčné pravidlá softvéru cad.assyst, ktoré sú potrebné pri rekonštrukcii. Uvádza dôvod prečo je potrebné riešiť danú problematiku. Rozoberá a vysvetľuje súčasný stav softvérového modulu. Ukončená je prvou analýzou zákazníckych dát, ktorá predstavuje východiskový stav pre našu prácu na tejto téme.

1.1. CAD softvér

S rozvojom informatickej oblasti sa globálne dostáva do popredia počítačom podporovaný návrh v rôznych sférach odvetví. „Skratka CAD je počítačom podporovaný návrh alebo počítačová podpora tvorby konštrukčnej dokumentácie. Ide o programové vybavenie pre geometrické a matematické modelovanie súčiastok a ich vlastností“ [7].

Okrem grafických činností CAD systémy umožňujú realizovať aj rôzne inžinierske výpočty a analýzy pričom spolupracujú s desiatkami iných podporovaných programov. Tieto programy zahŕňame do jedného celku ako CAD/CAM systémy.

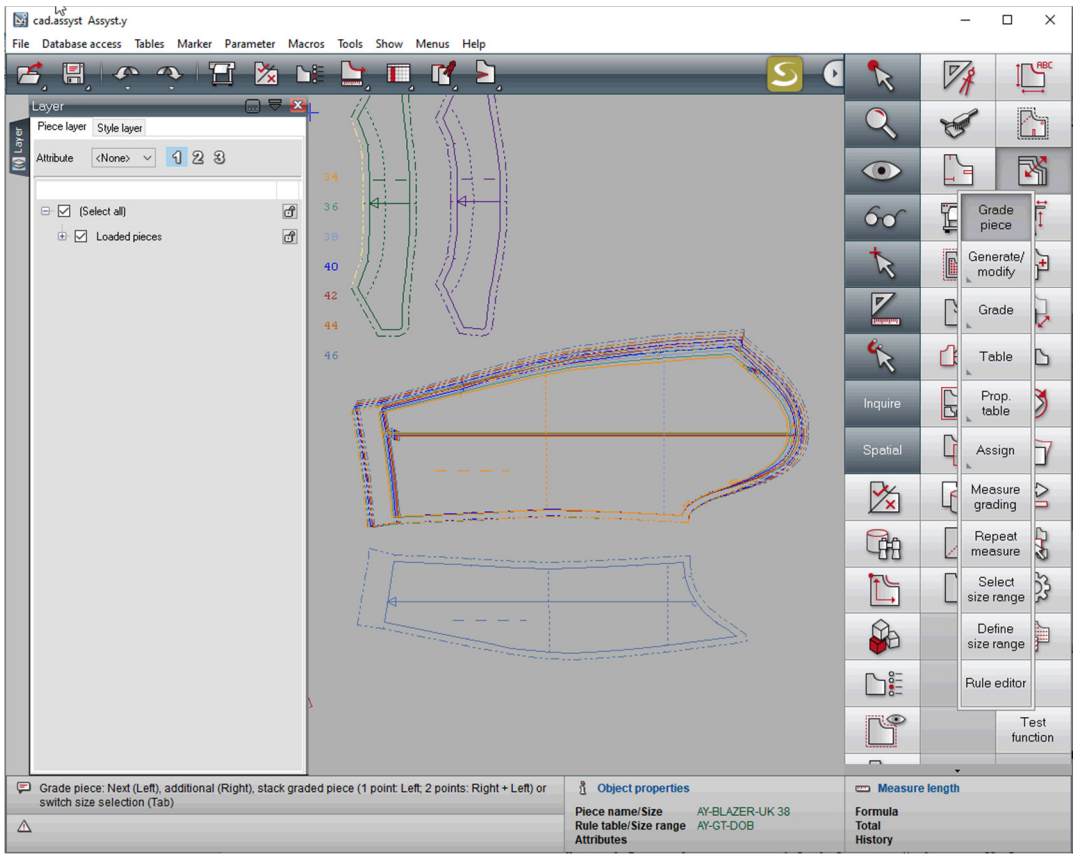
Odborníci v takýchto systémoch navrhujú výkresovú, konštrukčnú a výrobnú dokumentáciu k výrobkom. Proces návrhu vykonávajú s využitím prostriedkov výpočtovej techniky pokročilých konštrukčných softvérových aplikácií ako sú napríklad CATIA, UNIGRAPHICS, AutoCAD, cad.assyst, atď.

Každá z aplikácií ponúka možnosť vybrať si výstupný formát, v ktorom sa uloží celý výrobný výkres. Tieto výstupné formáty môžeme rozdeliť na špecifické, ktoré sú vlastné iba danej aplikácii a štandardizované, pre výmenu dát medzi rôznymi CAD aplikáciami.

„Prvé takéto systémy vznikli v čase, keď požiadavky na vzájomnú komunikáciu neboli tak veľké ako v súčasnosti. Štruktúra ich dát bola v princípe vždy zložitejšia ako napríklad štruktúra textových súborov. Navyše takmer každý výrobca takéhoto programu používa interne uzavretý formát dát, ktorý vyvíja súčasne s vývojom programu. Výrobcovia na nátlak užívateľov zvolili kompromis a vytvorili všeobecné výmenné formáty DXF a iné...“ [1].

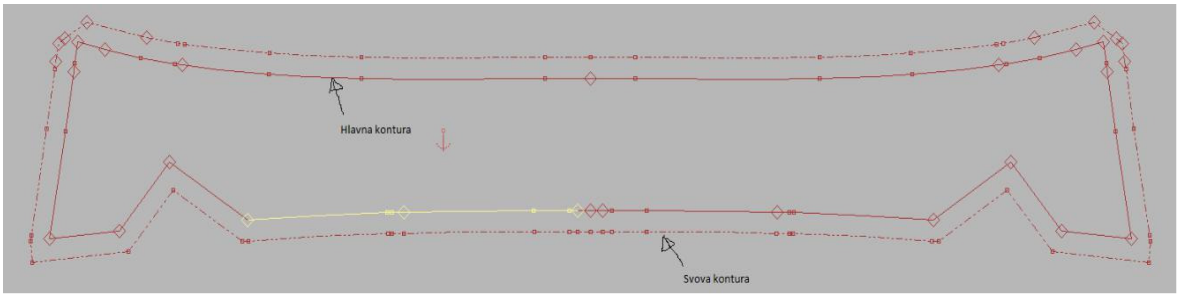
1.2. Vysvetlenie pojmov

Firma assyst pracuje, okrem iného, na softvéry s názvom cad.assyst, ktorý umožňuje dizajnérom navrhovať jednotlivé diely odevu (alebo iných predmetov, ktoré sa režú z látky) a zároveň tieto diely môžu byť stupňované v rôznych veľkostiach na základe stupňovacej tabuľky. Ukážka obrazovky softvéru cad.assyst s príkladom stupňovaného dielu je na obrázku 1.



Obr. č.1 – cad.assyst s príkladom stupňovaného dielu

Každý navrhnutý diel sa skladá z jednotlivých typov elementov a zároveň tento diel tvoria spravidla dve kontúry: hlavná a švová (pozri obrázok č.2). Pri stupňovanej hlavnej kontúre sa v každej veľkosti pridelí rovnaký švový prídavok. To znamená, že švová kontúra sa stupňuje spolu s hlavnou kontúrou. Veľkosť švového prídavku všeobecne závisí od stupňovacej tabuľky.



Obr. č.2 - zobrazenie dielu

- **hlavná kontúra** sa nereže, iba určuje miesto, v ktorom sú samotné diely spolu spájané (zošívané, zavarované a pod.).

- **švová kontúra** určuje predlohu, po ktorej sa reže pomocou rezacieho zariadenia.

Švová kontúra je vytvorená buď ako nezávislý alebo ako závislý šev a skladá sa z elementov.

- **nezávislý šev** je tvorený elementami, ktoré nie sú závislé od hlavnej kontúry. Je to nežiadaný stav, pretože posunutím bodov na hlavnej kontúre švová kontúra nemení svoj tvar.

- **závislý šev** sa skladá z elementov, ktoré sú závislé od elementov na hlavnej kontúre a k týmto elementom sú zadefinované konštrukčné pravidlá.

1.2.1. **Konštrukčné pravidlá**

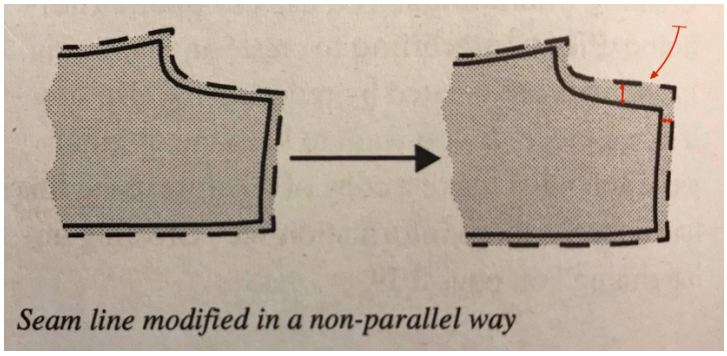
Informácie ako konštruovať závislý šev sú uložené na samotnom diely, na elementoch hlavnej kontúry alebo na bodoch medzi elementami. Ak to nie je špecifikované inak, cad.assyst vytvorí paralelnú kópiu elementov hlavnej kontúry. Ak návrhár posunutím zmení tvar hlavnej kontúry, cad.assyst automaticky zmení tvar šva.

Počas tvorby závislého šva sa dajú priradiť nasledujúce konštrukčné pravidlá:

- Elementom

  - **Paralelné**, kde veľkosť švového prídavku je rovnaká po celej dĺžke elementu. Takto vytvorený švový element je paralelný k elementu hlavnej kontúry [11].

  - **Neparalelné,** kde veľkosť švového prídavku je iná na začiatku a na konci elementu. Počas priebehu sa veľkosť prídavku mení lineárne vzhľadom k polohe medzi začiatkom a koncom, obrázok 3 [11].



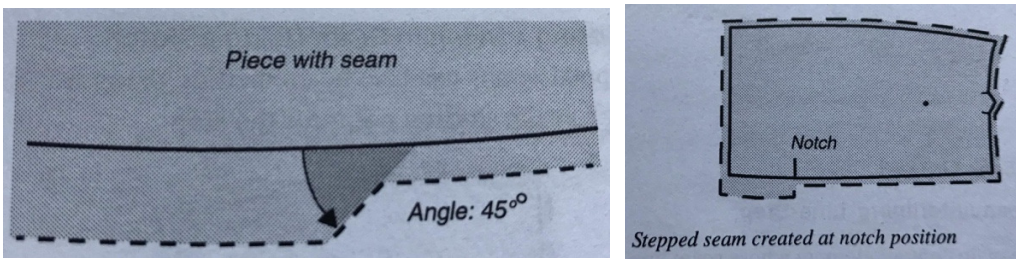
Obr. č.3 – zobrazenie neparalelného šva k elementu

  - **Švový schodík**

    - Vytvorený švový element má dve paralelné oblasti. Prechod medzi nimi môže byť definovaný vzhľadom na začiatok/koniec elementu, (dĺžka kroku zostane pri odstupňovaní rovnaká) alebo v pomere k odstupňovanému zástrihu (notch zobrazený na obrázku 4) na hlavnej kontúre, kde sa dĺžka kroku mení s veľkosťou dielu pri stupňovaní

podľa stupňovacieho pravidla zástrihu (stepped seam created at notch position) [11].

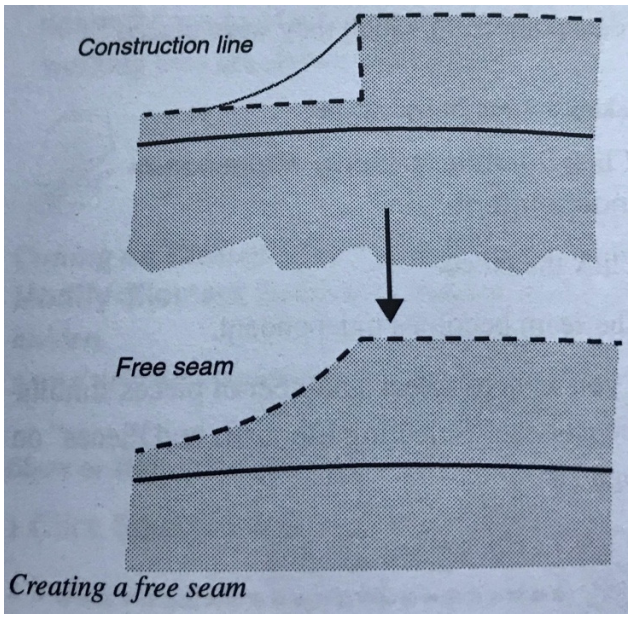
Na vytvorenie schodíku je potrebná šírka švového prídavku na začiatku, koncová šírka, krok a uhol. Uhol sa počíta od hlavnej kontúry v protismere hodinových ručičiek k švovej kontúre bez ohľadu na polohu kroku.



Obr. č.4 – zobrazenie konštrukčného pravidla švový schodík

- **Voľný šev**

  - Definuje sa pomocou ľubovoľnej konštrukčnej čiary, väčšinou sa používa vtedy, ak sa požadovaný tvar nedá dosiahnuť iným konštrukčným pravidlom, obrázok 5 [11].

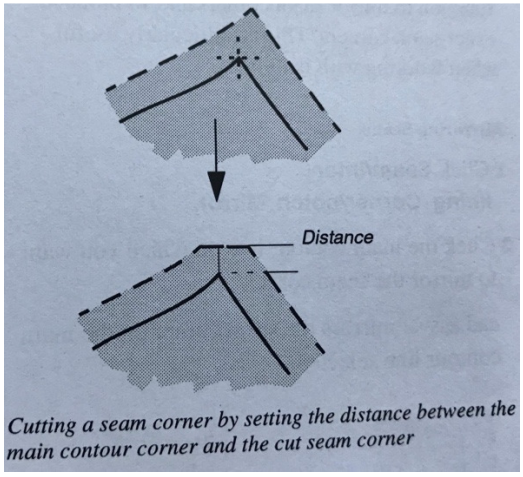


Obr. č.5 – zobrazenie voľného šva

- Bodom medzi elementami

  - **Zastrihnutý roh**

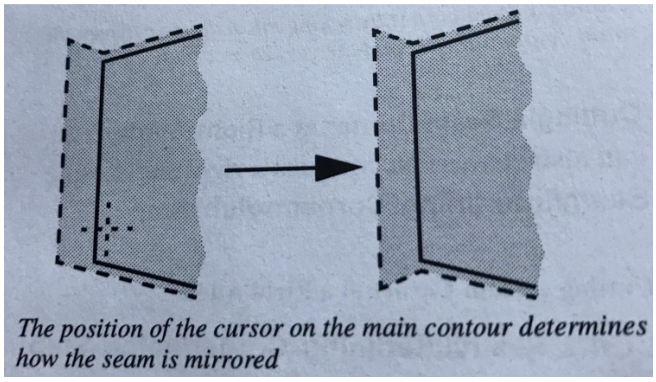
    - Je špecifikovaný vzdialenosťou priamky (orezávajúcej zrekonštruovanú švovú kontúru) od rohového bodu na hlavnej kontúre a jej uhlom, obrázok 6 [11].



Obr. č.6 – zastrihnutý roh

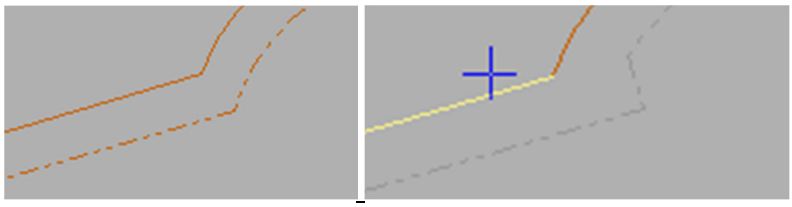
  - **Zrkadlený roh**

    - Zrkadlový roh sa vytvára zrkadlovo k vybranej rohovej čiare šva, obrázok 7 [11].



Obr. č.7 – zrkadlený roh

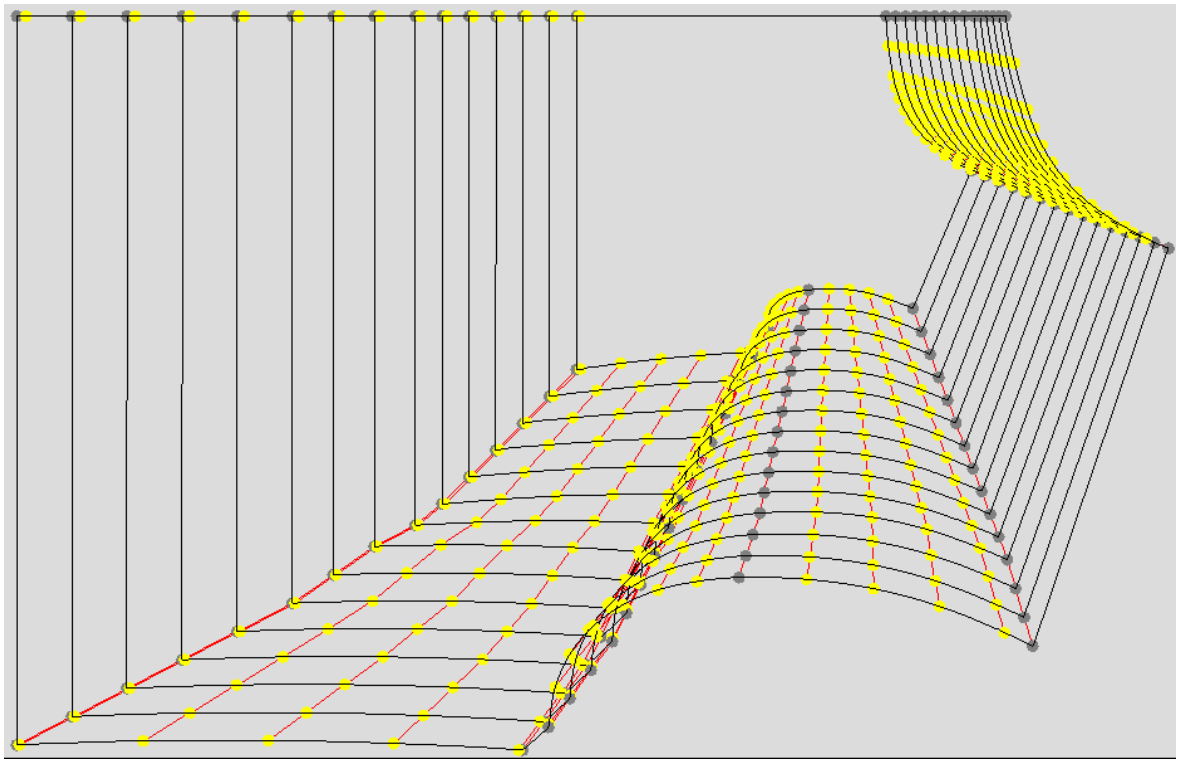
- **Kolmý roh**
    - Vytvára sa ako pravý uhol k vybranej rohovej čiare šva, obrázok 8 [11].



