UNIVERZITA KARLOVA

Přírodovědecká fakulta

Katedra aplikované geoinformatiky a kartografie

Studijní program: Geografie

Studijní obor: Geografie a kartografie

David Šklíba

3D DATOVÁ REPREZENTACE A VIZUALIZACE BLOKŮ BUDOV VZNIKLÝCH GENERALIZAČNÍ METODOU AGREGACE

3D DATA REPRESENTATION AND VISUALIZATION OF BUILDING BLOCKS GENERATED BY THE GENERALIZATION AGGREGATION METHOD

Bakalářská práce

Vedoucí bakalářské práce: RNDr. Lukáš Brůha, Ph.D.

Praha, 2021

Vysoká škola: Univerzita Karlova v Praze **Fakulta:** Přírodovědecká

Katedra: Aplikované geoinformatiky a kartografie **Školní rok:** 2020/2021

Zadání bakalářské práce

pro Davida Šklíbu

obor Geografie a kartografie

Název tématu:

3D datová reprezentace a vizualizace bloků budov vzniklých generalizační metodou agregace

Práce provede rešerši metod 3D generalizace budov a bloků budov se zaměřením na metodu agregace. Cílem praktické části je automatizovat vizualizaci výsledku agregace bloků LOD2 budov, která byla provedena pomocí matematické optimalizace. Výstupem této adaptované metody je informace o tom, které budovy se budou shlukovat do jednoho agregátu a jakou budou mít výšku. Cílem této práce bude odstranit geometrii, která se díky agregaci stala redundantní, a to při zachování původního objemu budov. Výsledný blok budov bude zarovnaný a budovy či jejich agregáty v něm budou spojitě navazovat.

Zásady pro vypracování

Rozsah grafických prací: dle potřeby

Rozsah průvodní zprávy: cca 40–60 stran

Seznam odborné literatury:

ABDUL-RAHMAN, A., PILOUK, M. (2008): Spatial Data Modelling for 3D GIS. Springer-Verlag Berlin Heidelberg, 289 pages, 978-3-540-74166-4.

BILJECKI, F., LEDOUX, H., STOTER, J. (2016): An improved LOD specification for 3D building models. Computers, Environment and Urban Systems, 59, s. 25–37.

GE, L. (2018): Generalization of LOD2 buildings with different roof structures. Journal of Spatial Science, 4, s. 19.

GUERCKE, R., GÖTZELMANN, T., BRENNER, C., SESTER, M. (2011): Aggregation of LOD1 building models as an optimalization problem. IPRS Journal of Photogrammetry and Remote Sensing, vol. 66.

KADA, M., (2011): Aggregation of 3D buildings using a hybrid data approach. Cartography and Geographic Information Science, 38 (2), s. 153–160.

Vedoucí bakalářské práce: RNDr. Lukáš Brůha, Ph.D.

Konzultant bakalářské práce: -

Datum zadání bakalářské práce: 1. 12. 2020

Termín odevzdání bakalářské práce: 30. 7. 2021

Platnost tohoto zadání je po dobu jednoho akademického roku.

…… .....……

Vedoucí bakalářské práce Vedoucí katedry

V Praze dne: 1. 12. 2020

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci zpracoval samostatně a že jsem uvedl všechny použité informační zdroje a literaturu. Tato práce, ani její podstatná část nebyla předložena k získání jiného nebo stejného akademického titulu.

Jsem si vědom toho, že případné použití výsledků, získaných v této práci mimo Univerzitu Karlovu v Praze, je možné pouze po písemném souhlasu této univerzity.

Svoluji k zapůjčení této práce pro studijní účely a souhlasím s tím, aby byla řádně vedena v evidenci vypůjčovatelů.

V Praze dne 30. 7. 2021 .....

David Šklíba

Poděkování:

Na tomto místě bych chtěl poděkovat především vedoucímu bakalářské práce RNDr. Lukáši Brůhovi, Ph.D. za cenné rady, připomínky a čas věnovaný konzultacím. Dále děkuji rodičům za podporu v průběhu celého studia.

3D datová reprezentace a vizualizace bloků budov vzniklých generalizační metodou agregace

Abstrakt

Bakalářská práce se věnuje tématu 3D datové reprezentace a vizualizace. Jejím hlavním cílem je automatizovat vizualizaci výsledku agregace bloků LOD2 budov, která byla provedena pomocí matematické optimalizace.

V první části je provedena rešerše metod 3D generalizace budov a bloků budov se zaměřením na metodu agregace včetně optimalizační úlohy, z níž pocházejí vstupní data této práce. Dále je podán výklad vlastního návrhu algoritmu. Nejprve jsou rozebrány jednotlivé fáze metody, následně je představena vlastní implementace v jazyce *Python 3*.

Princip vlastního algoritmu spočívá v odstranění redundantních dat pomocí vizualizace agregace budov se stejným typem střechy. Minimalizaci velikosti výsledného souboru dále obstará odstranění mezer mezi budovami a zarovnání stěn budov na vnější straně bloku. Veškeré změny geometrie probíhají za podmínky zachování velikostí objemů těl a střešních částí budov.

Klíčová slova: 3D GIS, generalizace, agregace, automatizace generalizace

3D data representation and visualization of building blocks generated by the generalization aggregation method

Abstract

The bachelor thesis deals with the topic of 3D data representation and visualization. Its main objective is to automate the visualization of the result of the aggregation of LOD2 building blocks, which was performed using optimization method.

In the first part, a survey of 3D building and building block generalization methods is performed, focusing on the aggregation method including the optimization problem from which the input data of this thesis comes. Next, an explanation of the actual algorithm design is given. The different phases of the method are discussed, followed by the actual implementation in *Python 3*.

The principle of the actual algorithm is to remove redundant data by visualizing an aggregation of buildings with the same roof type. The minimization of the size of the resulting file is further provided by removing gaps between buildings and aligning the walls of the buildings on the outside of the block. All geometry changes are done under the condition that the volumes of building bodies and roof parts are preserved.