Obr. č.8 – kolmý roh

1.2.2. **Trajektórie**

Trajektórie predstavujú prepojenia bodov medzi veľkosťami. Konkrétne sú to body, ktoré vznikli stupňovaním jedného a toho istého bodu základnej veľkosti. Trajektórie znázorňujú ako sa tento bod pri stupňovaní "pohybuje", (pozri obrázok č.9).



Obr. č.9 – zobrazenie trajektórií

Na obrázku 9, vidíme červenou farbou zobrazené prepojenia bodov vo veľkostiach. Kontúra dielu sa skladá z elementov, pričom body zlomu (zobrazené šedou farbou) sú hranice elementov, miesta, v ktorých je kontúra nespojitá. Medzi dvomi bodmi zlomu je krivka spojitá, teda má v každom bode presne definovanú dotyčnicu. Krivkové body (zobrazené žltou farbou) sú body, cez ktoré „spline algoritmus“ vytvorí spline. Tento spline vytvára automaticky podľa vnútorných pravidiel, dopočítava kontrolné body kubického splinu.

1.3. Formát AAMA/ASTM

Tento formát bol vytvorený na uľahčenie komunikácie medzi CAD/CAM systémami, ktoré reprezentujú dvojrozmerné ploché časti dielo [2]. Táto norma poskytuje aj konvencie na reprezentovanie súvisiacich informácií ako sú stupňovacie tabuľky, zástrihy, typy čiar a iné. Neposkytuje však konvencie na reprezentovanie závislosti elementov. Je v súlade s formátom DXF. Spoločnosť Autodesk vyvinula formát DXF na prenos údajov a výmenu technických výkresov. Do súboru sa ukladá pomocou ASCI znakov bez kompresie. Je to vektorový formát s podporou 256 farieb.

Užívatelia softvéru cad.assyst po exporte technických výkresov pomocou formátu AAMA/ASTM alebo importe z rôznych konkurenčných softvérov dokážu preniesť geometriu navrhnutých dielov, ale stratia konštrukčné pravidlá pre závislosť švovej kontúry.

1.4. Stupňovanie

Stupňovanie dielov je pomerné zvýšenie alebo zníženie veľkosti dielu. Účelom stupňovania je prispôsobiť celý rád typov a veľkostí ľudského tela do jedného štýlu základného dielu. Opačne je to proces premeny základnej veľkosti do ďalších veľkostí pomocou stupňovacej tabuľky, ktorá obsahuje stupňovacie pravidlá [3].

Stupňovacie pravidlá sú založené na ergonomických meraniach tela. Rozlišuje sa medzi ručným stupňovaním a digitálnym stupňovaním za pomoci CAD systémov [10].

Veľkostník alebo ekvivalentne aj stupňovacia tabuľka je sústava normalizovaných veľkostí všetkých odevov. Veľkostník môže byť mužský a ženský [8].

1.5. Metrika hodnotenia presnosti kontúry

Cieľom je pomocou konštrukčných pravidiel závislého šva a hlavnej kontúry vytvoriť novú (na hlavnej kontúre závislú) švovú kontúru, čo najviac podobnú pôvodnej (na hlavnej kontúre) nezávislej švovej kontúre. Nezávislá švová kontúra je naša geometrická predloha, ktorú dostaneme po importe pomocou formátu AAMA/ASTM.

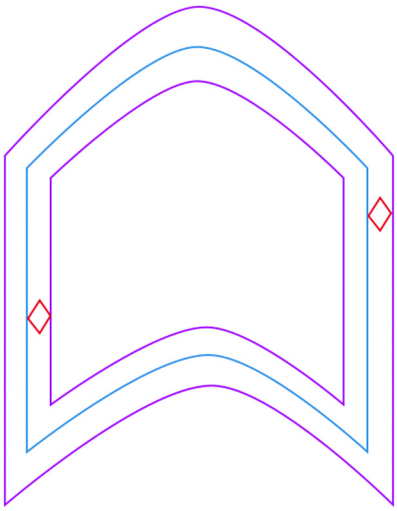
Výsledok postupného aplikovania ďalších a ďalších pravidiel závislej švovej kontúry, ktorá mení svoj tvar počas celej rekonštrukcie budeme nazývať aj pracovná verzia závislej švovej kontúry. Keďže porovnanie geometrickej predlohy a závislej švovej kontúry sa nedá riešiť ekvivalenciou, riešime to podobnosťou. Pre naše potreby je to dostačujúce, lebo v odevnom priemysle je tolerancia daná „množstvom prijateľnej odchýlky od určeného merania, od ktorého je možné vystrihnúť kúsky vzoru, pridať komponenty alebo šiť švy“ [5]. Na porovnávanie podobnosti existujú rôzne metódy, ale v existujúcom algoritme sa využíva Minkowského suma.

Štandardná tolerancia v odevnom priemysle je rádovo vyššia ako v iných odvetviach, pretože súčasné technologické postupy neumožňujú rezať látku tak presne, ako iné materiály. V existujúcej rekonštrukcii sa preto zvolila tolerancia 1mm.

1.5.1. Minkowského suma

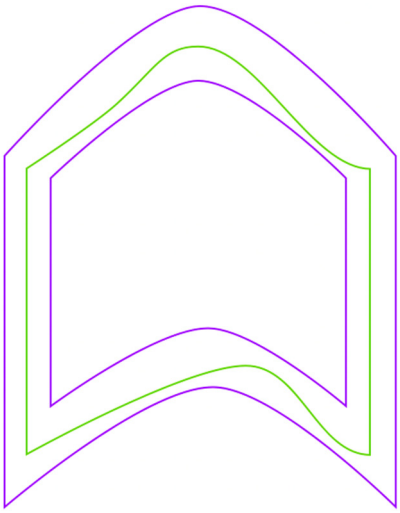
„Minkowského suma množín A a B je bodová množina $\cup_{b \in B} A^b$, kde A^b je množina A posunutá o vektor b, teda množina: $A^b = \{a + b | a \in A\}$. Minkowského suma množín A a B sa označuje $A \oplus B$.“ [4].

V našom prípade sa počíta Minkowského suma nad polygónmi, ktoré majú nekonvexný tvar. Na získanie plochy sa používa bodová množina tvaru diamantu, ktorý je vždy konvexného tvaru a preto je to ľahšia obmena tohto výpočtu.



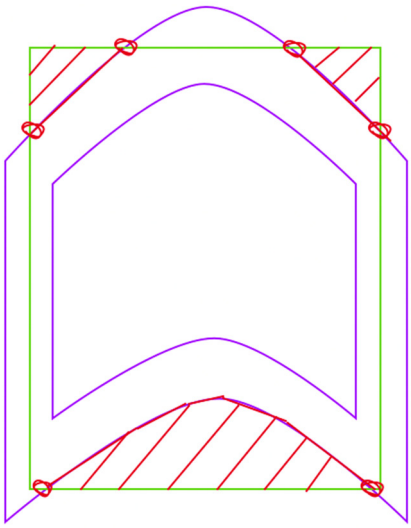
Obr. č.10 – obiehajúci diamant

Na obrázku 10 môžeme vidieť obiehajúci diamant, ktorý je zobrazený červenou farbou a reprezentuje bodovú množinu B. Raz obieha po vnútornej strane geometrickej predlohy, ktorá je zobrazená modrou farbou a reprezentuje bodovú množinu A a spolu vytvoria jednu Minkowského sumu. Druhý raz obieha po vonkajšej strane množiny A a tak vytvoria druhú Minkowského sumu. Týmto vznikne bodová množina C, ktorá je výsledkom týchto dvoch súm a na obrázku 10 je ohraničená hranicami zobrazenými fialovou farbou. Tieto hranice určujú plochu C (nielen jej veľkosť, ale aj tvar), v ktorej keď sa nachádza pracovná verzia závislého šva, tak o nej prehlásime, že je v tolerancii (pozri obrázok č.11).



Obr. č.11 – konštruovaný šev

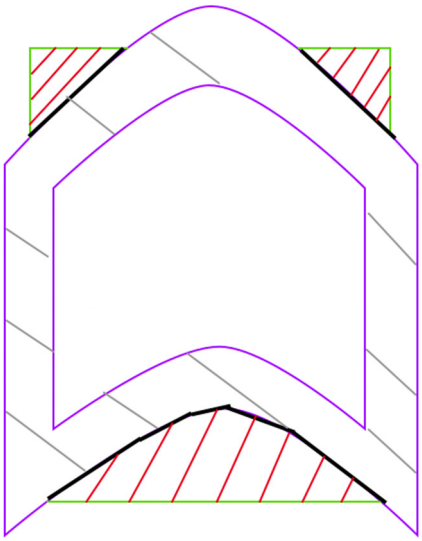
Na obrázku 11 je zelenou farbou zobrazený závislý šev, ktorý je svojou plochou v tolerancii podľa geometrickej predlohy, teda v množine C.



Obr. č.12 – konštruovaný šev mimo toleranciu

Na obrázku 12 môžeme vidieť červenou farbou vyšrafované zobrazenie plochy, ktorá je mimo tolerancie podľa geometrickej predlohy. Pracovná verzia závislého šva je mimo toleranciu a miesta, na ktorých vychádza z tolerančnej plochy sú označené červenými bodmi. Tieto body nazývame ako body odchýlky a ukazujú nám kde nastáva problém.

1.5.2. **Odchýlkové plochy**



Obr. č.13 – odchýlkové plochy

Na obrázku 13 môžeme vidieť, že pracovná množina švovej kontúry, ktorá je zobrazená zelenou farbou sa odchýlila a vychádza z tolerančnej oblasti (množina C), ktorá je zobrazená fialovou farbou. Odchýlkové plochy sú vyplnené červeným šrafovaním a hovoria nám o tom, ako by sme mali zmodifikovať šev, aby sme sa dostali do tolerancie. Ohraničenia plôch sa delia na dve časti. Prvá časť zobrazená zelenou farbou je naša pracovná švová kontúra. Druhá časť zobrazená čiernou farbou predstavuje tvar, ktorý chceme dosiahnuť.

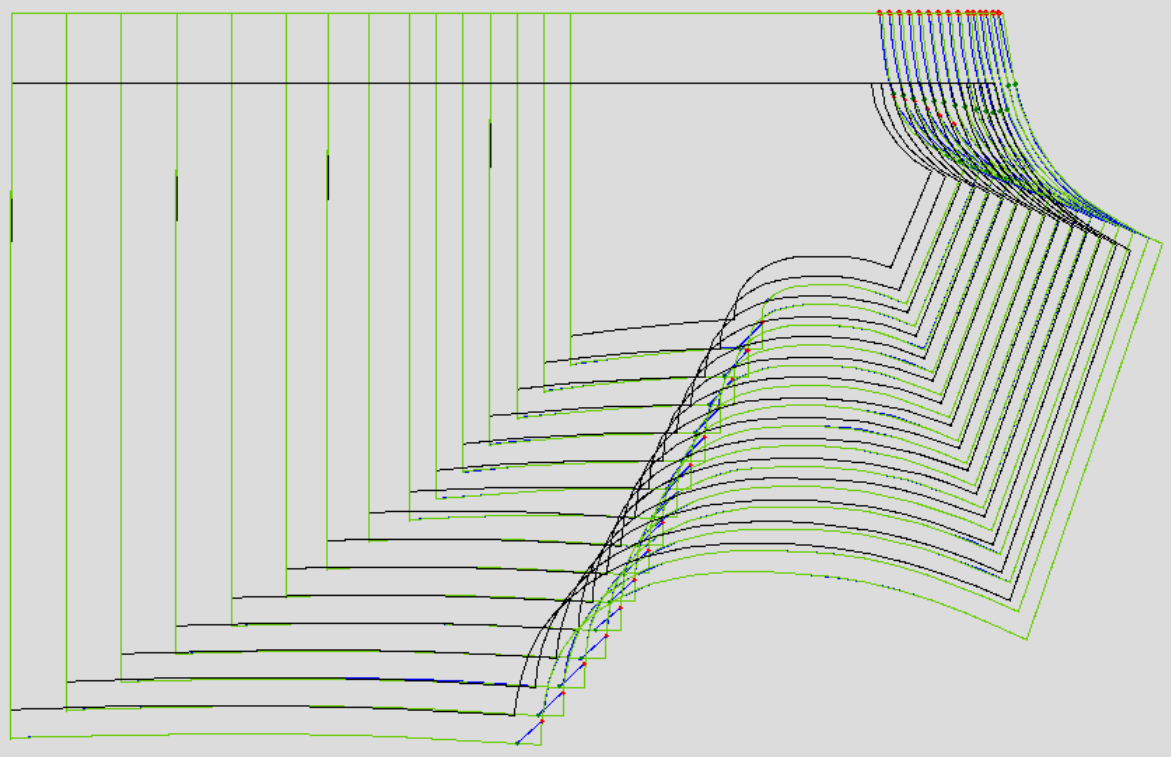
1.6. Proces rekonštrukcie

V podkapitole proces rekonštrukcie opíšeme podrobnejšie pojmy, ktoré sú použité v návrhovej kapitole (pozri obrázok č.14).

Obr. č.14 – zobrazenie rekonštrukcie

Na obrázku 14 je zobrazená rekonštrukcia závislého šva. Proces rekonštrukcie začína na hlavnej kontúre (zobrazená čiernou farbou) a pomocou aplikovania konštrukčných

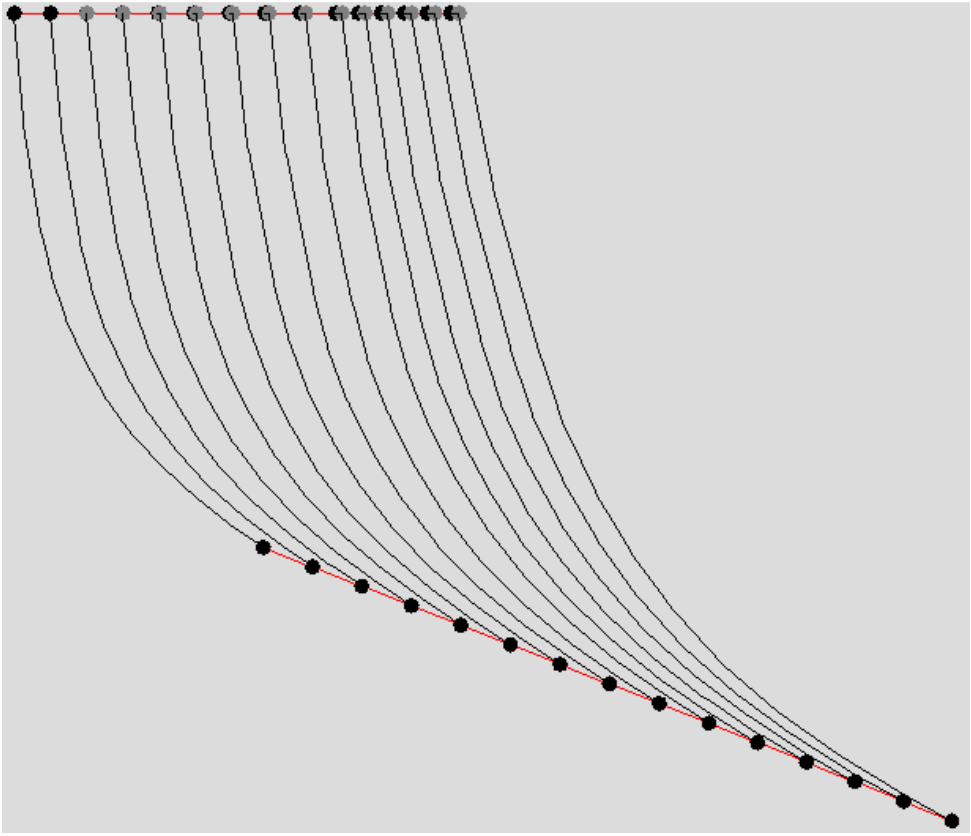
pravidiel sa snaží dosiahnuť tvar geometrickej predlohy (zobrazený modrou farbou). Na mieste, kde pracovná verzia závislého šva vychádza z tolerancie geometrickej predlohy sa vytvoria odchýlkové body (vychádzajúci bod je zobrazený červenou farbou, vchádzajúci bod je zobrazený tmavozelenou farbou). Proces rekonštrukcie väčšinou riešime pre všetky stupňované veľkosti, (pozri obrázok č.15).



Obr. č.15 – zobrazenie rekonštrukcie vo všetkých stupňovaných veľkostiach

Obrázok 15 zobrazuje proces rekonštrukcie vo všetkých stupňovaných veľkostiach. Vidíme, že sa nám nepodarilo dosiahnuť tvar geometrickej predlohy. V dolnej časti sa malo použiť konštrukčné pravidlo zastrihnutý roh a v hornej časti sa malo použiť konštrukčné pravidlo voľný šev (budeme zobrazovať ružovou farbou). Viac o týchto problémoch sa dozvieme v návrhovej kapitole.

Keďže budeme riešiť problém konštrukčného pravidla voľného šva a budeme v návrhovej kapitole dopĺňať trajektórie, budeme potrebovať zobrazovať jednotlivé veľkostné vrstvy v pomocnom zobrazovacom nástroji pre veľkostné vrstvy, (pozri obrázok č.16).