Keywords: 3D GIS, generalization, aggregation, generalization automation

Obsah

Seznam obrázků

1. Úvod

Od začátku 90. let minulého století zaznamenáváme hromadný nárůst aktivit usilujících o automatizaci generalizace. Nejprve byl kladen důraz na individuální operace, jako jsou zjednodušení, výběr nebo agregace, jejich vzájemná souhra se začala řešit až později (Müller, Zeshen 1992 cit. dle Sester 2020, s. 6). V dnešní době pozorujeme přesun zkoumání generalizace od tradiční kartografie k jejímu řešení v 3D prostoru. Významu tedy generalizace nabývá na jednotlivých úrovních detailu (anglicky level of detail, dále zkráceně LOD). Již pro mnoho operací existuje řada uspokojivých algoritmů, a tak dochází pouze k okrajovým vylepšením (Sester 2020). Preciznost provedení generalizace se uplatňovala zejména v tištěných mapových dílech. S technologickým rozvojem webových aplikací má běžný uživatel možnost si digitální mapu prohlédnout ve více měřítcích. Víceúrovňové mapové dílo nabízí možnost zobrazení většího počtu informací.

V 3D mapovém provedení se se změnou měřítka může měnit i LOD. V poslední době navíc neustále přibývá aplikací využívajících 3D modely budov. Slouží k vizualizačním záměrům spojeným s navigačními systémy nebo k různým simulacím, jako jsou šíření hluku, dopady katastrof či environmentální simulace (Guercke a kol. 2011). Na příkladu malých obrazovek navigačních zařízení je velmi důležitým aspektem kartografická generalizace, po jejímž dobrém provedení dokáže uživatel rychle a bezpečně interpretovat vzniklou situaci a učinit tak správné rozhodnutí.

2. Cíle práce

Hlavním cílem této práce je automatizovat vizualizaci provedené generalizace metodou agregace. Navržený postup odstraní geometrická data, která se stala po provedení generalizace nadbytečná. Vlastní algoritmus bude minimalizovat redundanci dat za podmínky nepozměněných objemů střechy a těla budov ve srovnání se stavem před vstupem do programu. Dílčím cílem je, aby se ve výsledném modelu zpracovávaného bloku budov nevyskytovaly žádné mezery mezi odvozenými agregáty budov. Zároveň v případě budov uspořádaných do obdélníkového bloku bude vnější obrys půdorysu celého bloku tvořit pravoúhlý čtyřúhelník, tedy žádná budova nebude vyčnívat směrem do ulice.

Práce navazuje na diplomovou práci Měchurová (2020), jejímž hlavním cílem bylo navrhnout a implementovat postup globálně optimální agregace 3D modelů budov na úrovni LOD2. Získané informace o optimálním provedení agregace jsou v práci i vizualizovány. Vizualizaci však chybí automatizace zpracování modelů budov v podobě simplifikace obrysu vzniklých bloků. Ve skriptu visualization.py práce Měchurová (2020) autorka na jednotlivé budovy aplikovala nově vypočtené výšky těla budovy a střešní části. Nová metoda v této práci na skript plynule navazuje.

3. Kartografická generalizace

V této části budou rozebrány teoretické principy týkající se hlavního tématu práce, a to kartografické generalizace. Pořadí jednotlivých kapitol odpovídá tomu, jak se daná témata postupně objevovala v odborné literatuře. Kapitola *3.1* rozebírá tradiční 2D kartografickou generalizaci. Kapitola *3.2* na ni poté navazuje generalizací v 3D prostoru. Její povahu v oblasti budov determinuje úroveň detailu rozebraná v podkapitole *3.2.1.* Vzhledem k tomu, že právě budovy jsou hlavním cílem zájmu této bakalářské práce, odvíjí se od toho názvy ostatních podkapitol. Zmínky z odborné literatury na toto téma jsou prezentovány od 3D generalizace samotných budov v podkapitole *3.2.2* přes agregaci budov do bloků v podkapitole *3.2.3* až po agregaci střech v podkapitole *3.2.4*. Kapitola *3.3* se věnuje problematice optimalizační úlohy, z níž pocházejí vstupní data této práce. Nejprve podkapitola *3.3.1* uvádí řešení optimalizační úlohy pro LOD1. Podkapitola *3.3.2.* následně představuje upravený průběh tohoto optimalizačního problému cílený na LOD2. Poslední kapitola *3.4* prezentuje způsoby výpočtů objemů budov s různými typy střech.

3.1 2D generalizace

Kartografická generalizace je metoda sloužící ke znázornění a zdůraznění určité informace podléhající zejména účelu mapy (Lysák 2018). V podstatě se jedná o výběr prvků, které budou a které zároveň nebudou zobrazeny. Větším zmenšováním reality se kladou větší nároky na provedení generalizace. V primární fázi generalizace, která probíhá již při mapování v terénu, se stanovením minimální mapovací jednotky určí, které objekty nebudou mapovány. Dále při vlastní kartografické generalizaci se využívá metod zjednodušování tvarů, seskupení objektů (agregace), či upravování klasifikace mapových znaků (úprava znakového klíče) (Sester 2020). V závěrečné fázi se nakonec provede generalizace s přihlédnutím k zaplnění mapy (náplň mapy), tj. řeší se nedostatky, které nebyly při generalizaci jednotlivých prvků samostatně patrné. Mimo účel a měřítko mapy kartografickou generalizaci podmiňuje charakter zobrazovaného území, kterým se rozumí výskyt, význam, počet, poloha a rozměr objektů.

Metody kartografické generalizace se rozlišují na grafickou (geometrickou) a konceptuální metodu. První způsob řeší grafické parametry kartografických znaků, jejichž

pozměnění se může zhostit sám kartograf, zatímco u druhého způsobu musí asistovat odborník na danou problematiku, neboť řeší vlastnosti znázorňovaných jevů (Lysák 2018).

3.2 3D generalizace

Po vzoru 2D generalizace řeší 3D generalizace přehlednost a čitelnost. Nově se zde objevuje požadavek na datovou velikost modelu. S tím je spojeno více úrovní detailu, na jejichž bázi model vzniká. Vzhledem ke způsobu vzniku vstupních dat (letecká fotogrammetrie, laserové skenování) jde při generalizaci o zdůraznění specifických kartografických vlastností objektů namísto od fotorealistického zobrazení. Problematika stále disponuje otevřenými otázkami, a to nalezením definice generalizačních požadavků a pravidel pro 3D objekty, implementací těchto pravidel a vytvořením mechanismů pro optimální výběr vhodného LOD pro danou analýzu (Sester 2020).

3.2.1 Úroveň detailu (LOD)

Pojem LOD byl poprvé použit v počítačové grafice jako nástroj zlepšení vizualizace. V CityGML (anglicky City geography markup language), který patří mezi standardizované datové modely pro ukládání 3D modelů měst a krajiny, je definováno pět úrovní detailu (Ge 2018). Nejméně detailní úrovní detailu je LOD0, v němž se budovy zobrazují jako jejich půdorysy nebo polygony vymezené krajními hranami střech. V LOD1 jsou již budovy reprezentovány pomocí hranolů ovšem pouze s plochými střechami. V přechodu do LOD2 dochází k rozšíření o více typů střech. Půdorysy budov už nemusí tvořit pravoúhlé čtyřúhelníky. V LOD2 se odehrává celá část této bakalářské práce. LOD3 se vyznačuje detailními částmi zdí a střech. LOD4 do modelu přidává interiéry budov.

3.2.2 3D generalizace budov

Většina algoritmů vzniklých v posledních dvou dekádách cílí na generalizaci samotných budov. Kada (2007) použil dvoustupňovou metodu. V první části je ve 2D promítnut půdorys budovy, který je následně disjunktně rozdělen na čtyřúhelníkové polygony. Ve druhé části dojde ke generalizaci střechy, a to vždy pro každý čtyřúhelníkový polygon, vzniklý metodou buněčného

rozložení, zvlášť. Upravením parametrů jednotlivých částí střechy dojde ke spojení geometrie a vytvoření generalizovaného objektu.

3.2.3 Agregace budov do bloků

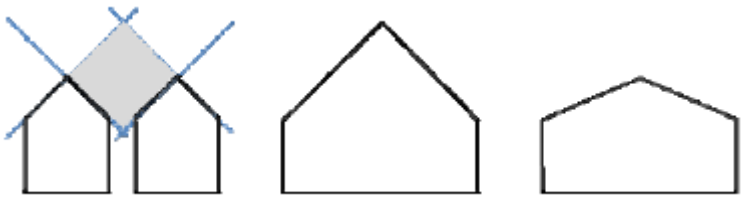
Damen, Kreveld, Spaan (2008) ve svém přístupu využívají morfologických operátorů. Daří se jim eliminovat mezery mezi budovami a agregovat je tak do jedné. Problematiku však řeší pouze nad půdorysy budov.

Kada 2011 rozšiřuje svoji myšlenku z roku 2007 o hybridní datový přístup, který umožňuje agregaci budov do bloků. Práce nadále využívá metodu cell decomposition, morfologické operace jsou ale tentokrát prováděny nad rastrovou reprezentací. Jako hlavní kritérium pro agregaci autor rozebírá faktor vzdálenosti, jehož způsob uchopení je pro sloučení budov do bloku stěžejní. Též je prezentována multifunkčnost algoritmu pro různé případy sousedství domů. Přístup využívá hraničních hodnot parametrů, tedy určuje situaci, kdy má dojít k agregaci.

Xie J a kol. (2012) přišel s hybridní metodou pro generalizaci a vizualizaci urbánních dat. Budovy jsou agregovány do bloků v závislosti na vzdálenosti pozorovatele. Algoritmus této hybridní metody funguje podobně jako v Damen, Kreveld, Spaan (2008) nad půdorysy budov.

3.2.4 Agregace střech

Na problematiku generalizace střech se ve svém přístupu zaměřuje Kada (2011). Naráží však na problém neexistujících kartografických zákonitostí v záležitosti agregace sousedních střech, svírajících mezi sebou tvar údolí. Tato práce využívá jeden z prezentovaných způsobů řešení, viz obr. č. 1 (Figure 11., Kada 2011):



Obr. č. 1: Způsob řešení agregace budov vytvářejících mezi střechami tvar údolí

Na omezené množství algoritmů pro generalizaci různých typů střech a na mezi nimi nedokonalé řešení prostorových vztahů se snaží reagovat Ge (2018). Ten na rozdíl od předchozích přístupů uchopuje řešení problému v opačném pořadí. Nejprve definuje 3 základní typy střech: a) pultová, b) sedlová nebo valbová, c) stanová. Dále je vytvořen algoritmus pro automatické rozpoznání daného typu na základě počtu polygonů a jejich normálových vektorů. V závislosti na složitosti kombinací typů střech autor rozdělil budovy na jednoduché, kombinované a komplexní. Pozornost je věnována generalizaci jednotlivých variací kombinovaných střech. Nakonec probíhá agregace generalizovaných budov. Do této práce vstupují pouze jednoduché budovy, a proto není potřeba jakkoliv na začátku provádět jejich generalizaci.

3.3 Optimalizační úloha

V první podkapitole bude popsána optimalizační úloha zveřejněná v článku Guercke a kol. (2011). Následně na ni v druhé podkapitole naváže představení jejího vylepšení v Měchurová (2020).