Obr. č.16 – zobrazenie veľkostných vrstiev s trajektóriami

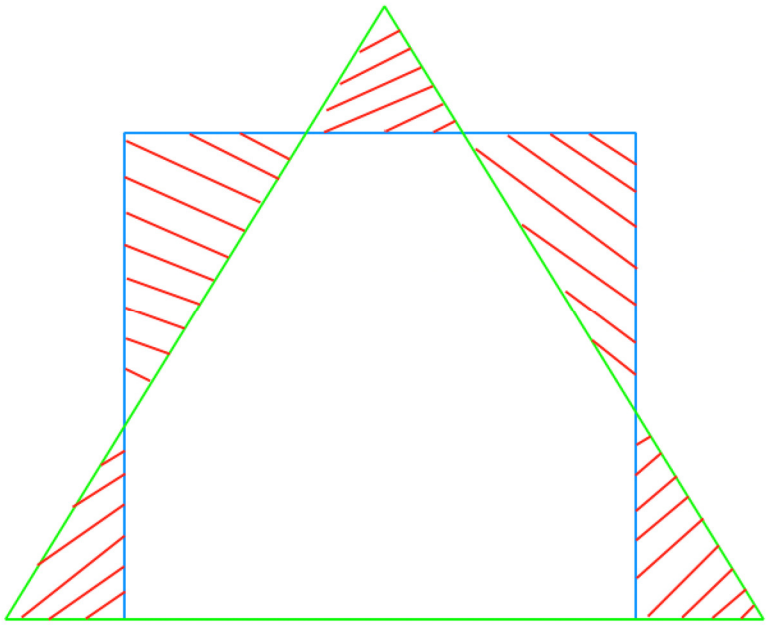
Obrázok 16 zobrazuje tvar konštruovaného dielu medzi odchýlkovými bodami z obrázku 15. Čiernou farbou sú zobrazené odchýlkové body, ktoré tvoria kompletné trajektórie (červená priamka spájajúca tieto body). V tomto pomocnom zobrazovacom nástroji pre veľkostné vrstvy budú šedou farbou zobrazované zlomové body a krivkové body budú zobrazované žltou farbou.

1.7. Globálna optimalizácia

„Jednoducho povedané, optimalizácia je pokus maximalizovať požadované vlastnosti systému a súčasne minimalizovať jeho nežiadúce vlastnosti. Čo sú tieto vlastnosti a ako efektívne sa dajú vylepšiť, záleží na danom probléme" [6]. Algoritmus firemnej aplikácie cad.assyst, nazývaný globálny optimalizátor rieši kategóriu optimalizácie nekonvexných problémov.

Výsledok hodnotiacej funkcie je založený na súčte odchýlenia dvoch plôch: geometrickej predlohy (zobrazená modrou farbou) a pracovnej verzie závislého šva

(zobrazená zelenou farbou). Odchýlkové plochy sú zobrazené červeným šrafovaním, (pozri obrázok č.17).



Obr. č.17 – plochy odchýlenia

Súčtom takýchto plôch získame hodnotu, pomocou ktorej vieme povedať na koľko sa približujeme pracovnou verziou závislého šva ku geometrickej predlohe. Z toho vyplýva, že čím menšie odchýlkové plochy dosahujeme, tým menšiu hodnotu dostávame, a teda sa približujeme ku geometrickej predlohe.

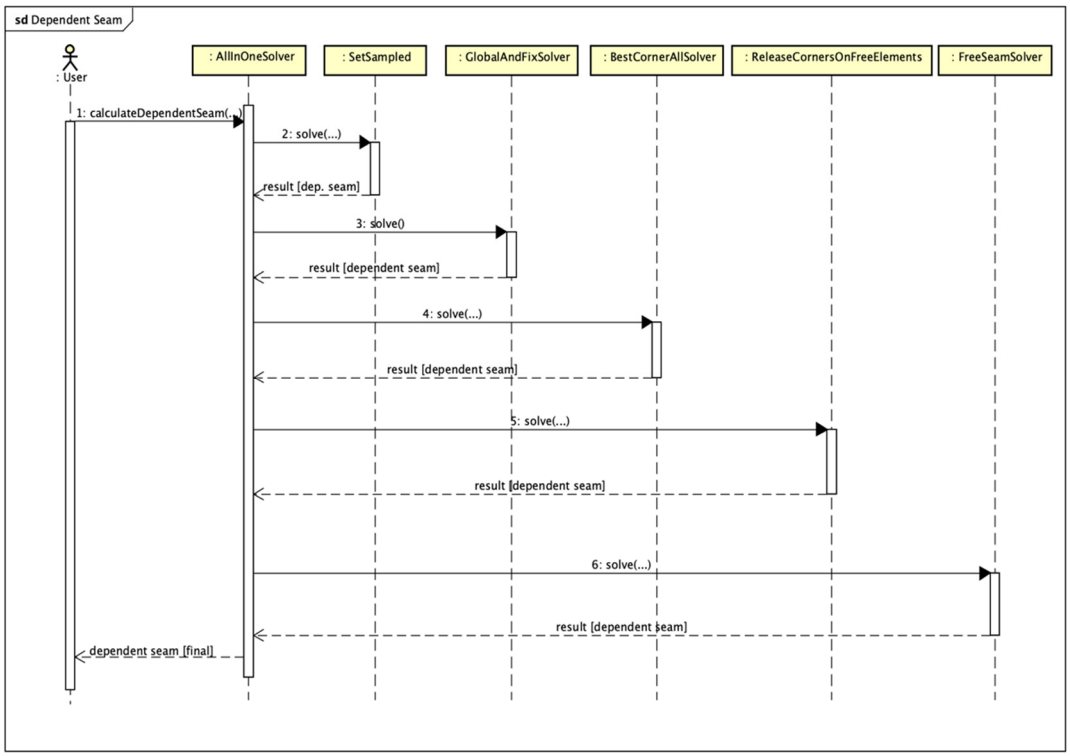
Globálny optimalizátor potrebuje na svoje fungovanie hodnotiacu funkciu, ktorá určí, či sa približujeme k cieľu alebo sa od neho vzďaľujeme. Softvérový modul používa na vyhodnocovanie plochu oblasti, ktorá vznikla medzi dvomi Minkowského sumami (diamant obiehajúci zvnútra a zvonku kontúry), (pozri obrázok č.12). Problém hľadania definície závislého šva je vysoko nekonvexný a preto sa v našom prípade používa práve tento algoritmus na riešenie danej problematiky.

1.8. Súčasný stav rekonštrukcie

Podkapitola súčasného stavu rekonštrukcie popisuje softvérový modul sekvenčným UML diagramom a obsahuje úvodnú analýzu, ktorá vysvetľuje podrobnejšie problematiku, ktorú sa chystáme riešiť.

1.8.1. **Sekvenčný diagram**

"Sekvenčné diagramy modelujú interakcie medzi objektami. Rovnako ako diagramy aktivít predstavujú aj procesy, ale zameriavajú sa na výmenu správ a nie na prezentáciu všetkých možných procesných ciest" [9].



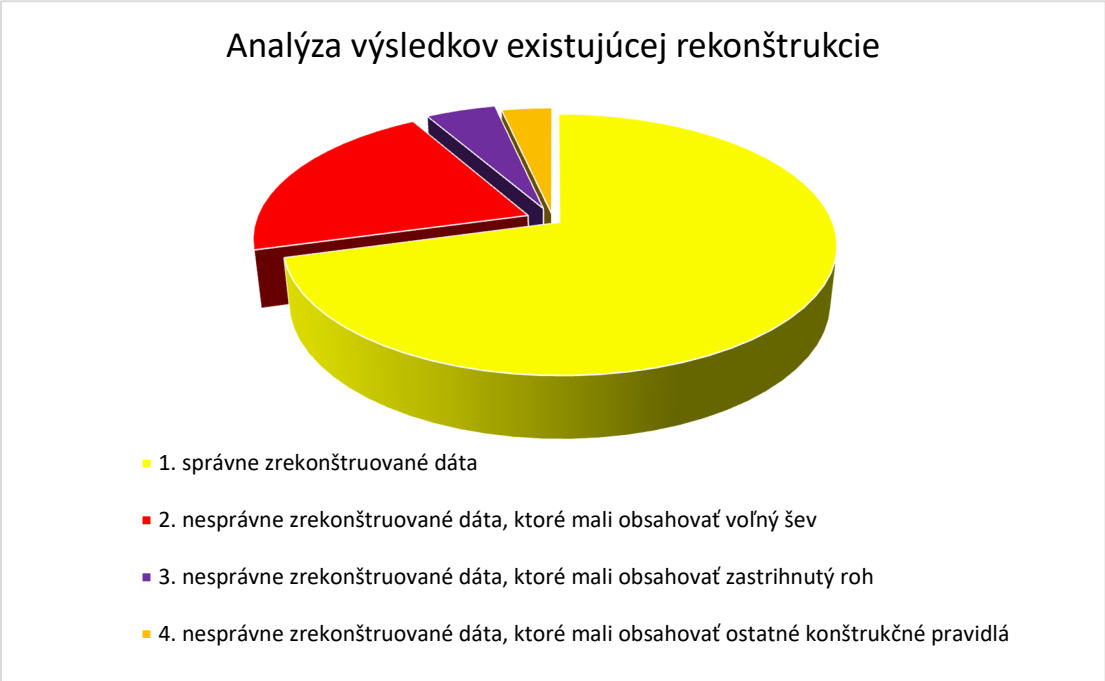
Obr. č.18 – sekvenčný diagram zobrazujúci vytvorenie závislého šva

Na obrázku 18 je zobrazený scenár, kde používateľ zadá požiadavku na spočítanie závislého šva podľa danej geometrickej predlohy. Po vytvorení požiadavky na spočítanie sa zavolá algoritmus AllInOneSolver, ktorý následne v sekvencii krokov volá jednotlivé algoritmy, ktoré aplikujú konštrukčné pravidlá a tým rozširujú a upravujú pracovnú verziu závislého šva s cieľom dosiahnuť ekvivalentnú podobu ku geometrickej predlohe. Algoritmus SetSampled analyticky aplikuje paralelné a neparalelné konštrukčné pravidlo a švový schodík na elementoch. Algoritmus GlobalAndFix aplikuje paralelné konštrukčné pravidlá na elementoch a používa globálnu optimalizáciu, kde sa snaží priblížiť najmenšej hodnote, ktorú dostane z hodnotiacej funkcie. Algoritmus BestCornerAllSolver analyticky aplikuje pravidlá typu zastrihnutý roh, zrkadlený roh a kolmý roh. Algoritmus ReleaseCornersOnFreeElements sa pozerá na rohové elementy, ktorým nebolo priradené

žiadne konštrukčné pravidlo a pri zlepšení hodnoty, ktorú dostaneme z hodnotiacej funkcie sa toto pravidlo použije. Na konci sa volá FreeSeamSolver, ktorý doplní do pracovnej verzie závislého šva v miestach medzi bodmi odchýlky (nezávislý) voľný šev, ktorý vystrihne z predlohy. Sú to miesta, ktoré sa nedali vytvoriť pomocou ostatných konštrukčných pravidiel iných ako voľný šev. A zároveň vráti závislú švovú kontúru, ktorá je v tolerancii k pôvodnej predlohe.

1.8.2. **Analýza problematiky**

Kompletná analýza dát vychádza z množiny 300 konštruovaných dielov, (pozri obrázok č.19).



Obr. č.19 – úvodná analýza dát

Úvodnou analýzou sme sa dopracovali k poznatkom, že najväčšia časť problémových prípadov nastáva práve pri vytváraní konštrukčného pravidla voľný šev. Na obrázku 19 vidieť, že $^{2}/_{3}$ dát sa podarilo zrekonštruovať súčasnému softvérovému modulu. Zo zvyšnej tretiny sa ¾ nepodarilo dosiahnuť kvôli konštrukčnému pravidlu voľný šev. Zo zostávajúcej ¼ dát sa $^{2}/_{3}$ nepodarilo dosiahnuť kvôli konštrukčnému pravidlu zastrihnutý roh. Ostávajúcu skupinu dát tvoria ostatné konštrukčné pravidlá.

2. Návrh

V kapitole návrhu sa venujeme opisu navrhovaných modulov, ktoré riešia jednotlivé skupiny problémových dát. Najskôr rozoberieme prečo súčasný softvérový modul nedokázal vyriešiť tieto skupiny dát a následne navrhneme vlastné riešenie problému. Navrhnuté moduly musia spĺňať softvérové požiadavky firmy.

Vychádzame z úvodnej kompletnej analýzy dát, (pozri obrázok č.19), kde môžeme vidieť, že najväčšiu skupinu problémových dát tvorí konštrukčné pravidlo voľný šev. Nasledujúca skupina je tvorená konštrukčným pravidlom zastrihnutý roh. Z toho vyplýva, že hlavný dôraz budeme klásť práve na tieto dve skupiny dát.

Špecifikácia požiadaviek pre softvérové moduly v jednotlivých bodoch:

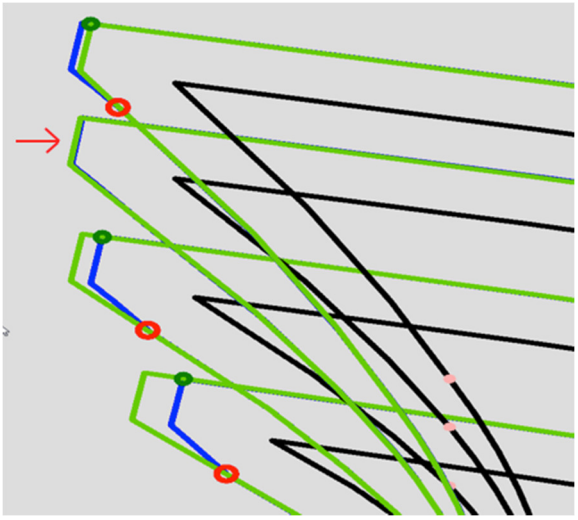
- Navrhnúť softvérový modul, ktorý vyrieši najväčšie problémové skupiny dát s konštrukčnými pravidlami voľný šev, zastrihnutý roh.
- Naimplementovať časovo nenáročné algoritmy, ktoré budú použiteľné súčasným softvérovým modulom.
- Vytvoriť konštrukčné pravidlo voľný šev na potrebných miestach
- Zjednotiť voľno-švové oblasti v bezprostrednej blízkosti vedľa seba
- Doplniť zlomové body spolu s trajektóriami, na miestach, ktoré boli odignorované
- Správne nájsť rohové elementy, na ktorých sa má použiť konštrukčné pravidlo zastrihnutý roh
- Konštrukčnému pravidlu zastrihnutý roh prednastaviť geometrické parametre, aby globálny optimalizátor dokázal dokončiť výpočet a dostal sa ku geometrickej predlohe

2.1. Voľný šev

Problémovú skupinu dát s voľným švom rozdelíme do dvoch podkapitol. V prvej podkapitole budeme riešiť problém, ktorý vznikol nerovnakým počtom odchýlkových bodov. V druhej podkapitole budeme riešiť problém stratenia zlomových bodov spolu s nekompletnými trajektóriami.

2.1.1. Doplnenie odchýlkových bodov

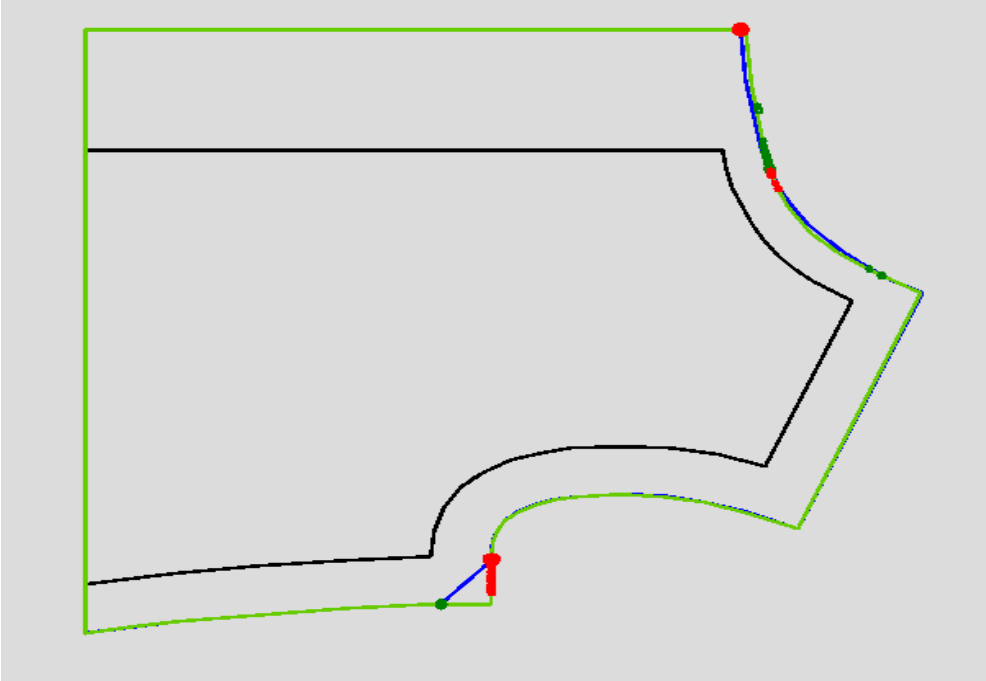
V prvej podkapitole riešime problém, kde v niektorých stupňovaných veľkostiach zmizli body odchýlky a teda nastal stav, kedy nemáme rovnaký počet bodov vo všetkých veľkostiach, (pozri obrázok č.20). Tento problémový prípad nastal, pretože sa použilo konštrukčné pravidlo kolmý roh (pozri kapitolu konštrukčné pravidlá), ale keďže je to konštrukčný diel z iného CAD softvéru, cad.assyst nemá definované špeciálne rohové pravidlo pre takýto druh rohového elementu. V takýchto prípadoch bolo doteraz používané konštrukčné pravidlo voľný šev.



Obr. č.20 – nerovnaký počet odchýlkových bodov

Pôvodný softvérový modul predpokladal rovnaký počet bodov odchýlky naraz vo všetkých veľkostiach a preto takúto skupinu dát odignoroval. Takýto jav nastal v momente, kedy pracovná verzia závislého šva (ilustrovaná zelenou farbou) bola v niektorej zo stupňovaných veľkostí v tolerancii (miesto zobrazuje červená šípka). To spôsobilo, že sa

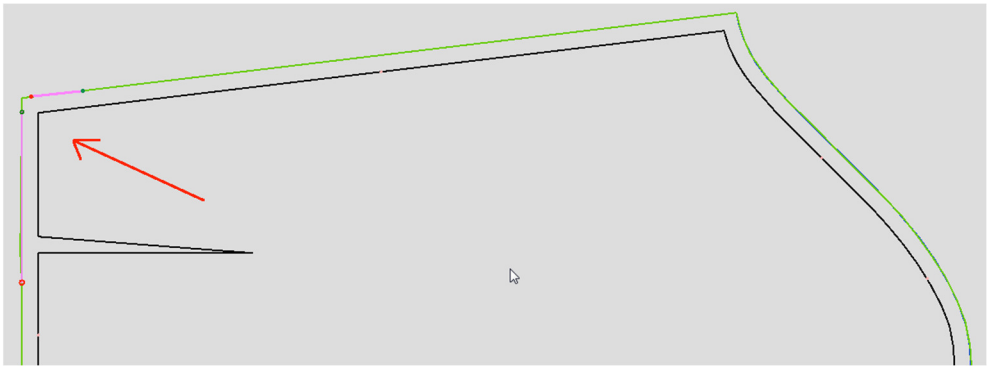
v takejto veľkosti nevytvorili body odchýlky (zobrazené ako zelené a červené body), (pozri obrázok č.20).