3.3.1 Optimalizační úloha pro úroveň LOD1

Autoři představují pojem centrum agregátu. Tímto centrem se rozumí jedna z budov původního datasetu. Tato budova tak identifikuje daný agregát. Příslušnost budovy do agregátu je tedy vyjádřena přiřazením této budovy do jeho centra. Centrum agregátu je potom přiřazeno samo sobě. Příslušnost budovy danému centru lze vyjádřit binárně pro každou budovu v ležící v množině B (Cb1, Guercke a kol. 2011):

$$\forall v \in B: \sum_{u \in B} X_{uv} = 1$$

Pro ujištění se, že došlo ke spojení pouze sousedních budov, je navržen model toků (flow model). V prvním kroku se definuje graf sousedství, ve kterém uzly reprezentují jednotlivé budovy. Sousedící budovy se pak spojí pomocí oblouku. Pokud lze přepravit komoditu z libovolného vertexu (bod vymezující hranu) do centra agregátu bez toho, aby po cestě agregát opustil, považuje se agregát za propojený. Každá budova, která není centrem, generuje odtok

rovný jedné, centrum poté přijímá všechny přítoky a negeneruje žádný odtok. Rozdíl odtoku a přítoku v budovách, které nejsou centrem, je roven jedné, zatímco v centrech se pohybují v rozmezí od -M do 0 (Cf2, Guercke a kol. 2011):

$$\forall v \in B: \quad \sum_{a=(v,u)\in A} f_a - \sum_{a=(u,v)\in A} f_a \geq 1 - X_{vv}(M+1)$$

$$\sum_{a=(v,u)\in A} f_a - \sum_{a=(u,v)\in A} f_a \leq 1 - X_{vv}$$

Pokud na hraně existuje pozitivní tok, uzly na obou koncích hrany náleží stejnému centru (Cf3, Guercke a kol. 2011):

$$\forall (u,v) \in A, \forall c \in B: \; X_{cu} \geq X_{cv} + (F_{(u,v)} - 1)$$

,přičemž pouze jedna hrana zdroje generuje pozitivní tok.

Co se týče agregace LOD1 modelů budov, je jedním z hlavních témat určení výšky agregátu. Tato výška se určí během optimalizačního procesu za minimalizace objektivní funkce v momentě, kdy jsou uspokojeny všechny podmínky (*objective_function*, Guercke a kol. 2011):

$$min: \sum_{v\in B} X_{vv} + W \sum_{u,v\in B} \Delta V_{uv}$$

Pro zjednodušení objektivní funkce zde hlavní roli hraje změna pseudo-objemu budovy, která je přiřazena centru agregátu. Aby se minimalizoval počet agregátů, je tato objemová změna násobena parametrem W, její hodnota tak bude menší než 1. Výška agregátu odpovídá hodnotě, při které dochází k nejmenší změně součtu pseudo-objemů budov v agregátu. Neznámé hodnoty objemové změny a výšky výsledného agregátu získáváme z definice (CΔV, Guercke a kol. 2011):

$$\forall v \in B: \; \Delta V_{uv} \geq (Bh(v) - H_u)A(v) - (1 - X_{uv})MB_{vol.v}$$

$$\Delta V_{uv} \geq -(Bh(v) - H_u)A(v) - (1 - X_{uv})MB_{vol.v}$$

Kde Hu značí výšku výsledného agregátu s centrem u, $h(v)$ výšku původní budovy, ΔVuv objemové změny a $Mvol$ libovolné číslo větší, než je největší možná objemová změna, která může pro danou budovu *v* nastat.

Mimo omezení zajišťující validnost topologie výsledného modelu by měla být nastavena hranice maximálního možného rozdílu výšky budovy v agregátu a jeho centra. Případně je možné stanovit maximální možnou změnu objemu budovy agregátu.

3.3.2 Rozšíření optimalizačního problému na úroveň LOD2

Hlavní změna přichází v přidání informace o typu střechy. Zatímco LOD1 uvažuje pouze ortogonální tělesa různých tvarů, budova v LOD2 již není ortogonální a rozlišuje typ střechy. Autorka uvádí, že „hlavní myšlenkou rozšíření optimalizačního problému je přidání omezujících podmínek, týkajících se jednotlivých parametrů, které se vážou na střechy jako je: typ střechy, objem střechy, orientace, výška“ (Měchurová 2020, s. 38). Algoritmus umožňuje i agregaci budov s odlišným typem střechy, záleží na hodnotě vstupního parametru ε, který nabývá celých čísel v intervalu $< 0{,}5 >$. Při hodnotě $\varepsilon = 5$ v optimalizační úloze nezáleží na typu střechy, naopak nastavením hodnoty $\varepsilon = 0$ bude docházet jen k agregaci budov se stejným typem střech. Definice omezení typu střech potom vypadá následovně:

$$X_{uv}\left(Rt(v) - Rt(u)\right) \leq \varepsilon \qquad \forall x \in \mathrm{B}: Rt(u), Rt(v) > 0$$

$$X_{uv}\left(Rt(v) - Rt(u)\right) \geq -\varepsilon \qquad \forall x \in \mathrm{B}: Rt(u), Rt(v) > 0$$

Z kapitoly *3.3.1* již známe proměnnou X_{uv}, nově však přibyla proměnná Rt, která pomocí celého čísla z intervalu <1,5> představuje typ střechy. Hodnoty odpovídají těmto typům:

$$\begin{array}{l} \textit{hodnota R} \\ \textit{typ střechy} \end{array} : \begin{array}{ccccccccc} 1 & & 2 & & 3 & & 4 & & 5 \\ \textit{stanová} & \rightarrow & \textit{rovná} & \rightarrow & \textit{pultová} & \rightarrow & \textit{sedlová} & \rightarrow & \textit{valbová} \end{array}$$

Dále rozdíl orientací střech musí být menší než předem stanovený úhel ε_{Ro}.

$$X_{uv}\left(Ro(v) - Ro(u)\right) \leq \varepsilon_{Ro} \qquad \forall x \in \mathrm{B}: Ro(u), Ro(v) > 0$$

$$X_{uv}\left(Ro(v) - Ro(u)\right) \geq -\varepsilon_{Ro} \qquad \forall x \in \mathrm{B}: Ro(u), Ro(v) > 0$$