Obr. č.21 – rozkopírovanie bodov odchýlky

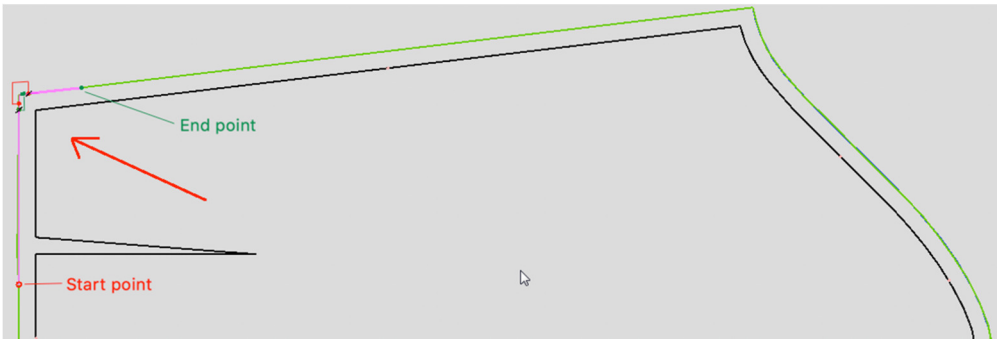
Tento problém sme navrhli riešiť kopírovacím modulom, ktorým rozkopírujeme body zo všetkých veľkostí do základnej veľkosti, aby sme dokázali zachytiť aj také úseky, ktoré neobsahujú body odchýlky (body ilustrované červenou a zelenou farbou), (pozri obrázok č.21). Tieto body odchýlky sú zakódované na švovej kontúre a preto budeme potrebovať vytvoriť transformačný modul, ktorým premapujeme jednotlivé body na hlavnú kontúru.

Pri rozkopírovaní odchýlkových bodov môže vzniknúť situácia, kedy sa vytvoria voľno-švové oblasti vedľa seba. Podľa softvérovej špecifikácie musíme predísť vzniknutému problému, teda vytváraniu voľno-švových oblastí (ilustrované ružovou farbou) na elementoch v bezprostrednej blízkosti vedľa seba, (pozri obrázok č.22).



Obr. č.22 – voľno-švové oblasti v bezprostrednej blízkosti

Tento problém sme navrhli riešiť posunutím rozkopírovaných bodov na švovej kontúre a to tým spôsobom, že miesto, na ktorom pracovná verzia závislého šva vychádza z tolerancie posunieme smerom doľava a miesto, v ktorom vchádza do tolerancie posunieme smerom doprava, (na miesto posunu ukazuje červená šípka), (pozri obrázok č.23)



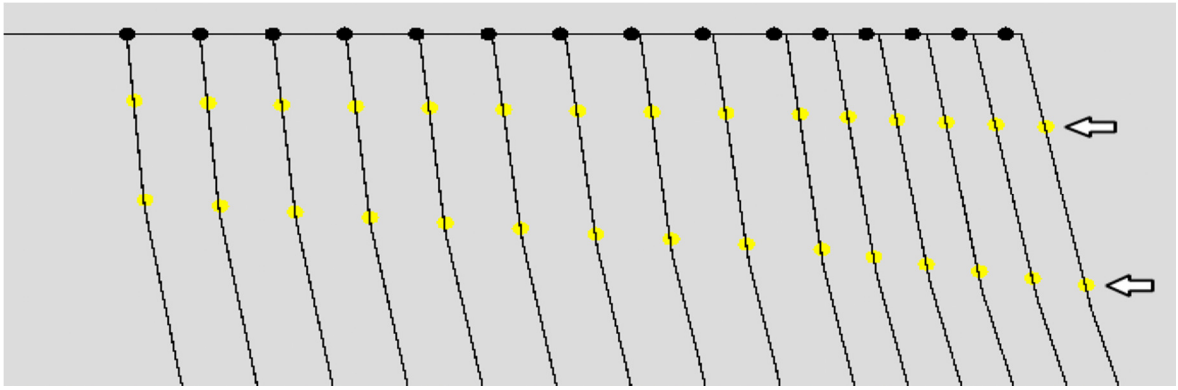
Obr. č.23 – zjednotenie odchýlkových bodov posunutím

Týmto spôsobom zabezpečíme, že sa body odchýlok zjednotia. Posunutím o konštantu zabránime vytváraniu malých voľno-švových oblastí na elementoch v bezprostrednej blízkosti vedľa seba. Pri takomto posune ale musíme dávať pozor, aby sme vybrali správny úsek, teda začiatok a koniec zjednotených oblastí, (‘start point’ ako bod začiatku a ‘end point’ ako bod konca), preto si vytvoríme funkciu hľadania začiatku voľno-švových oblastí.

Na záver budeme potrebovať zjednocovací modul, ktorým odprojektované odchýlkové body spojíme a teda vytvoríme voľno-švové oblasti vo všetkých veľkostiach stupňovaného dielu.

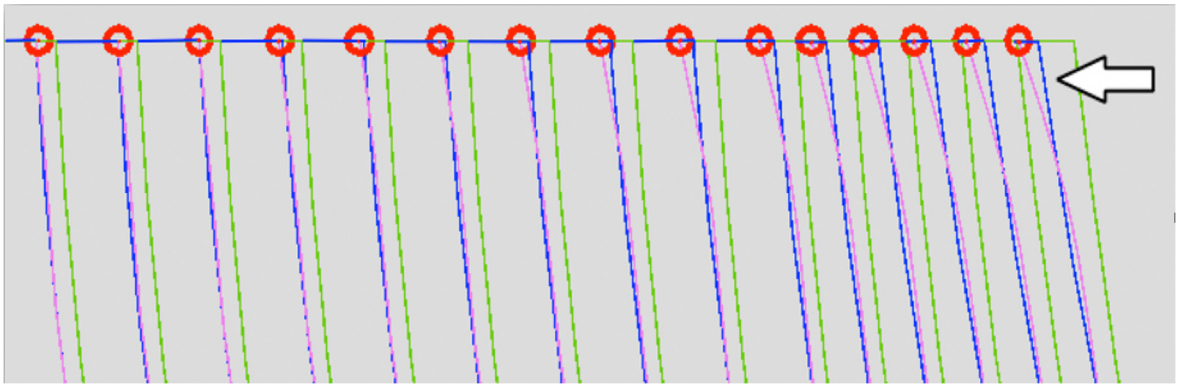
2.1.2. **Doplnenie trajektórií**

V ďalšej časti softvérový modul spracováva švové elementy a posiela ich modulu cad.assyst, nazvanému ‘spline-line-solver‘, ktorý vráti výsledný závislý švový element. ‘Spline-line-solver‘ sa snaží správne aproximovať kontúru aplikovaním najmenšieho počtu krivkových bodov, aby sa priblížil k tolerancii, (pozri obrázok č.24).



Obr. č.24 – doplnenie krivkových bodov pomocou ‘spline-line-solveru’

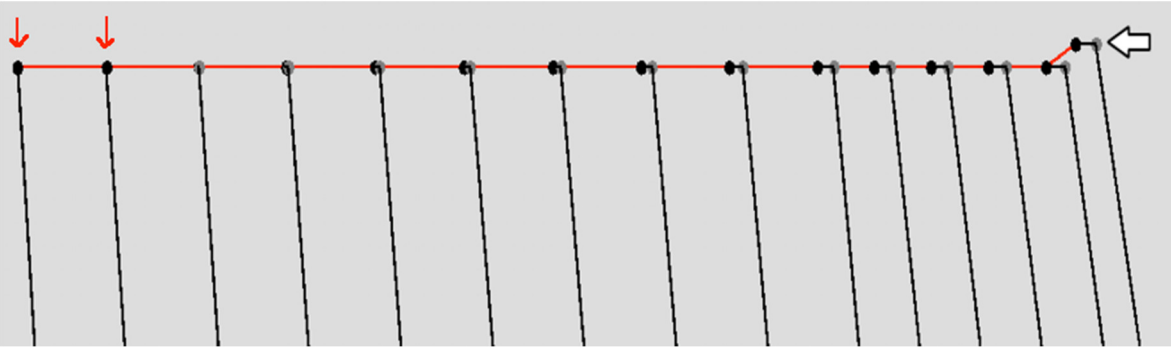
Na obrázku 25 vidíme, že v tomto prípade aplikovaním krivkových bodov vytvoril oblúk (zobrazený šípkou), ktorý je v tolerancii ku geometrickej predlohe (zobrazená modrou farbou). Voľný šev (zobrazený ružovou farbou) má tvar oblúku, pretože odchýlkový bod (zobrazený červenou farbou) v prvých dvoch veľkostiach leží na pracovnej verzii závislého šva (zobrazená zelenou farbou) a nie na geometrickej predlohe (zobrazená modrou farbou), (detailnejšie to zobrazuje obrázok č.26).



Obr. č.25 – chýbajúci zlomový bod

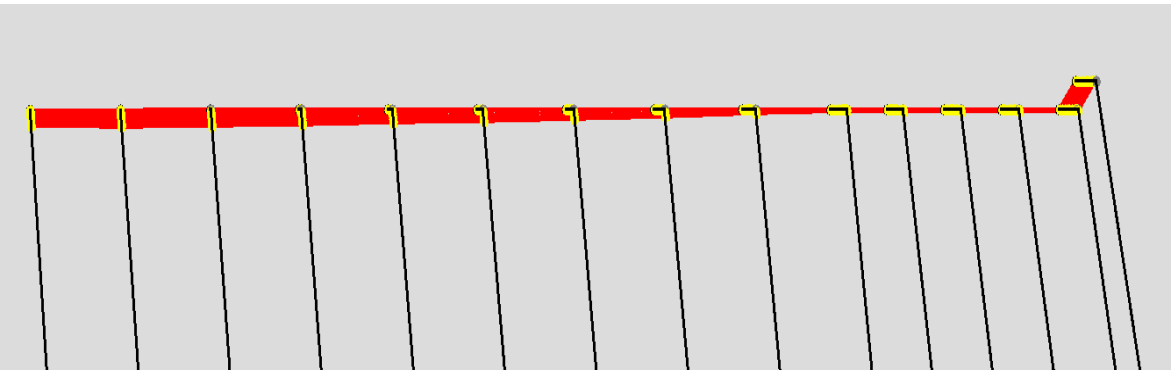
Existujú však prípady, kde pomocou aplikovania krivkových bodov nedokáže vytvoriť spline, ktorý je v tolerancii.

V druhej podkapitole preto budeme riešiť problém, ktorý vznikne pri riešení jednotlivých úsekov voľno-švovej kontúry, ktoré nedokáže modul ‘spline-line-solver‘ vyriešiť.



Obr. č.26 – zlomový bod, ktorý neobsahuje kompletné trajektórie

Na obrázku 26 vidieť, že v prvých dvoch veľkostiach sa stratili zlomové body (miesto stratenia zlomových bodov, ktoré sú označené šedou farbou je označené červenými šípkami) a preto zlomové body, na ktoré ukazuje biela šípka neobsahujú kompletnú trajektóriu. (Kompletné trajektórie sú ilustrované červenou priamkou.) Aby ‘spline-line-solver‘ dokázal správne pracovať potrebuje rovnaký počet podelementov. Z toho vyplýva, že potrebujeme doplniť chýbajúci zlomový bod a spojiť ho s existujúcimi zlomovými bodmi, aby sme vytvorili kompletnú trajektóriu.

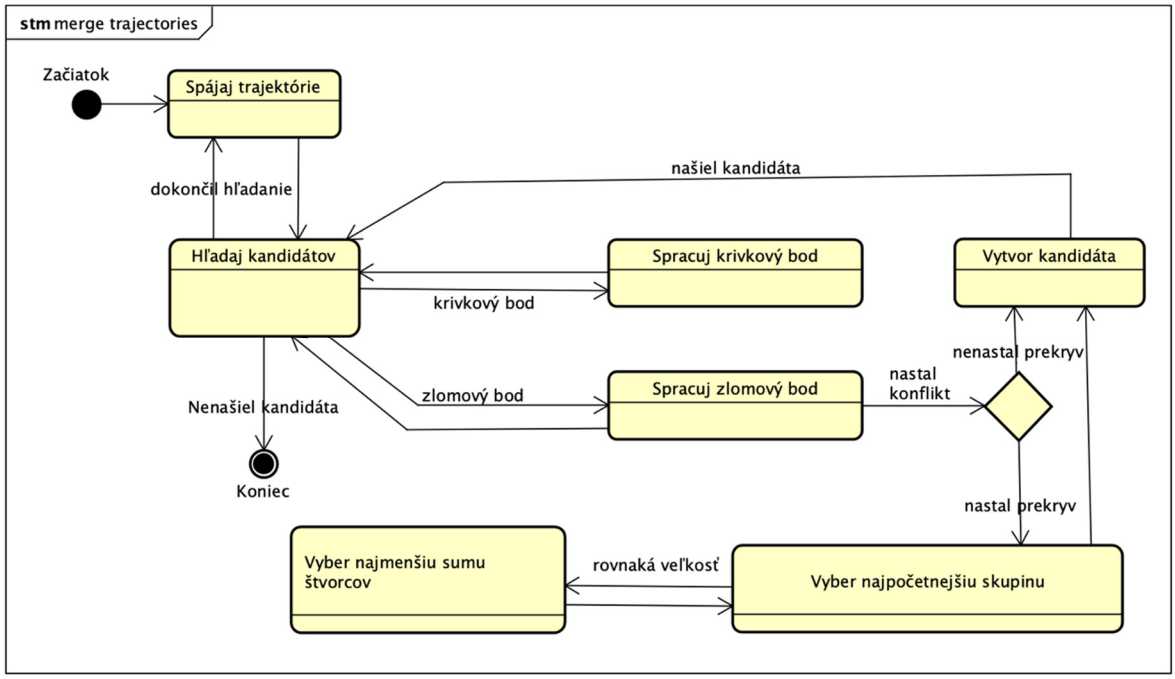


Obr. č.27 – doplnenie trajektórií pre každý zlomový bod

Na riešenie tohto problému budeme musieť vytvoriť softvérový modul „TrajectoryCompletizer“, pomocou ktorého rozkopírujeme zlomové body a doplníme

trajektórie. Projektujeme zlomové body, ktoré sa nachádzajú medzi dvoma kompletnými trajektóriami teda v oblasti voľného šva. Každý samostatný zlomový bod kopírujeme pomocou pomeru na danom elemente do ostatných veľkostí a označíme ho ako krivkový bod. Z projektovanej veľkosti ostane tento bod ako zlomový a v ostatných si ho uložíme ako krivkový bod. Týmto spôsobom si budeme udržiavať informáciu, o tom z akej veľkosti daný bod elementu pochádza. Na konci ho spojíme s ostatnými novovytvorenými krivkovými bodmi a tým mu vytvoríme kompletnú trajektóriu. Týmto spôsobom zaručíme, že doplnené krivkové body s vytvorenými trajektóriami sa nikdy neprekrížia, (pozri obrázok č.27).

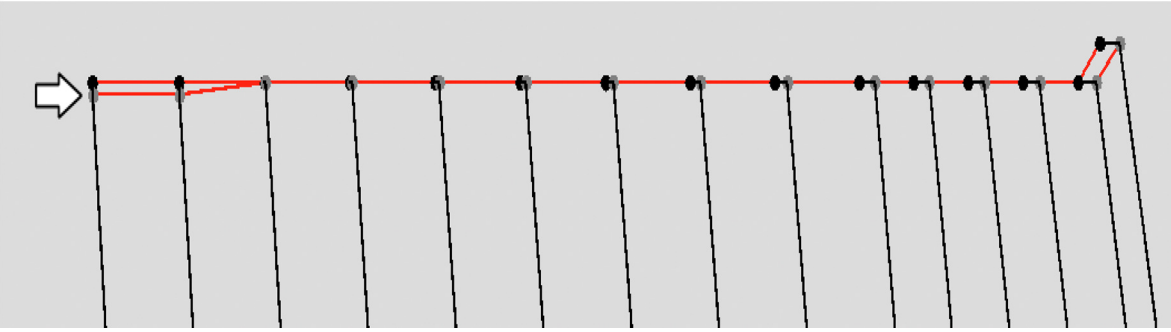
Nasleduje proces spájania, kde použijeme techniku „zametacej priamky“, pomocou ktorej budeme spájať novovytvorené trajektórie. V prvom kroku nájdeme všetkých potenciálnych kandidátov na spájanie.



Obr. č.28 – stavový diagram zobrazujúci hľadanie kandidátov

Stavový diagram obrázok 28, reprezentuje doménu, v ktorej hľadáme kandidátov obsahujúcich kompletné trajektórie. Proces hľadania kandidátov pozostáva zo spracovania zlomových a krivkových bodov na jednotlivých úrovniach veľkostí. Akonáhle prichádzame na konflikt v niektorej stupňovanej veľkosti vytvoríme kandidáta. Pri vytváraní nového kandidáta musíme kontrolovať či nenastáva prekryv s existujúcimi kandidátmi. Ak nastane problém s prekrývaním kandidátov, tak vyberáme kandidáta, ktorý je tvorený

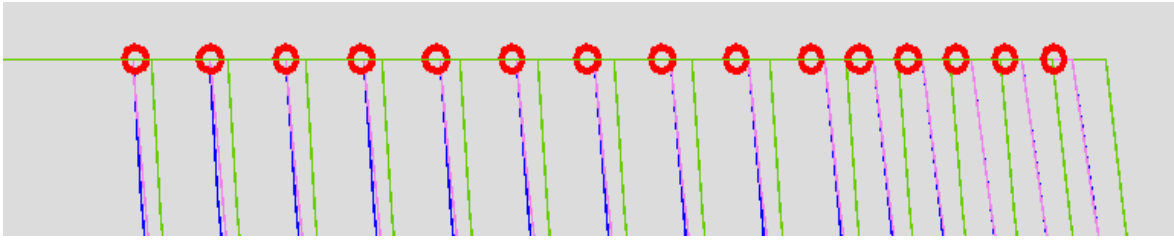
najpočetnejšou skupinou trajektórií. Ak však máme rovnaký počet rozhodujeme podľa metódy najmenšej sumy štvorcov veľkosti trajektórií. Následne kandidáta uložíme a pokračujeme v hľadaní ďalej. Na konci hľadania si z kandidátov spojením vytvoríme kompletné trajektórie a krivkové body zmeníme na zlomové. Proces opakujeme, pokiaľ nám funkcia hľadania nenájde žiadneho vhodného kandidáta, teda ak nenastane žiadny konflikt.