V neposlední řadě rozdíl výšek budov nesmí přesahovat nastavený parametr ε_{Ro}, a to zvlášť pro výšku těla budovy a výšku střešní části. Pro oba případy se zvlášť použije podmínka:

$$X_{uv}\left(Bh(v) - Bh(u)\right) \leq \varepsilon_{Ro} \qquad \forall x \in \mathrm{B}: Bh(u), Bh(v) > 0$$

$$X_{uv}\left(Bh(v) - Bh(u)\right) \geq -\varepsilon_{Ro} \qquad \forall x \in \mathrm{B}: Bh(u), Bh(v) > 0$$

Nastavení omezujících podmínek pro objemové změny střešního objektu vychází z podmínky *CΔV* (kapitola *3.3.1*):

$$\forall v \in B:\ \Delta VR_{uv} \geq (Rh(v) - HR_u)K(v) - (1 - X_{uv})MR_{vol.v}$$

$$\Delta VR_{uv} \geq -(Rh(v) - HR_u)K(v) - (1 - X_{uv})MR_{vol.v}$$

Kde K je konstanta vypočítaná pro každou budovu ze vztahu:

$$K(v) = \frac{V_R(v)}{Rh(v)}$$

V optimalizační úloze figuruje objektivní funkce, která se primárně snaží minimalizovat počet agregátů v daném modelu. Vstupují do ní 2 parametry WB, WR a nacházející se v oboru reálných čísel na intervalu (0,1). Určují, jaký vliv budou mít objemové změny těla (WB) a střechy budovy (WR) na výsledek optimalizační úlohy. Čím více se blíží hodnota jedné, tím větší vliv parametr má. Objektivní funkce je potom definována jako:

$$min: \sum_{v \in B} X_{vv} + WB \sum_{u.v \in B} \Delta V_{uv} + WR \sum_{u.v \in B} \Delta VR_{uv}$$

3.4 Výpočty objemů

Vzhledem k tomu, že se v Měchurová (2020) objevovaly výpočty objemů budov, byly využity následující funkce:

body_volume (wallcoords, footprintcoords)

vypočítá výšku těla budovy z listu souřadnic stěn kolmých na půdorys. Objem těla budovy potom není nic jiného než součin délek všech hran kvádru:

$$V = a * b * c$$

roof_volume (footprintcoords, rooftype, roofcoords)

vypočítá výšku pouze střešní části a podle typu střechy spočítá její objem.

Stanová (*pyramidal*) jako objem jehlanu:

$$V = \frac{1}{3}(S * v)$$

Šikmá (*shed*) a sedlová (*gabled*) jako polovina kvádru výšky střechy:

$$V = \frac{1}{2}(a * b * c)$$

Valbová (*hipped*) jako objem klínu:

$$V = \frac{1}{6} * b * v(2a + a1)$$

4. Vstupní data

Vzhledem k návaznosti na Měchurová (2020) pocházejí vstupní data právě z jejích skriptů. Ty jsou celkem tři (buildingparameters.py, optimization_solver.py a visualization.py). Po řadě na sebe navazují, jednotlivé výstupy tedy slouží jako vstupy dalším skriptům. Konkrétně výstup skriptu buildingparameters.py vstupuje dále do optimalizační úlohy, tato práce z něj jako první vstup využívá informace o typech střech a jejich orientacích vztažených k jednotlivým budovám. Druhým vstupem této bakalářské práce je výstupní soubor skriptu optimization_solver.py, který pomocí centrální matice poskytuje informaci o objektech, které se mají agregovat. Posledním vstupem je výstupní OBJ soubor skriptu visualization.py. Tento soubor zobrazuje 3D model budov s aplikovanými parametry vypočtenými optimalizační úlohou.

4.1 Omezení vstupních dat

Typ střech musí být jako pátá položka souboru ve formátu JSON. Informace musí být uložena ve slovníku, kde je klíčem identifikátor objektu (dále jen obj_id) a hodnotou číselné označení typu střechy. Orientace střech, je podobně jako typ, uložena jako osmá položka stejného souboru. Vyjadřuje ji úhel mezi horní hranou střechy a osou x, viz Měchurová (2020).

Centrální matice nesoucí informaci o agregaci konkrétních budov musí být uložena ve formátu JSON. V něm se vyskytuje jako slovník. Každý klíč začíná textovým řetězcem „center_matrix_“ a dále pokračuje identifikátory dvou objektů. Hodnoty jsou pak vyjádřeny binárně, „1“ vyjadřuje náležitost budovy k centru agregace, případně že se jedná o tentýž objekt; „0“ agregaci zamítá.

Pro správné přečtení souboru ve formátu OBJ je třeba dodržet následující strukturu: hlavička, vrcholy (každý musí být definován na samostatném řádku), plochy tvořené indexy vrcholů (každá opět na samostatném řádku). Pro každou budovu jsou pak tyto plochy odděleny názvem (obj_id) budovy.

5. Generalizace bloku budov metodou agregace

V této kapitole bude představeno vlastní provedení generalizace bloku budov metodou agregace. V příloze č. 1 je znázorněno grafické schéma metody. Nejdříve budou v podkapitole *5.1* rozebrány kritéria vstupního modelu bloku budov. Poté bude následovat vysvětlení jednotlivých částí algoritmu, a to příprava dat k agregaci v podkapitole *5.2*, samotná agregace v podkapitole *5.3*, zarovnání samostatných budov do bloku v podkapitole *5.4* a nakonec kontrola mezi sousedními agregáty v podkapitole *5.5*.

5.1 Kritéria vstupního modelu bloku budov

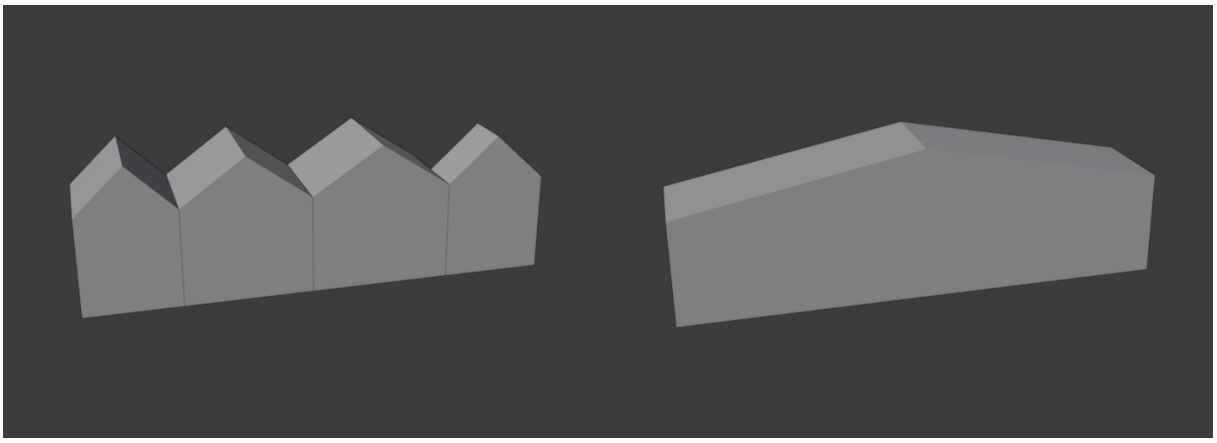
Jednotlivé půdorysy musí mezi stranami svírat pravý úhel, těla budov potom představují kvádr, či krychli. Stěny sousedních budov musí být rovnoběžné. V rámci agregátu se musí vyskytovat budovy se stejným typem střechy a se stejnou hodnotou její orientace. Program je omezen na pět typů střech, které se skrývají pod těmito čísly: 1 – stanová, 2 – rovná, 3 – pultová, 4 – sedlová, 5 – valbová. V případě, že pro agregát existují vůči souřadnicové ose x 2 hodnoty orientace, musí tyto budovy obsahovat rovný typ střechy. Budovy musí být uspořádány do pravoúhlého bloku připomínající obdélník či písmena L a U, případně do linie. Pro jiný tvar bloku algoritmus nevrátí validní výsledek.

5.2 Příprava dat k agregaci

Nejsnáze lze k většině výpočtů přistupovat za předpokladu rovnoběžnosti hran budov se souřadnicovými osami. Proto je na začátku zjištěna orientace celého bloku vůči ose x. V případě hodnoty orientace různé od 0° je třeba převést souřadnice do vlastního souřadnicového systému, kde bude daná rovnoběžnost platit. V dalším kroku přichází na řadu zarovnání vnějších stěn bloku. Na konci této fáze tvoří vnější strany půdorysu bloku pravoúhlý čtyřúhelník, viz obr. č. 7.

5.3 Agregace

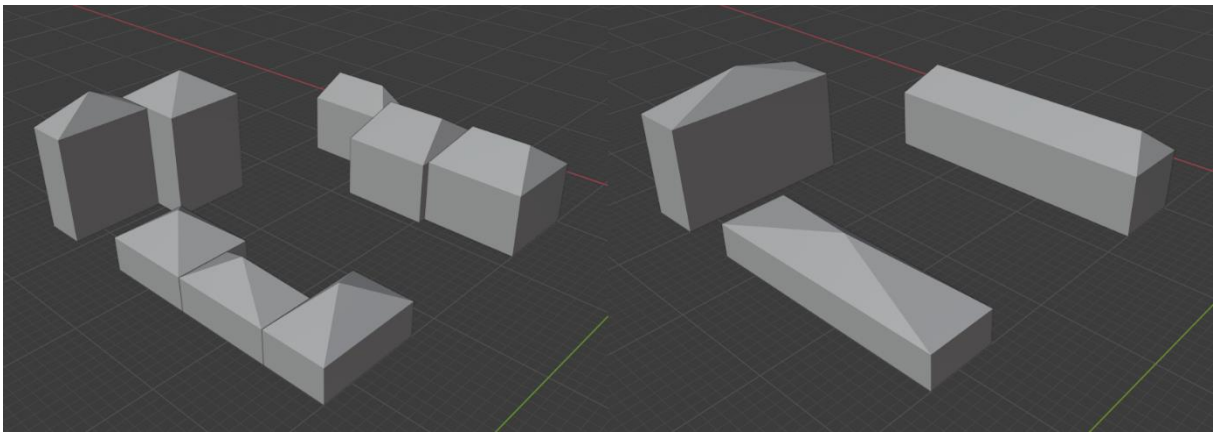
Z výsledku optimalizační úlohy známe n-tice budov, u kterých má dojít k agregaci. Vlastní program v jedné iteraci provede agregaci právě jedné n-tice. V případě, že agregát tvoří roh bloku budov, je podle orientace vůči souřadnicovým osám rozdělen na dvě části. Pro každou část je poté agregace provedena zvlášť. Hlavními podmínkami pro provedení agregace jsou shodný typ střech a hodnota orientace. Při stejných hodnotách orientace střech však spolu sousední budovy mohou tvořit tvar údolí, i v tomto případě agregace proběhne, viz obr. č. 1.