Obr. č.29 – výsledný stav po spájaní trajektórií

Týmto postupom sme doplnili chýbajúce zlomové body, (na miesto doplnenia zlomových bodov ukazuje biela šípka), ktorým sme doplnili chýbajúce trajektórie, (pozri obrázok č.29).

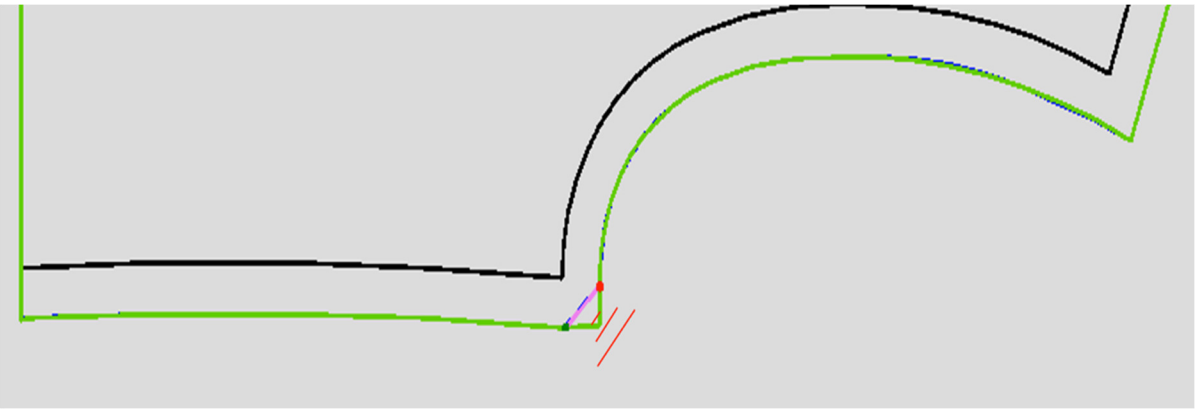
Po doplnení trajektórií nemusí ‘spline-line-solver’ dopĺňať krivkové body a teda nevytvorí oblúk na miestach, kde mu chýbali zlomové body čiže voľný šev má tvar geometrickej predlohy, (pozri obrázok č.30).



Obr. č.30 – výsledný stav po doplnení chýbajúcich trajektórií

2.2. Zastrihnutý roh

Druhá skupina problémových dát je tvorená konštrukčným pravidlom zastrihnutý roh. Toto rohové pravidlo je špecifikované vzdialenosťou priamky a uhlom od rohového bodu na hlavnej kontúre. Súčasný softvérový modul používa na toto pravidlo globálny optimalizátor, ktorý prednastavuje geometrické hodnoty zastrihnutému rohu, teda uhol a vzdialenosť od rohového bodu na hlavnej kontúre. Po prednastavení parametrov sa určí hodnota podľa hodnotiacej funkcie. Globálny optimalizátor sa riadi výsledkom hodnotiacej funkcie.

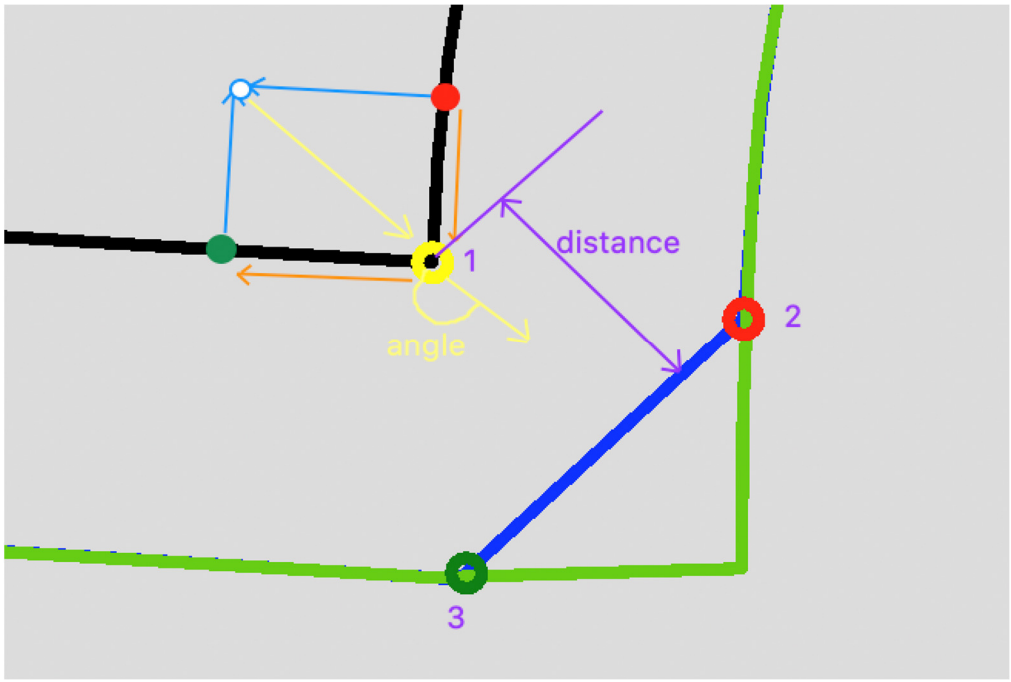


Obr. č.31 – prednastavovanie geometricky nesprávnych hodnôt

Na obrázku 31 sa použilo konštrukčné pravidlo voľný šev (ilustrovaný ružovou farbou), pričom správnym výsledkom rekonštrukcie malo byť použite pravidla zastrihnutý roh. Prednastavením nesprávnych hodnôt (ilustrované červenou farbou), ktoré sú odchýlené od pôvodnej geometrickej predlohy sa globálny optimalizátor nedokáže priblížiť k menšej hodnote, ktorej výsledok je určený podľa hodnotiacej funkcie, a preto nedokáže prísť na správny výsledok. Musíme navrhnúť riešenie, aby sme geometricky prednastavili správne geometrické parametre zastrihnutému rohu. Následne pomocou hodnotiacej funkcie dostaneme menšie hodnoty a globálny optimalizátor sa za pomoci menších hodnôt dokáže priblížiť k cieľovej geometrickej predlohe, teda nulovému výsledku hodnotiacej funkcie. Nulový výsledok hodnotiacej funkcie znamená, že pracovná verzia závislého šva je ekvivalentná s geometrickou predlohou.

Obr. č.32 – zobrazenie vhodného kandidáta pre použitie rohového konštrukčného pravidla

Aby sme zistili, že môžeme použiť rohové pravidlo, musíme si vytvoriť softvérový modul „TryCutCornerSolver“, ktorým skontrolujeme, že sa nachádzame na rohu dvoch elementov. Keďže body odchýlok sú zakódované na švovej kontúre budeme musieť tieto body z elementu na švovej kontúre prepočítať podľa percentuálneho pomeru na element hlavnej kontúry. Za rohového kandidáta budeme považovať taký element, o ktorom zistíme, že vychádzajúci bod odchýlky sa nachádza v rozmedzí (80%, 100%) elementu hlavnej kontúry a vchádzajúci bod odchýlky sa nachádza v rozmedzí (0%, 20%) nasledujúceho elementu hlavnej kontúry. Z toho vyplýva, že výsledok rozdielu indexov takýchto elementov sa rovná jednej. Vtedy môžeme predpokladať, že sme našli vhodného kandidáta rohového elementu, (pozri obrázok č.32). Následne si uložíme rohového kandidáta ako odchýlkové body na elementoch švovej kontúry spolu s indexom rohového elementu.



Obr. č.33 – výpočet osi uhla a vzdialenosti pre zastrihnutý roh

Vieme, že konštrukčné pravidlo zastrihnutý roh je špecifikované vzdialenosťou priamky a uhlom od rohového bodu na hlavnej kontúre. Keďže sme našli rohového kandidáta môžeme pomocou geometrického vektora vypočítať uhol aj vzdialenosť akou bude zastrihnutý roh vytvorený, (pozri obrázok č.33).

Keďže poznáme rohový bod (označený žltou farbou) a odchýlkové body na hlavnej kontúre, (ktoré sú ilustrované červenou a zelenou farbou) vieme si vypočítať jednotkové vektory. Pomocou vektorov si vypočítame bod osi uhla (ilustrovaný biely bod). Podľa bodu osi uhla a rohového bodu si vieme vypočítať vektor osi uhla. Pomocou odchýlkových bodov na švovej kontúre si vypočítame čiarový vektor. Výsledný uhol vypočítame ako rozdiel uhlov čiarového vektora a vektora osi uhla. Vzdialenosť vypočítame pomocou odchýlkových bodov na švovej kontúre a rohového bodu.

Vypočítané geometrické parametre prednastavíme zastrihnutému rohu. Následne zavoláme globálny optimalizátor, aby dokončil výpočet a dosiahol tvar geometrickej predlohy.

3. Implementácia

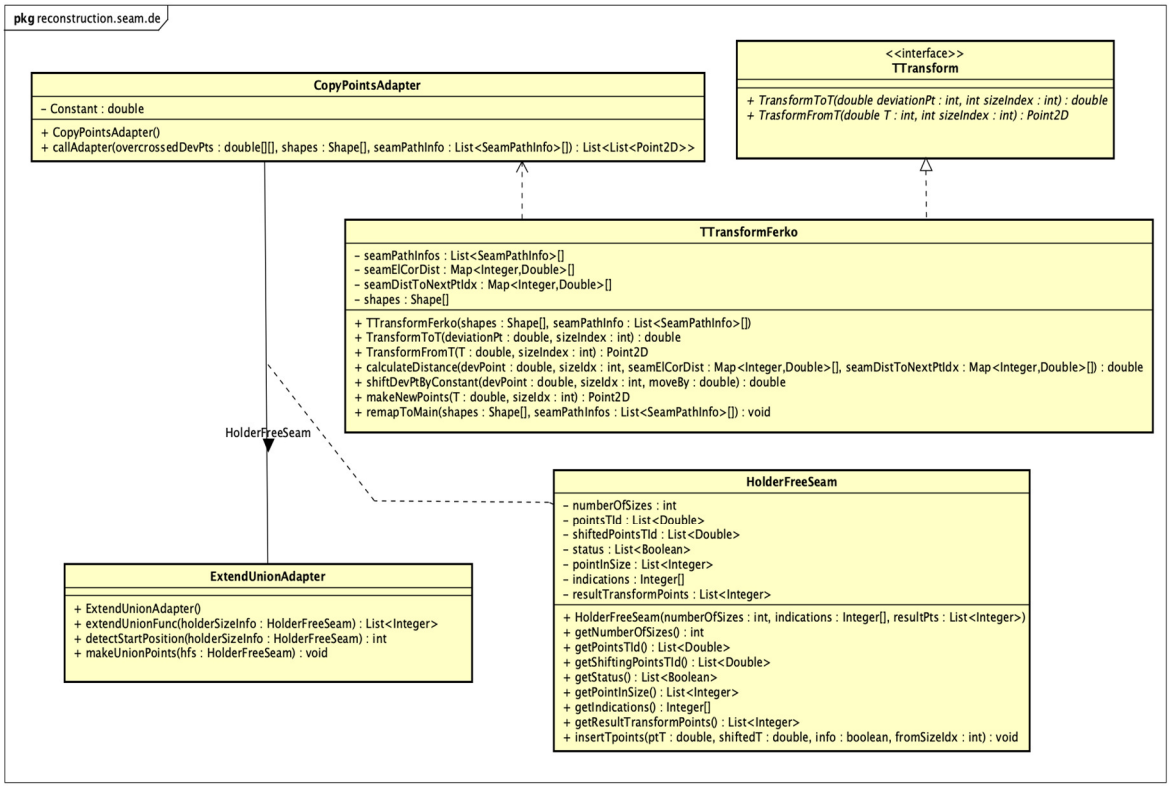
V kapitole implementácie sa venujeme podrobnejšie implementácii jednotlivých softvérových modulov, ktoré budú v súlade s návrhom. Kapitola implementácie rozoberá implementáciu konštrukčných pravidiel: voľný šev a zastrihnutý roh, ktorý tvorili najväčšiu skupinu problémových prípadov.

Softvérové moduly budeme implementovať v programovacom prostredí eclipse použitím programovacieho jazyka java, pretože aplikácia cad.assyst je implementovaná práve v tomto jazyku. Naše softvérové dielo nie je dostupné verejne, lebo je to firemný kód. Jeho funkčnosť som pripravený demonštrovať oponentovi práce.

3.1. Implementácia konštrukčného pravidla voľný šev

Pomocou UML triednych diagramov popisujeme vizuálnu reprezentáciu, štruktúru modulu a vzťahy medzi triedami.

3.1.1. Proces doplnenia odchýlkových bodov



Obr. č.34 – diagram tried zobrazujúci doplnenie odchýlkových bodov

Na obrázku 34 sú zobrazené štrukturálne informácie softvérového modulu, ktorý rieši problém odchýlkových bodov. Trieda ‘CopyPointsAdapter‘ zabezpečuje, aby sa pretransformované body odchýlok pomocou triedy ‘TTransformFerko‘ združili zo všetkých veľkostí do ‘java.utils.Arraylistu‘. Trieda ‘TTransformFerko‘ implementuje funkcie interfejsu ‘TTransform‘. Funkcie tejto transformácie majú za úlohu prepočítať odchýlkové body do takzvaného T čísla, ktoré predstavuje číslo elementu na hlavnej kontúre a jeho percentuálnu časť. Podľa softvérovej špecifikácie máme zabrániť vytváraniu voľno-švových oblastí v bezprostrednej blízkosti a teda použijeme algoritmus, ktorý nám pretransformované body odchýlky posunie o zadanú konštantu, (pozri obrázok č.35).

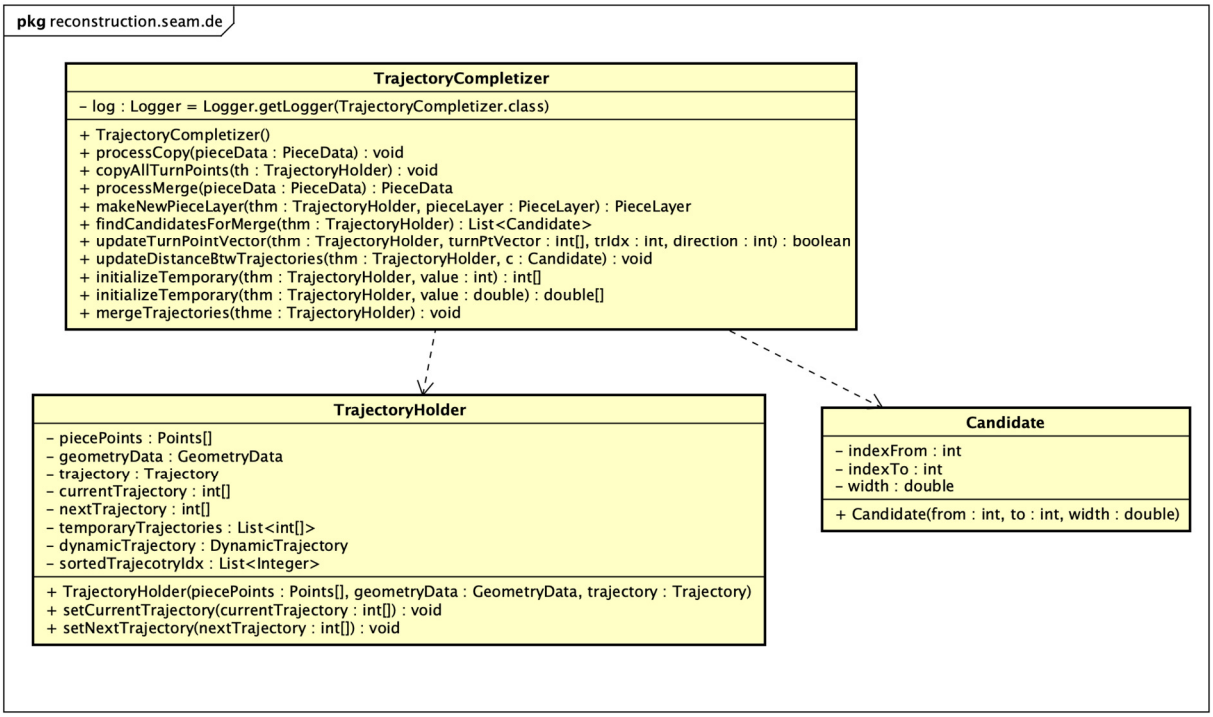
```
public double shiftDevPtByConstant(double devPoint, int sizeIdx, double moveBy) {
    double newRatio = -1;
    double moveByAbs=Math.abs(moveBy);
    int devPtIdx = (int)Math.floor(devPoint);
    double remRatio = devPoint-devPtIdx;
    if(Math.signum(moveBy)>=0d) remRatio=1-remRatio;
    int numPolyPts=seamDistToNextPtIdx[sizeIdx].size();
    while(newRatio < 0) {
        double subElLen=seamDistToNextPtIdx[sizeIdx].get((devPtIdx)%numPolyPts); // special case, last element, use modulo
        double remLength=subElLen*remRatio; // remaining length
        if (remLength>=moveByAbs) {
            newRatio=(remLength-moveByAbs)/subElLen;
            if (Math.signum(moveBy)>=0d) newRatio=1-newRatio;
        } else {
            moveByAbs-=remLength;
            if (Math.signum(moveBy)>=0) devPtIdx++;
            else {
                devPtIdx--;
                if(devPtIdx<0) devPtIdx += numPolyPts;
            }
            remRatio=1d;
        }
    }
    double result=(double)((devPtIdx)%numPolyPts)+newRatio; // special case, handle if devPtIdx==0, use modulo
    return result;
}
```

Obr. č.35 – zobrazuje algoritmus posunu odchýlkových bodov o konštantu

Trieda ‘HolderFreeSeam’ je kompozícia, ktorá zabezpečuje komunikáciu medzi triedami ‘CopyPointsAdapter’ a ‘ExtendUnionAdapter’. V triede ‘ExtendUnionAdapter’ si pomocou vlastného komparátora utriedime T čísla (pretransformované odchýlkové body). Nasleduje proces hľadania začiatkov a koncov voľno-švových oblastí ako index bodu začiatku a index bodu konca. Na záver ‘CopyPointsAdapter‘ pomocou transformácie zakódované T čísla pretransformuje do všetkých veľkostí ako odchýlkové body začiatku a konca voľno-švovej oblasti.

Pomocou takejto implementácie dosiahneme body voľno-švových oblastí vo všetkých stupňovaných veľkostiach.