Obr. č. 2: Agregace bloku budov, jejichž střechy tvoří mezi budovami tvar údolí

Algoritmus ve skriptu této práce po celou dobu ponechává hranu společné stěny stejně dlouhou z důvodu předejití kolizi se sousední budovou. Primárně se tedy prodlouží šířka (stěna kolmá na společnou stěnu). V případě potenciální kolize agregátu (po zvětšení jeho šířky) s jinou budovou, kdy by mělo dojít k porušení topologie (budovy by se překrývaly), dojde též k modifikaci výšky. Maximální povolené překrytí sousední budovy nově vzniklým agregátem je nastaveno na 3/10 její původní délky. Pokud je totiž původní délka překryté budovy velmi krátká a došlo by k jejímu výraznému zmenšení, nemusel by být na první pohled dobře rozpoznatelný typ střechy. Zkracováním nejvyšší hrany se valbová střecha přibližuje stanové. Jakmile známe novou délku a šířku, můžeme dopočítat výšku střešní části. Umístění nejvyšších vrcholů většiny typů střech plyne z jejich definice. Nejasnost ale nastává u valbové střechy. Algoritmus v tomto případě vytvoří vrcholy střechy ze dvou od sebe nejvzdálenějších nejvyšších vrcholů střech budov agregátu. Nakonec program zkontroluje topologickou validitu modelu. V případě kolize mezi samotnými agregáty dojde k posunutí toho agregátu, jež do svého souseda zasahuje velikostí své nové šířky, viz obr. č. 8. Ke kolizi může dojít i v rámci jednoho agregátu, a to potom, co je rozdělen na dvě části, jejichž slučování probíhá nezávisle na sobě. Průnik dvou nově vzniklých částí agregátu je od jedné z částí odečten, viz znázornění

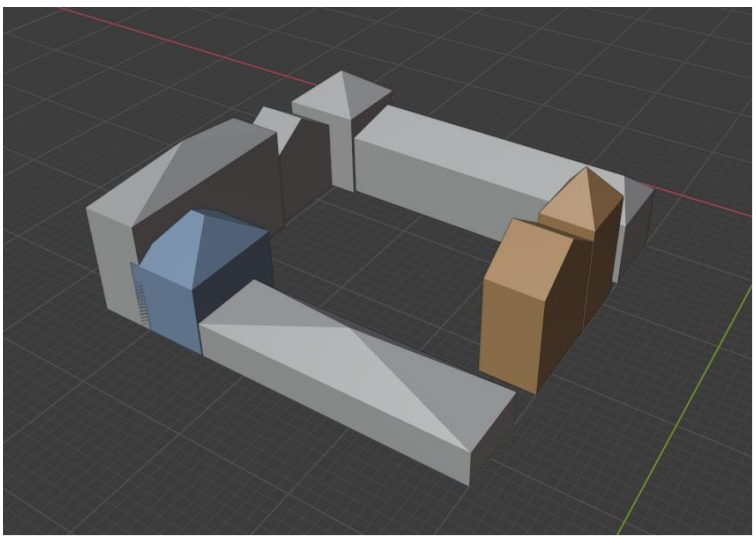
průběhu na obr. č. 9. Na obr. č. 3 lze vidět příklad finálního provedení agregace na testovacích datech.



Obr. č. 3: 3D model budov před agregací (vlevo) a po agregaci (vpravo)

5.4 Zarovnání samostatných budov do bloku

Po provedení agregace mohou nastat dva případy. Obě situace jsou názorně vidět na obr. č. 4. Buďto dojde ke zvětšení mezer mezi budovou a agregátem (béžové budovy) nebo nově vzniklý agregát překryje sousední budovu (modrá budova). V poslední fázi algoritmus tyto nedostatky odstraňuje.



Obr. č. 4: 3D model po provedení agregace

Jestliže na nějakém místě došlo k překryvu samostatné budovy, zmenší se její délka tak, aby se překryv odstranil. Se zmenšením délky se musí prodloužit šířka.

Pokud se vyskytuje mezera mezi samostatnou budovou a nově vzniklým agregátem, dojde k posunu stěny samostatné budovy až na úroveň agregátu, jehož geometrie tak zůstává nepozměněna. Následně musí dojít ke zmenšení její šířky.

Na rohovou budovu však nelze uplatnit žádný ze zmíněných postupů, neboť by budova tímto způsobem mohla překrýt některou ze sousedních budov. Proto musí dojít ke změně výšky těla budovy.

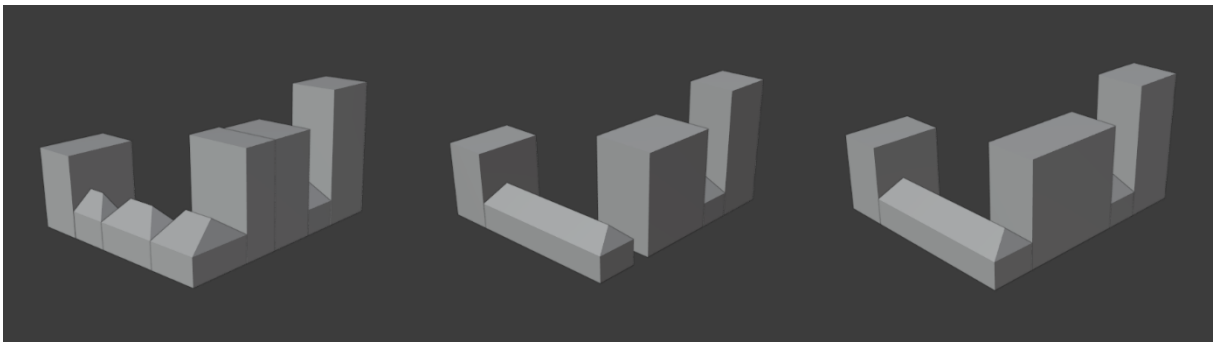
V případě, kdy se mezera nachází mezi samostatnými budovami, jsou stěny obou budov posunuty proti sobě o stejnou vzdálenost.

Pokud se vytvořením agregátů změnil tvar vnějšího obrysu půdorysu bloku budov, algoritmus dokáže umístění změny identifikovat a pro budovy se stejnou orientací vůči souřadnicovým osám rekurzivně provede jejich zarovnání na úroveň vnější stěny agregátu.