3.1.2. **Proces doplnenia trajektórií**



Obr. č.36 – diagram tried zobrazujúci doplnenie trajektórií

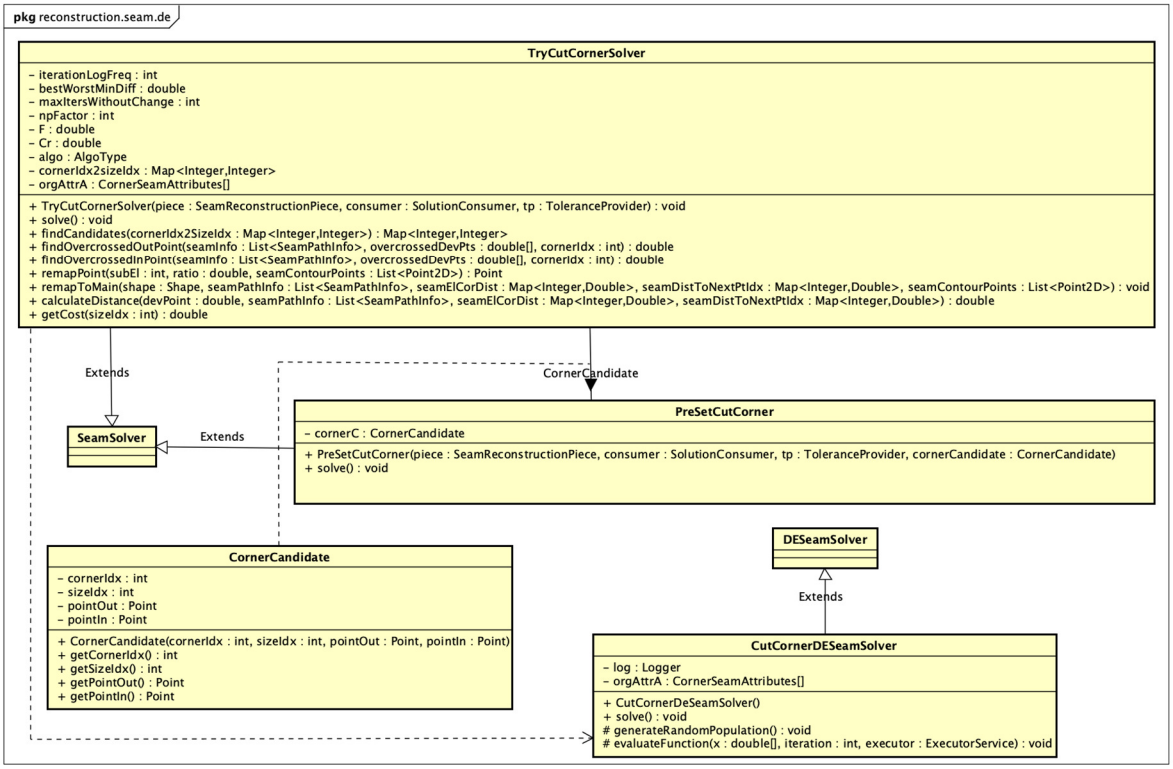
Na obrázku 36 sú zobrazené štrukturálne informácie softvérového modulu, ktorý rieši problém nekompletných trajektórií. Pomocou triedy ‘TrajectoryCompletizer’ riešime doplnenie zlomových bodov a k nim kompletné trajektórie. Funkcia ‘processCopy’ rozkopíruje zlomové body, ktoré nemajú kompletné trajektórie do ostatných veľkostí ako krivkové body v pomere, v akom sa nachádzali na pôvodnom stupňovanom elemente a vytvorí z nich novú kompletnú trajektóriu. Na uloženie informácii používame triedy ‘Candidate’ a ‘TrajectoryHolder’, pomocou ktorých ostatné funkcie komunikujú. Funkcia ‘processMerge’, ktorej cieľom je spájať novovytvorené trajektórie, potrebuje vyhodnocovaciu funkciu ‘findCandidatesForMerge‘, (pozri obrázok č.37).

```
//Function that finds various disjoint subsets
public List<Candidate> findCandidatesForMerge(TrajectoryHolder thm) {
    List<Candidate> groupsToMerge=new ArrayList<Candidate>();
    List<Integer> currentGroup=new ArrayList<Integer>();
    int[] turnPtVector=null;
    boolean conflict=false;
    int idx=0;
    turnPtVector = initializeTemporary(thm, 0);
    while(idx < thm.sortedTrajectoryIdx.size()) {
        int trIdx = thm.sortedTrajectoryIdx.get(idx);
        if (log.isDebugEnabled())
            log.debug("TrIdx:" + idx + " trajectoryIndex:"+trIdx+" | "+currentGroup.toString());
        conflict=updateTurnPointVector(thm, turnPtVector, trIdx, 0); // test for conflict
        if(!conflict) {
            currentGroup.add(idx);
            conflict=updateTurnPointVector(thm, turnPtVector, trIdx, 1); // update turn points vector
            idx++;
        } else { // remove first trajectory
            int from = currentGroup.get(0);
            int to = currentGroup.get(currentGroup.size()-1);
            Candidate c = new Candidate(from, to, -1);
            if (currentGroup.size()>1) { // check for overlap candidates
                int uts=groupsToMerge.size();
                if(uts > 0) {
                    Candidate lastC=groupsToMerge.get(uts-1);
                    int toLast=lastC.indexTo;
                    if (from<=toLast) {
                        int diff=(lastC.indexTo-lastC.indexFrom)-(to-from);
                        if (lastC.width<0d) updateDistanceBtwTrajectories(thm, lastC);
                        updateDistanceBtwTrajectories(thm, c);
                        if (diff<0 || (diff==0 && lastC.width > c.width)) {
                            groupsToMerge.remove(uts-1);
                        } else
                            c=null;
                    }
                }
                if (c!=null) groupsToMerge.add(c);
            }
            // update turn pt vector
            int trId = thm.sortedTrajectoryIdx.get(from);
            conflict=updateTurnPointVector(thm, turnPtVector, trId, -1);
            currentGroup.remove(0);
            conflict=false; // assume no conflict anymore
        }
    }
    return groupsToMerge;
}
```

Obr. č.37 – funkcia hľadania kandidátov

Na obrázku 37 je zobrazený proces hľadania kandidátov vhodných na spájanie. Kandidátov ukladáme ako množinu novovytvorených trajektórií. Po úspešnom nájdení kandidátov, tieto jednotlivé množiny spojíme a vytvoríme novú trajektóriu. Proces hľadania opakujeme, až pokiaľ nám funkcia hľadania vhodných kandidátov nevráti žiadneho kandidáta na spájanie. Na konci sa vytvorí nová veľkostná vrstva, ktorá obsahuje doplnené výsledné zlomové body s kompletnými trajektóriami.

3.2. Implementácia konštrukčného pravidla zastrihnutý roh



Obr. č.38 – diagram tried zobrazujúci pravidlo zastrihnutého rohu

Pomocou UML diagramov opíšeme vizuálnu reprezentáciu systémového modulu spolu s štruktúrou. Na obrázku 38 vidíme, že trieda 'TryCutCornerSolver', ktorá implementuje funkciu 'solve' z nadtriedy 'SeamSolver', v tele funkcie používa metódu 'findCandidates', ktorej cieľom je odchýlkové body premapovať na hlavnú kontúru a zistiť, že odchýlkový bod, ktorý vychádza z tolerancie sa nachádza v rozmedzí (80%, 100%) elementu hlavnej kontúry a vchádzajúci bod odchýlky sa nachádza v rozmedzí (0%, 20%) nasledujúceho elementu hlavnej kontúry. Týmto spôsobom zaručíme, že sa nachádzame na rohovom elemente hlavnej kontúry a môžeme použiť rohové konštrukčné pravidlo. Rohového kandidáta si ukladáme v triede 'CornerCandidate', ktorá obsahuje index elementu na hlavnej kontúre, veľkosť v ktorej sa nachádza a odchýlkové body na švovej kontúre. Táto trieda je kompozícia medzi triedami 'TryCutCornerSolver' a 'PreSetCutCorner'. Trieda 'PreSetCutCorner' dostáva referenciu na rohového kandidáta. Cieľom tejto triedy je v metóde 'solve' geometricky správne prednastaviť hodnoty zastrihnutému rohu, teda vzdialenosť a uhol rohovému elementu na hlavnej kontúre, (pozri obrázok č.39).

```java
@Override
public void solve() {
    int elemIn = cornerC.getCornerIdx();
    int elemOut = (elemIn-1 >= 0) ? elemIn-1 : (elemIn-1)+piece.elementSeamAttributes.length;
    int size = cornerC.getSizeIdx();
    Point2D[] pointsOut = piece.mainContourFineApproximatedElements[size][elemOut];
    Point2D ptoen= pointsOut[pointsOut.length-1];
    Point2D ptost = pointsOut[pointsOut.length-2];
    Point2D[] pointsIn = piece.mainContourFineApproximatedElements[size][elemIn];
    Point2D ptist = pointsIn[0];
    Point2D ptien = pointsIn[1];
    Point startPs = new Point(ptist.getX(), ptist.getY());
    // axis point
    Vector vectorUnit = new Vector(ptost.getX()-ptoen.getX(),ptost.getY()-ptoen.getY(), true);
    Vector vectorUnit2 = new Vector(ptien.getX()-ptist.getX(), ptien.getY()-ptist.getY(), true);
    Point vectorBi = new Point(vectorUnit2.getDx()+vectorUnit.getDx(), vectorUnit2.getDy()+vectorUnit.getDy());
    Point axis = new Point(ptist.getX()+vectorBi.getX(), ptist.getY()+vectorBi.getY());
    Vector bisector = new Vector(ptist.getX()-axis.getX(), ptist.getY()-axis.getY(),true);
    Point pointOut = cornerC.getPointOut(); // red point
    Point pointIn = cornerC.getPointIn(); // green point
    // make line vector
    Vector lineVector = new Vector(pointIn.getX()-pointOut.getX(), pointIn.getY()-pointOut.getY(), true);
    lineVector = Vector.getOrthogonalVector(lineVector, false);
    //make new cut corner
    double distance = Line2D.ptLineDist(pointIn.getX(),pointIn.getY(),pointOut.getX(),pointOut.getY(),startPs.getX(),startPs.getY());
    double angle = (lineVector.getAngle()-bisector.getAngle());
    piece.cornerSeamAttributes[cornerC.getCornerIdx()] = new Cut(distance, Math.toDegrees(angle),0);
}
```

Obr. č.39 – geometrické prednastavovanie parametrov zastrihnutého rohu

Na obrázku 39 vidíme, že pomocou java implementácie ‘java.util.Vector’ si vypočítame geometrické vektory. Keďže máme začiatok rohového elementu a odchýlkové body na hlavnej kontúre, vieme si vypočítať vektory. Pomocou týchto dvoch vektorov dostávame bod osi uhla. Pomocou bodu osi uhla a bodu rohového elementu si vypočítame vektor osi uhla. Java implementáciou ‘java.awt.geom.Line2D’ si vieme pomocou bodu začiatku rohového elementu na hlavnej kontúre a čiarového vektoru vypočítaného z odchýlkových bodov na švovej kontúre vypočítať vzdialenosť priamky od rohového elementu na hlavnej kontúre. Uhol vypočítame ako rozdiel dvoch vektorov: čiarového a vektoru osi uhla. Na koniec vytvoríme zastrihnutý roh na správnom rohovom indexe hlavnej kontúry. Následne pomocou triedy ‘CutCornerDESeamSolver’, ktorá implementuje globálny optimalizátor z nadtriedy ‘DESeamSolver’, dokončujeme prednastavené hodnoty. Určí sa hodnota pomocou hodnotiacej funkcie a porovná sa s hodnotou pred vytvorením konštrukčného pravidla zastrihnutý roh. Ak celková hodnota je menšia, priradený zastrihnutý roh sa ponechá pracovnej verzii závislého šva.

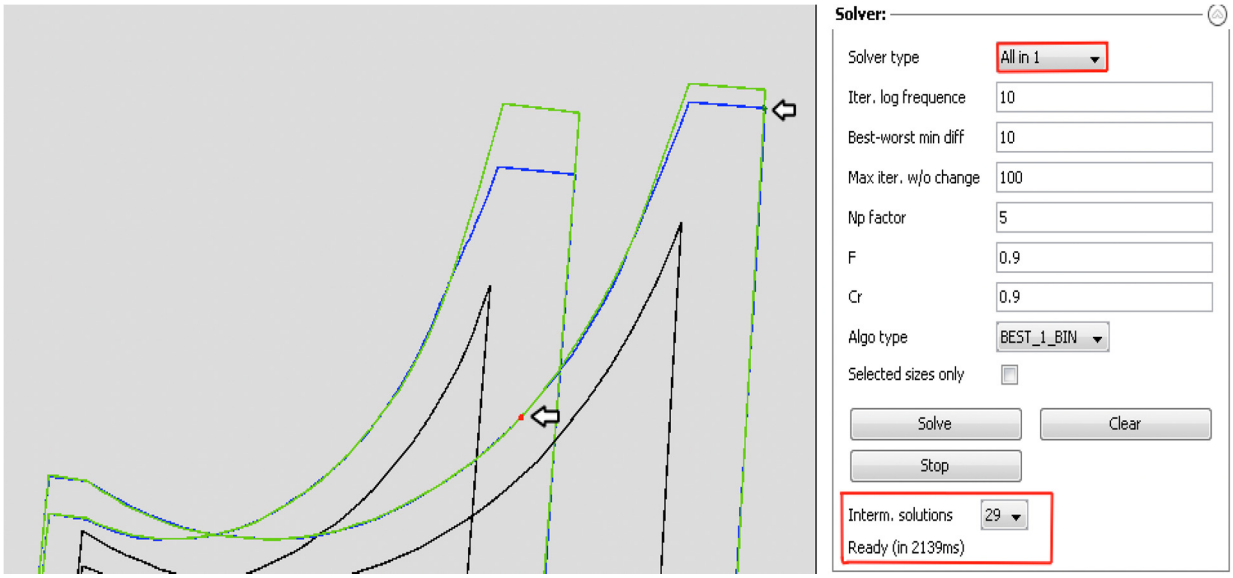
4. Testovanie

V kapitole testovanie sa venujeme testovaniu jednotlivých softvérových modulov, ktoré sme navrhovali a implementovali z najväčších skupín problémových dát.
Kapitola testovania teda obsahuje:

- Obrazovú dokumentáciu pred implementovaním modulov
- Grafické zhodnotenie s popisom pred implementovaním modulov
- Obrazovú dokumentáciu po implementovaní modulov
- Grafické zhodnotenie s popisom po implementovaní modulov
- Celkové zhodnotenie s popisom softvérového modulu s možnosťami ďalšieho vývoja

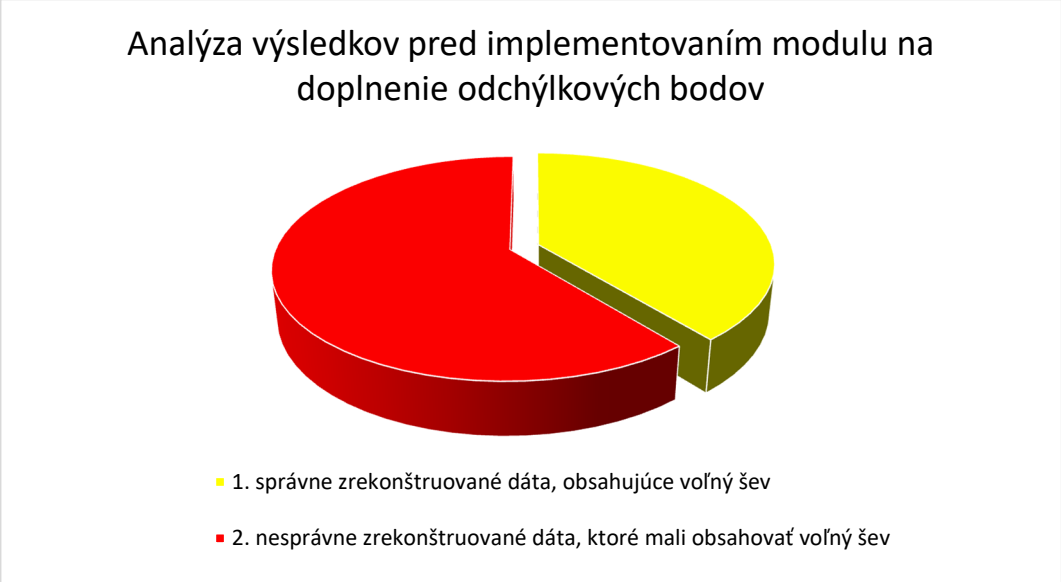
4.1. Testovanie konštrukčného pravidla voľný šev

Obrázok 40 zobrazuje rekonštrukciu závislého šva pred implementovaním doplnenia odchýlkových bodov. Vidíme, že sa použil typ riešenia ‘all in 1’, ktorý bližšie popisuje kapitola súčasný stav rekonštrukcie. Bielymi šípkami sú zobrazené odchýlkové body, ktoré sa v jednej veľkosti vytvorili a v druhej nie. Obrázok zobrazuje počet iterácii a čas, ktoré algoritmus ‘all in 1’ potreboval nato, aby postupne aplikoval konštrukčné pravidlá a tým zmodifikoval pracovnú verziu závislého šva (zobrazená zelenou farbou), aby bola v tolerancii ku geometrickej predlohe (zobrazená modrou farbou). Taktiež vidíme, že pracovná verzia závislého šva nedosiahla aplikovaním štandardných pravidiel tvar geometrickej predlohy a preto musíme použiť na mieste odchýlenia konštrukčné pravidlo voľný šev.



Obr. č.40 – reprezentuje rekonštrukciu závislého šva pred implementáciou doplnenia odchýlkových bodov

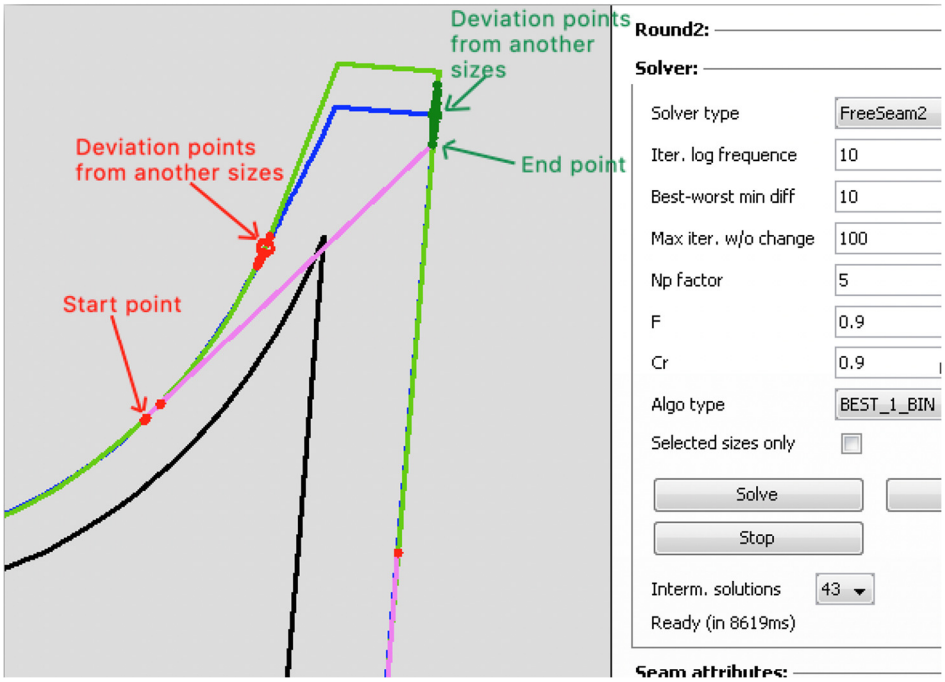
Grafické zhodnotenie pred implementovaním konštrukčného pravidla voľný šev môžeme vidieť na obrázku 41, vychádza z množiny 60 konštruovaných dielov. Vidíme, že $^{2}/_{3}$ dát nebolo správne zrekonštruovaných, pretože súčasný softvérový modul očakával rovnaký počet odchýlkových bodov vo všetkých stupňovaných veľkostiach.



Obr. č.41 – analýza pred doplnením odchýlkových bodov

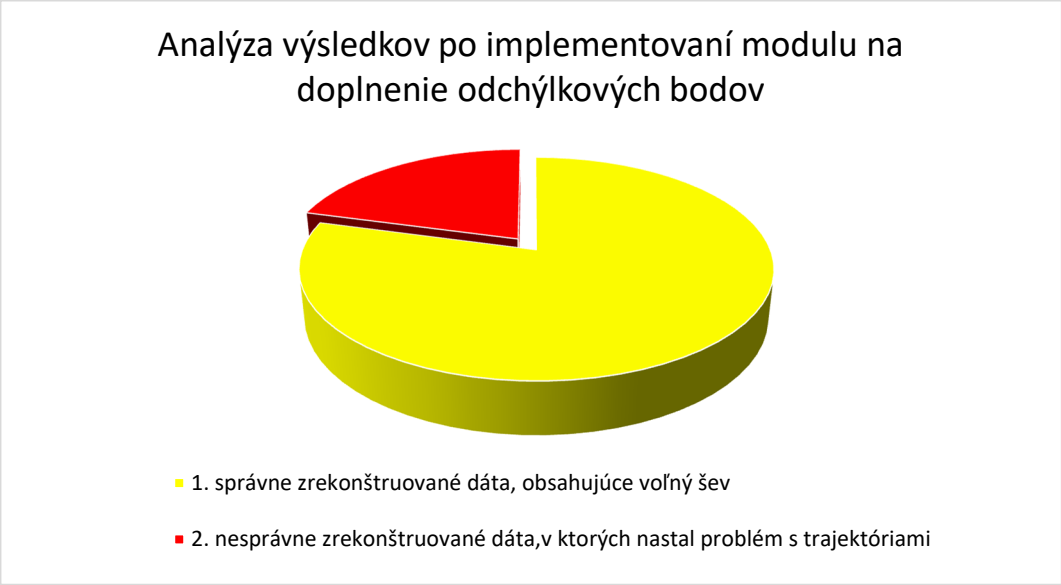
Na obrázku 42 môžeme vidieť rozkopírovanie odchýlkových bodov z ostatných stupňovaných veľkostí a nájdenie správnych bodov začiatku a konca voľno-švových oblastí.

A zároveň vidíme, že po implementovaní modulu na doplnenie odchýlkových bodov "spline-line-solver" pomocou aplikovania najmenšieho počtu krivkových bodov nedokázal správne aproximovať kontúru a vytvoriť spline, ktorý by bol v tolerancii ku geometrickej predlohe a preto vytvoril voľný-šev (zobrazený ružovou farbou) ako priamku medzi bodmi začiatku a konca voľno-švových oblastí.



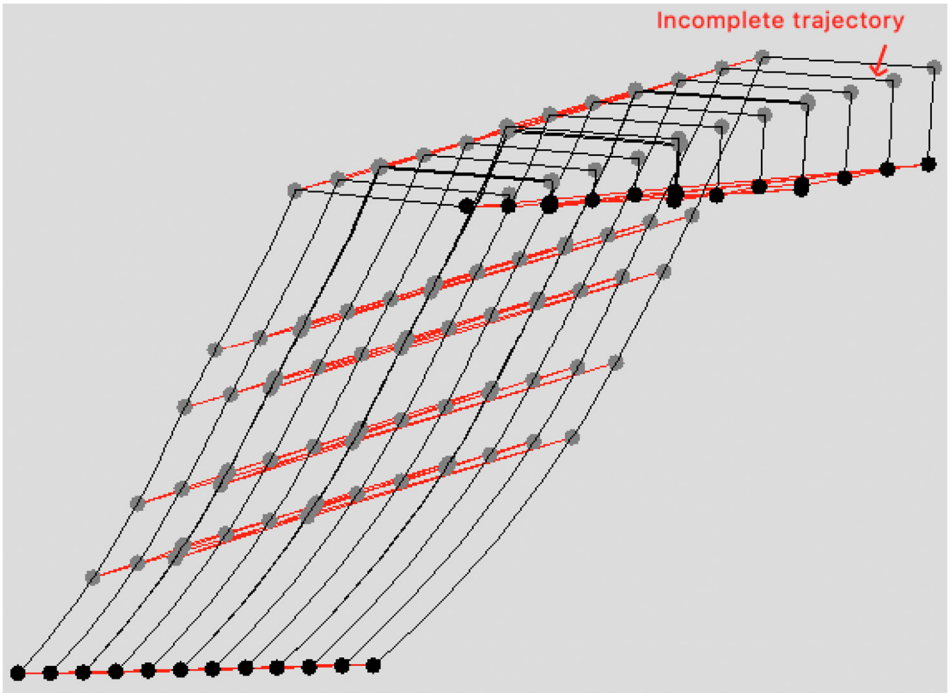
Obr. č.42 – zobrazenie rekonštrukcie po implementácií doplnenia odchýlkových bodov

Grafické zhodnotenie po implementácii doplnenia odchýlkových bodov pre konštrukčné pravidlo voľný šev môžeme vidieť na obrázku 43. Vidíme, že ¾ dát sa nám podarilo správne zrekonštruovať, pretože sme súčasnému softvérovému modulu doplnili odchýlkové body vo všetkých stupňovaných veľkostiach. ¼ dát sme po implementovaní modulu na doplnenie bodov nedokázali vyriešiť.



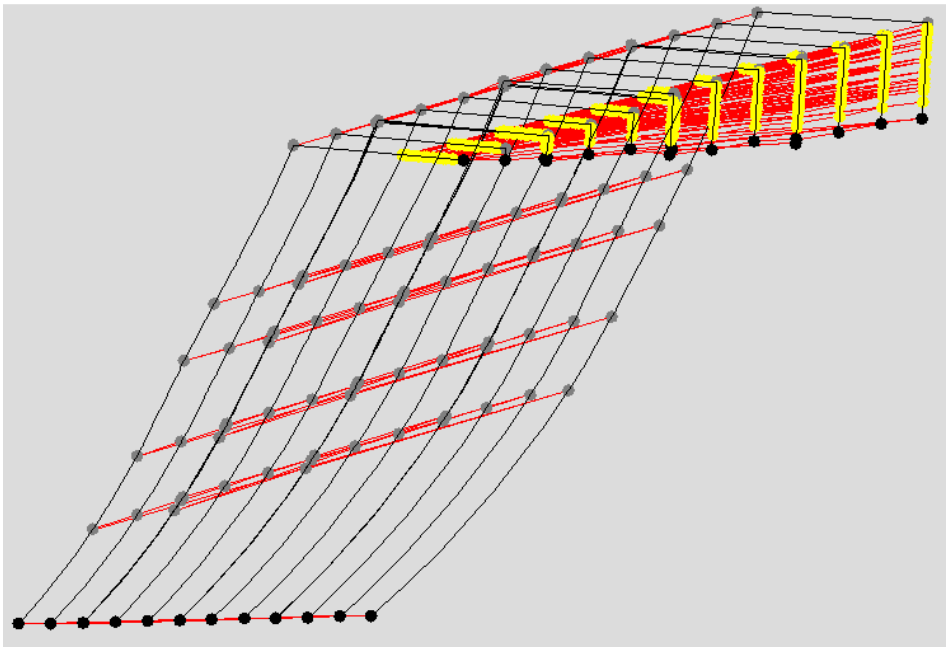
Obr. č.43 – analýza po doplnení odchýlkových bodov

Tieto problémové prípady spôsobila strata zlomového bodu v niektorej zo stupňovaných veľkostí čo zapríčinilo nekompletné trajektórie, (pozri obrázok č.44)



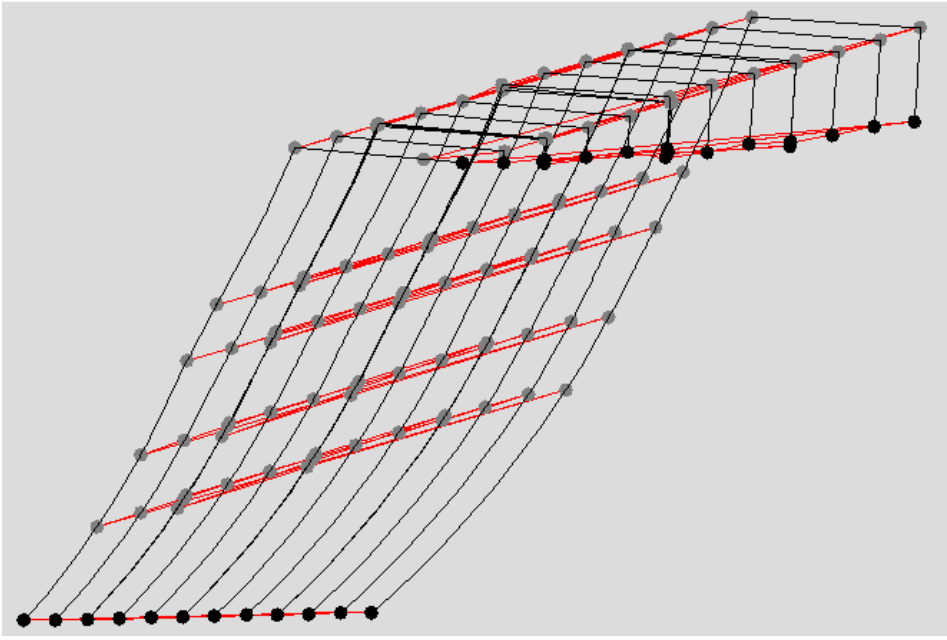
Obr. č.44 – zobrazenie nekompletnej trajektórie

Na obrázku 45 vidíme novovytvorené trajektórie pre každý zlomový bod z obrázku 44, ktorý neobsahoval žiadnu trajektóriu.



Obr. č.45 – doplnenie trajektórií pre každý zlomový bod

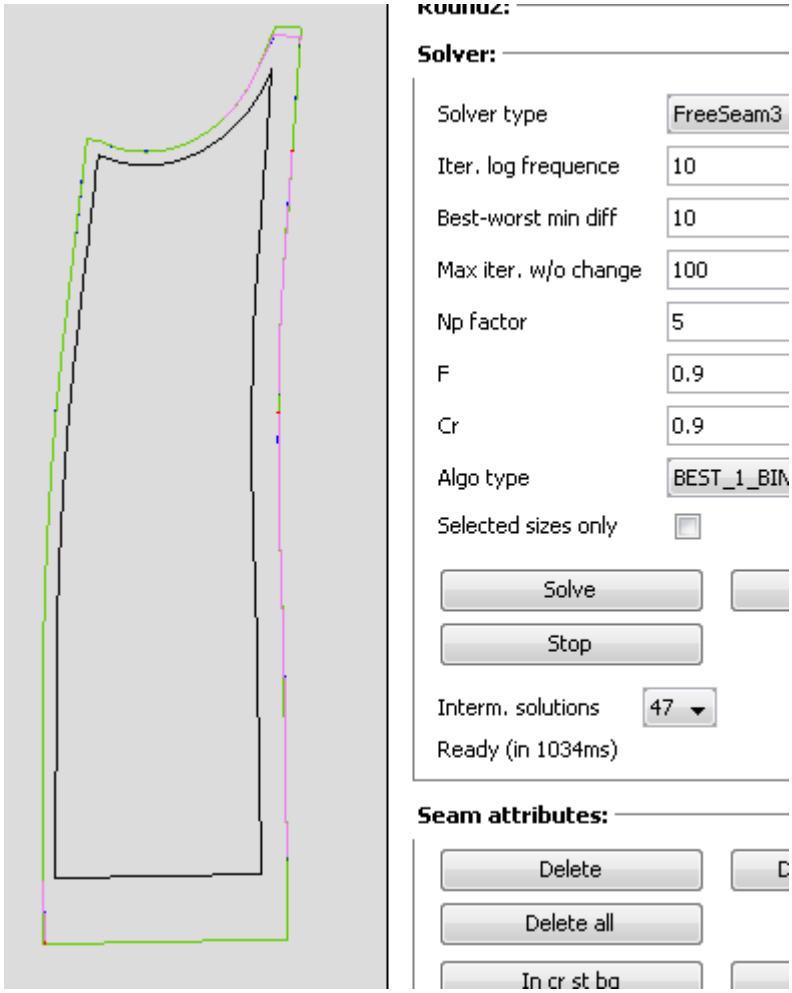
Na obrázku 46 vidíme, že po procese hľadania kandidátov a spájania novovytvorených trajektórií sme správne doplnili kompletnú trajektóriu zlomovým bodom, ktoré predtým neobsahovali žiadnu trajektóriu.



Obr. č.46 – doplnená trajektória

Na obrázku 47 vidíme, že po implementovaní modulov pre konštrukčné pravidlo voľný šev sme úspešne dosiahli tvar geometrickej predlohy. Na potrebných miestach sa

vytvoril voľný šev (ružovou farbou), ktorý bol pred implementovaním našich modulov odignorovaný.

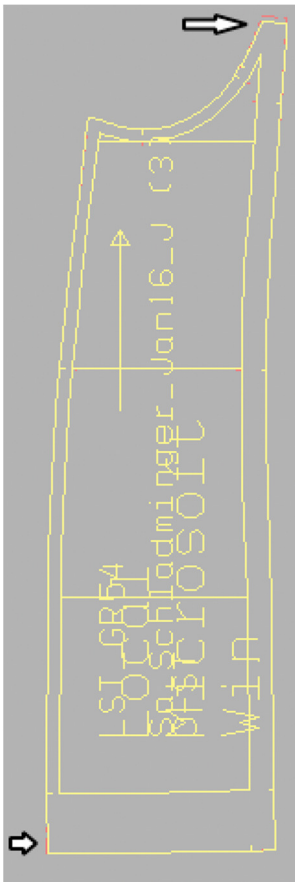


Obr. č.47 – zobrazuje výsledok procesu doplnenia trajektórií

Na obrázku 48 vidíme grafické zhodnotenie po implementovaní softvérového modulu na doplnenie trajektórií. Vidíme, že väčšinu prípadov pre konštrukčné pravidlo voľný šev sa nám podarilo úspešne zrekonštruovať. Nesprávne zrekonštruované dáta obsahovali nesúvislé elementy alebo nekorektné stupňovanie už pred samotnou rekonštrukciou.

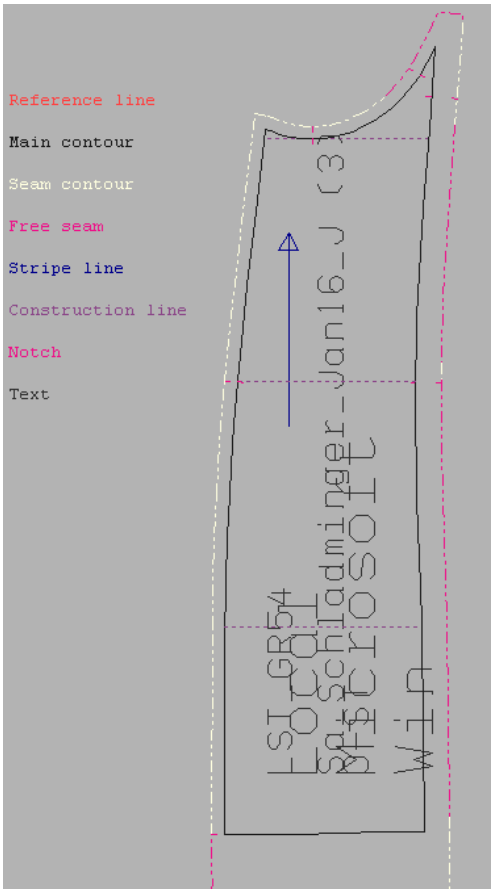
Obr. č.48 – analýza po doplnení trajektórií

Na obrázku 49 vidíme výsledok rekonštrukcie závislého šva v aplikácii cad.assyst pred implementovaním softvérových modulov. Miesta ktoré vychádzajú z tolerančnej plochy sú označené bielymi šípkami.



Obr. č.49 – zobrazenie rekonštrukcie závislého šva v aplikácií cad.assyst

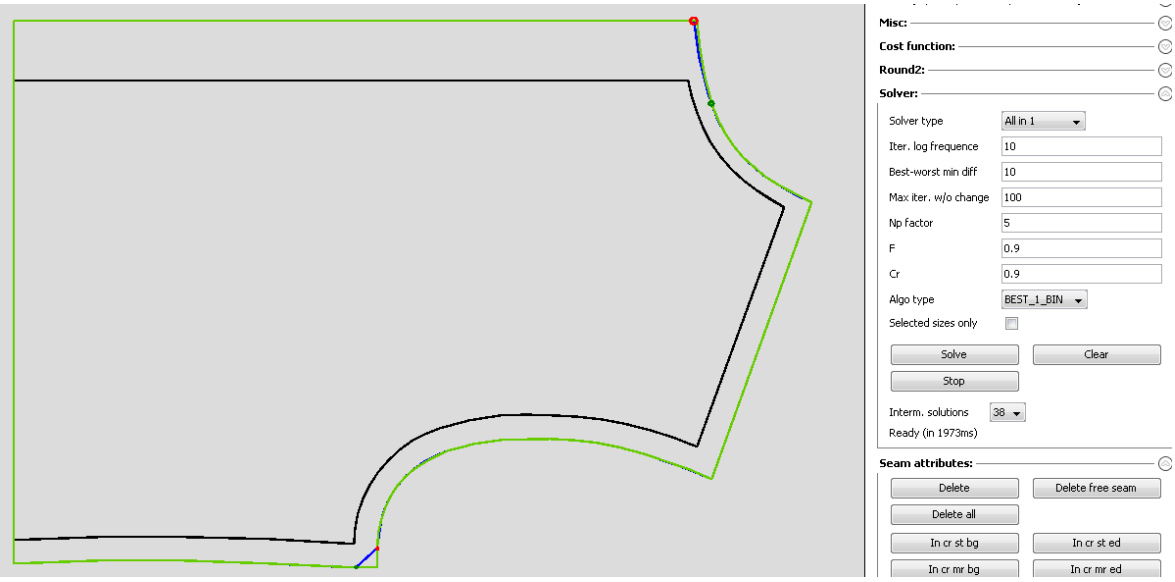
Na obrázku 50 vidíme úspešný výsledok rekonštrukcie závislého šva v aplikácii cad.assyst po implementácií našich softvérových modulov.



Obr. č.50 – zobrazenie rekonštrukcie závislého šva v aplikácií cad.assyst

4.2. Testovanie konštrukčného pravidla zastrihnutý roh

Na obrázku 51 vidíme, že pred implementovaním softvérového modulu pre zastrihnutý roh sme sa nedokázali priblížiť ku geometrickej predlohe (zobrazená modrou farbou). Proces rekonštrukcie skončil s nedokončenou verziou závislého šva. V aplikácií cad.assyst by sa nám zobrazil takýto diel žltou farbou ako chybový.



Obr. č.51 – rekonštrukcia závislého šva pred implementáciou zastrihnutého rohu

Grafické zhodnotenie pred implementovaním konštrukčného pravidla zastrihnutý roh môžeme vidieť na obrázku 52, vychádza z množiny 35 konštruovaných dielov. Vidíme, že $^1/_5$ dát nebola správne zrekonštruovaná, pretože súčasný softvérový modul geometricky nesprávne prednastavoval hodnoty zastrihnutému rohu.

Obr. č.52 – analýza pred implementovaním prednastavovania geometrických parametrov

Na obrázku 53 vidíme, že po implementovaní modulov pre konštrukčné pravidlá: voľný šev a zastrihnutý roh sme úspešne dosiahli tvar geometrickej predlohy. Na miesto,

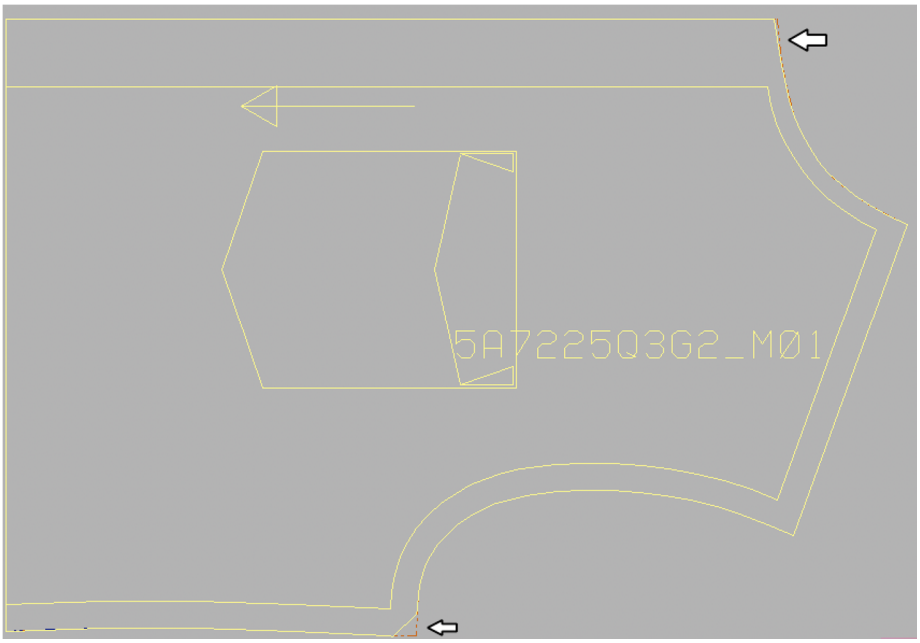
ktoré zobrazuje biela šípka sme prednastavili zastrihnutý roh „Cut“ s parametrami pre vzdialenosť a uhol k rohovému elementu na hlavnej kontúre.

Obr. č.53 – rekonštrukcia závislého šva po implementácii zastrihnutého rohu a voľného šva

Na obrázku 54 vidíme, že sa nám pomocou správneho prednastavovania geometrických hodnôt podarilo dosiahnuť priaznivé výsledky. Ostávajúce prípady, ktoré sa správne nezrekonštruovali obsahovali problém s nesprávnym stupňovaním hlavnej kontúry.

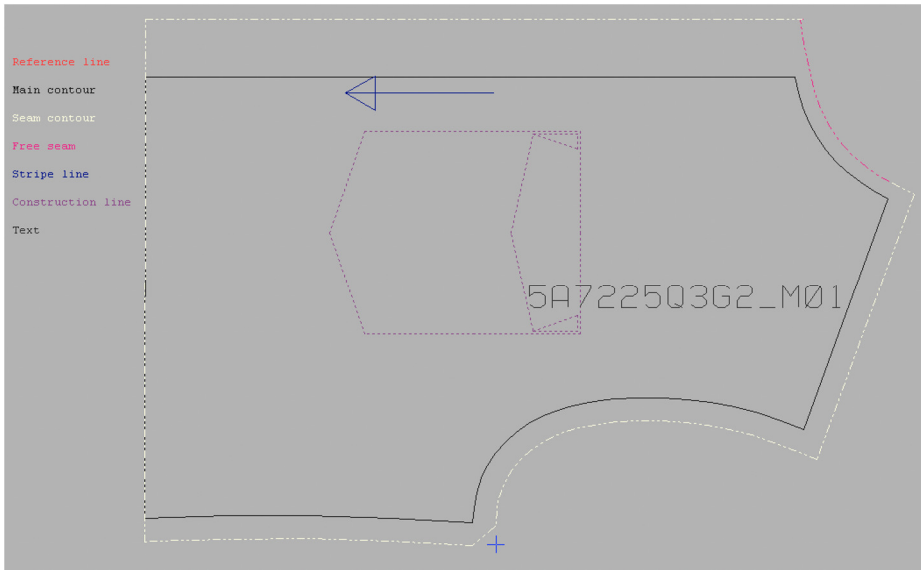
Obr. č.54 – analýza po implementovaní prednastavovania geometrických parametrov

Na obrázku 55 vidíme výsledok rekonštrukcie závislého šva v aplikácii cad.assyst pred implementovaním softvérových modulov. Miesta ktoré vychádzajú z tolerančnej plochy sú označené bielymi šípkami.



Obr. č.55 – zobrazenie rekonštrukcie závislého šva v aplikácií cad.assyst

Na obrázku 56 vidíme úspešný výsledok rekonštrukcie závislého šva v aplikácii cad.assyst po implementácií našich softvérových modulov. Miesto použitia konštrukčného pravidla zastrihnutý roh je zobrazené modrým kurzorom.



Obr. č.56 – zobrazenie rekonštrukcie závislého šva v aplikácií cad.assyst

4.3. Možnosti rozšírenia automatickej rekonštrukcie

Na obrázku 19 vidíme, že možností na rozšírenie je ešte mnoho. Najväčšie problémové skupiny s voľným švom a zastrihnutým rohom sa nám podarilo úspešne navrhnúť, implementovať a vyriešiť. V ostatných konštrukčných pravidlách sa vo väčšine prípadov vyskytuje problém s rozkúskovanými elementami. Keďže tieto elementy sú nesúvislé musíme opakovane pohybovať pracovnú množinu závislého šva na jednotlivých elementoch. Takýto posun úzko súvisí s časom rekonštrukcie, pretože musíme aplikovať konštrukčné pravidlá na každý z elementov osobitne a častokrát nedosiahneme správny výsledok.

Riešením takéhoto problému by bolo ešte pred samotnou rekonštrukciou vyhľadať na hlavnej kontúre nesúvislé elementy a v prípade existencie ich spojiť do jedného elementu. Týmto spôsobom by sme zabránili nesprávnym výsledkom a odľahčili časovú náročnosť rekonštrukcie závislého šva.

4.4. Hodnotenie softvérového modulu

Na obrázku 57 vidíme naskenovanú kópiu slovného ohodnotenia, v ktorej sa ku naimplementovaným softvérovým modulom vyjadril externý školiteľ a vedúca testovacieho tímu. Dôveryhodnosť tohto dokumentu dotyčné osoby vyjadrili vlastnoručným podpisom.

Bewertung Bachelor Thesis

František Tomana

Firma Assyst ist einer der Top-5-Softwarehersteller auf dem Markt der Bekleidungsindustrie weltweit. Um diesen Trend auch weiter erhalten zu können, sind kontinuierliche Software Verbesserungen und Innovationen unabdingbar.

Herrn Tomana wurde der Stand der Entwicklung im Bereich der Datenkonvertierung vorgestellt, mit Fokus auf das Modul der automatischen Semantics-Rekonstruktion. Herr Tomana hat von Anfang an großes Interesse gezeigt, sich in das Thema eingearbeitet und den Kunden-Workflow verstanden. Er hat Strategie entwickelt und vorgestellt, um möglichst viele existierende Lücken in dem Softwaremodul mit relativ wenig Aufwand zu schließen. Hierzu hat er seine theoretischen Kenntnisse zielgenau vertieft, hat sich mit dem Qualitätssicherungsteam mögliche Testmethoden angeschaut, analysiert und festgehalten. Seiner Strategie nach hat er sorgfältig die primäre Kategorisierung der Testergebnisse durchgeführt, die dann als Basis für die nachstehenden Verbesserungen fungierte. Diese wurden später in Absprache mit dem Entwicklungsteam implementiert, erneut getestet und entsprechend dem Bedarf nachjustiert. Das Softwaremodul hat dadurch eine entscheidende Verbesserung in der Zuverlässigkeit der Funktionalität erhalten.

Von dieser Verbesserung profitiert auch die Software cad.assyst, denn die automatische Semantics-Rekonstruktion ist Teil dieser Applikation. Die positive Rückmeldung des Qualitätssicherungsteams mit einer Abnahme des optimierten Moduls bestätigt die Qualität der Implementierung in höchstem Maß. Ich gehe ganz davon aus, dass die Zufriedenheit der Kunden dies ebenfalls zeigen wird.

In Dornach, den 27. Mai 2020

Ulrike Reng
Qualitätssicherung

Ing. Karol Žitňanský
Entwicklung

Obr. č.57 – slovné ohodnotenie po nemecky

Hodnotenie bakalárskej práce

František Tomana

Firma assyst je jedným z 5-tich najlepších výrobcov softvéru v odevnom priemysle na celom svete. Na pokračovanie v tomto trende sú nevyhnutné neustále zlepšenia a inovácie softvéru.

Pán Tomana bol oboznámený so stavom vývoja v oblasti konverzie dát so zameraním na modul automatickej rekonštrukcie sémantiky. Od začiatku prejavil pán Tomana veľký záujem, oboznámil sa s danou témou a porozumel pracovným postupom zákazníkov. Vyvinul a predstavil stratégiu na odstránenie čo najväčšieho množstva existujúcich problémov v softvérovom module s relatívne malým úsilím. Za týmto účelom cielene prehlboval svoje teoretické znalosti, skúmal možné skúšobné metódy v spolupráci s tímom zaistenia kvality, analyzoval ich a zaznamenával. Podľa svojej stratégie starostlivo vykonal primárnu kategorizáciu výsledkov testov, ktorá potom slúžila ako základ pre zlepšenia uvedené nižšie. Tieto boli neskôr implementované po konzultácii s vývojovým tímom, znovu testované a podľa potreby upravené. Výsledkom je, že softvérový modul dostal rozhodujúce zlepšenie v spoľahlivosti a funkčnosti.

Softvér cas.assyst tiež ťaží z toho zlepšenia, pretože automatická rekonštrukcia sémantiky je súčasťou tejto aplikácie. Pozitívna spätná väzba od tímu zabezpečovania kvality s prijatím optimalizovaného modulu potvrdzuje kvalitu implementácie do najvyššej miery. Plne predpokladám, že to preukáže aj spokojnosť zákazníka.

V Dornachu, 27. Mája 2020

Ulrike Reng
Zabezpečenie kvality

Ing. Karol Žitňanský
Vývoj

Obr. č.58 – slovné ohodnotenie po slovensky

Na obrázku 58 vidíme slovenský preklad firemného ohodnotenia.

Záver

Výsledkom našej práce je rozšírený softvérový modul firmy assyst, ktorý sme po úvodnej analýze dát rozdelili do dvoch problémových skupín. Najväčšiu problémovú skupinu tvorí konštrukčné pravidlo voľný šev, ktorého riešenie sme rozdelili do dvoch častí. Nasledujúcu skupinu tvorí konštrukčné pravidlo zastrihnutý roh. Jednotlivé moduly sme implementovali v programovacom jazyku java a spolu obsahujú približne 1 350 riadkov kódu.

Pôvodný softvérový modul nedokázal zachytiť úseky, ktoré mali obsahovať voľný šev, z dôvodu nevytvorenia odchýlkových bodov vo všetkých stupňovaných veľkostiach. Navrhli sme modul a následne sme ho naimplementovali, ktorým sme rozkopírovali odchýlkové body zo všetkých veľkostí a tým sme zabránili, aby boli problémové miesta v stupňovaných veľkostiach vynechané. Z toho vyplýva, že ak v niektorej zo stupňovaných veľkostí pracovná verzia závislého šva vychádzala z tolerančnej oblasti, tak sme aplikovali na túto oblasť voľný šev vo všetkých stupňovaných veľkostiach.

Po implementácii procesu dopĺňania odchýlkových bodov sme zistili, že softvérový modul nedokáže spracovať jednotlivé úseky voľno-švovej kontúry, z dôvodu stratenia zlomového bodu v niektorej zo stupňovaných veľkostí, teda stratenie zlomového bodu spôsobilo nekompletnú trajektóriu týchto bodov. Preto sme navrhli modul a následne ho naimplementovali, ktorým sme pre zlomové body, ktoré neobsahovali trajektórie vytvorili nové trajektórie. Následne sme z týchto novovytvorených trajektórií vybrali vhodného kandidáta a spojili sme ich. Týmto postupom sme vytvorili novú trajektóriu zlomovým bodom. Po vytvorení kompletnej trajektórie nemusí 'spline-line-solver' správne aproximovať cez zlomové body elementov pomocou aplikovania krivkových bodov, z toho vyplýva že dokáže vytvoriť voľný šev ekvivalentný k pôvodnej geometrickej predlohe, ktorá predstavovala nezávislú švovú kontúru.

Pôvodný softvérový modul nedokázal vytvoriť konštrukčné pravidlo zastrihnutý roh na potrebných rohových elementoch hlavnej kontúry, z dôvodu nesprávneho prednastavovania geometrických parametrov. Preto sme navrhli modul, ktorý dokáže vyhodnotiť, že dané odchýlkové body patria rohovému elementu hlavnej kontúry. Následne

sme pomocou vektorov vypočítali a prednastavili geometricky správne parametre zastrihnutému rohu, aby mohol globálny optimalizátor pomocou hodnotiacej funkcie dokončiť výpočet a dosiahnuť tvar, ktorý je v tolerancii s danou geometrickou predlohou.

Hlavný prínos našej práce spočíva v tom, že poskytujeme zákazníkom plne funkčný modul, pomocou ktorého dokážu zrekonštruovať väčšinu dát bez pomoci ručného kreslenia v CAD aplikácií. Najväčšie problémové skupiny sa podarilo s využitím vhodných heuristík úspešne implementovať do súčasného softvérového modulu firmy assyst. Implementované moduly boli riadne otestované a doladené podľa spätnej väzby testerov. Práca je plne pripravená a bola spustená do prevádzky firemnej aplikácie cad.assyst.

Literatúra

[1] Výmena dat medzi ruznými systémy CAD, Ing. Lubomír Čevela, číslo publikácie 01, rok 2001. Dostupné na internete: https://automa.cz/cz/casopis-clanky/vymena-dat-mezi-ruznymi-systemy-cad-2001_01_33435_2321/ , [*Citované dňa: 26.10.2019*]

[2] Standard practice for sewn products pattern data interchange-data format(withdrawn 2019), ASTM international, standard number ASTM D6673-10. Dostupné na internete: https://www.astm.org/Standards/D6673.htm , [*Citované dňa: 09.11.2019*]

[3] Advanced pattern making techniques: pattern grading, Ruth ann reyes-loiacano, 2020 Isn'that Sew, LLC. Dostupné na internete: http://isntthatsew.org/pattern-grading/, [*Citované dňa: 09.11.2019*]

[4] Minkowského suma, Radek Výrut, Katedra matematiky, Fakulta aplikovaných ved, Západočeská univerzita v Plzni. Dostupné na internete: http://home.zcu.cz/~rvyrut/WWW-KMA/publications/gcg2006.pdf , [*Citované dňa: 25.01.2020*]

[5] Accuracy and quality control, BBC the world's leading public service broadcaster , číslo publikácie 10, článok Textile-based materials, 2020 BBC. Dostupné na internete: https://www.bbc.co.uk/bitesize/guides/zjc3rwx/revision/10 , [*Citované dňa: 25.01.2020*]

[6] Differential Evolution, KennethV. Price, Rainer M. Storm, Jouni A. Lampien, Springer Berlin Heidelber New York, Springer 2005

[7] Computer aided design and manufacturing, K. Lalit Narayan, K. Mallikarjuna, Rao, M.M.M. Sarcar, Prentice Hall of India, New Delhi-110001, 2008

[8] Odevnícke názvoslovie, Christoph J. Jurga, P. Hamžík, Š.Galusek, Slovenské vydavateľstvo technickej literatúry v Bratislave, 1963

[9] UML 2.5 Das umfassende Handbuch, Christoph Kecher, Alexander Salvanos, Ralf Hoffmann-Elbern, Rheinwerk Computing, 2017

[10] Bekleidung Schnittkonstruktion fur Damenmode, Guido Hofenbitzer, Europa-Lehrmittel, 24. August 2009

[11] Cad.assyst 7, Assyst Bullmer inteligent solutions, assyst GmbH, 2000