Poslední fází zůstává úprava geometrie střech. Opět, jako v případě agregátů, dochází k výpočtům nových výšek střešních částí z již známých vzorců zmíněných v kapitole *3.4*. Oproti přístupu k agregátům však nastává jedna změna. Vzhledem k algoritmem možné změně délky hrany domu s valbovou střechou rovnoběžné s nejvyšší hranou střechy je potřeba přepočítat polohu vertexů ohraničujících právě tuto nejvyšší hranu. Výstupní soubor zachovává formát OBJ, uživatel si akorát zvolí jeho název a místo úložiště.

5.5 Kontrola mezi sousedními agregáty

V případě dvou sousedících agregátů může jejich agregací dojít ke vzniku nové mezery mezi nimi. Pro kontrolu této nežádoucí situace je s agregáty zacházeno přesně tak, jak bylo zacházeno se samostatnými budovami v předchozí kapitole. Řešením je vždy prodloužení délky a zkrácení šířky jednoho z agregátů. Na obr. č. 5 si můžeme zleva prohlédnout přeměnu dat od vstupního modelu přes fázi ukončenou v kapitole *5.4* po finální podobu upravenou právě v tomto kroku.



Obr. č. 5: Vyplnění mezery mezi agregáty

6. Implementace vlastního algoritmu

Na začátku skriptu byla zavedena globální proměnná $EPSILON = 0.01$, která představuje vyšší hodnotu, než je maximální možná chyba výpočtů programu odpovídající 1 cm. Postupně bude sloužit jako parametr některých funkcí. Před postupným voláním funkcí algoritmu jsou ze vstupů zavedeny 2 proměnné v podobě slovníků budov. Slovník *roof_types* uchovává informaci o typu střechy a slovník roof_orientation obsahuje hodnoty její orientace. Každý další parametr bude vysvětlen vždy u první funkce, ve které se objeví.

get_object_orientation (object_parts)

Ze souřadnic půdorysu libovolné budovy určí vektor hrany půdorysu směřující od osy x. Za parametr object_parts je dosazen slovník budov obsahující zvlášť stěny a souřadnice vertexů jednotlivých částí budovy. Pomocí vztahu:

$$\cos\alpha = \frac{\vec{\mathbf{u}} \cdot \vec{\mathbf{v}}}{|\vec{\mathbf{u}}| \cdot |\vec{\mathbf{v}}|}$$

kde u je vektor osy x v kladném směru a v je vektor hrany půdorysu, dojde k výpočtu úhlu svíraného oběma vektory. V případě, že $úhel >= 90°$, je od něj 90° odečteno.

Pokud úhel převyšuje hodnotu EPSILON, tedy hrany budov nejsou rovnoběžné se souřadnicovými osami, nejdříve funkce:

get_point_of_rotation (objects, vertices)

pomůže zjistit vertex, jímž procházející přímka rovnoběžná s osou z bude sloužit jako osa rotace. Za parametr *objects* je třeba dosadit slovník budov obsahující stěny určené indexy vertexů, za parametr vertices potom seznam souřadnic všech vertexů. Následně se funkce:

move_and_rotate_objects (vertices, rotation_point, angle, roof_orientation)

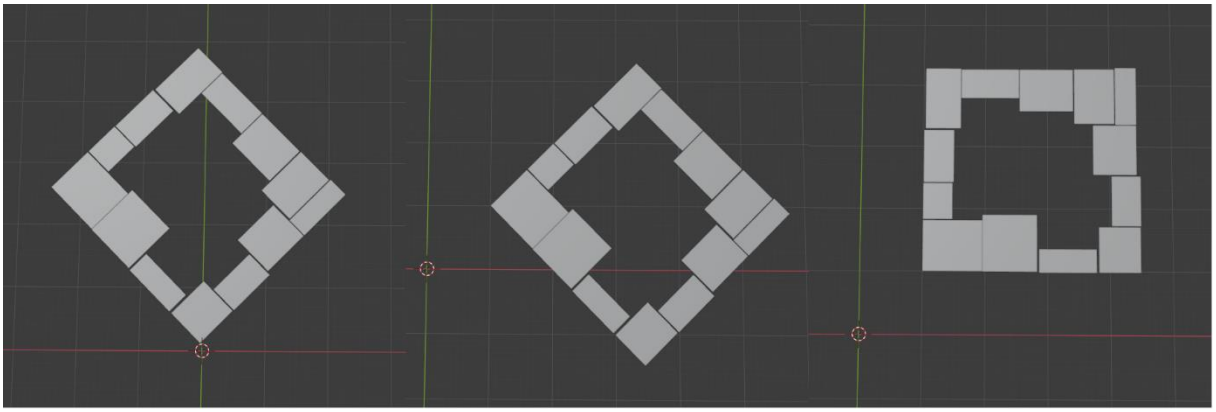
postará o posun a rotaci celého bloku. Ve směru od předtím zjištěného bodu je k souřadnici x, y = [10, 10] stanoven směrový vektor. O tento vektor jsou v následujícím kroku posunuty všechny vertexy. Rotace bloku se provede pomocí matice rotace, která vychází ze vztahu:

$$\mathbf{R}_2 = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

kde úhel α = 90° - angle. Dále je od bodu o souřadnicích x, y = [10, 10] k jednotlivým vertexům iterativně vytvářen sloupcový směrový vektor, u něhož dojde k otočení podle vztahu:

$$\mathbf{R_2} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Otočením vektoru se vertex dostane na své nové místo. Dále funkce přepočítá orientace střech jednotlivých budov. Dva neznámé parametry vznikly provedením předchozích funkcí: *rotation_point* jako výstup funkce *get_point_of_rotation ()*, *angle* jako výstup funkce *get_object_orientation ()*. Níže na obr. č. 6 si lze všimnout vlivu funkce na půdorys celého bloku. Vlevo se nachází podoba půdorysu před vstupem do funkce, prostřední případ odpovídá situaci po posunutí a ukázka vpravo zachycuje polohu půdorysu vůči souřadnicovým osám po provedení rotace.



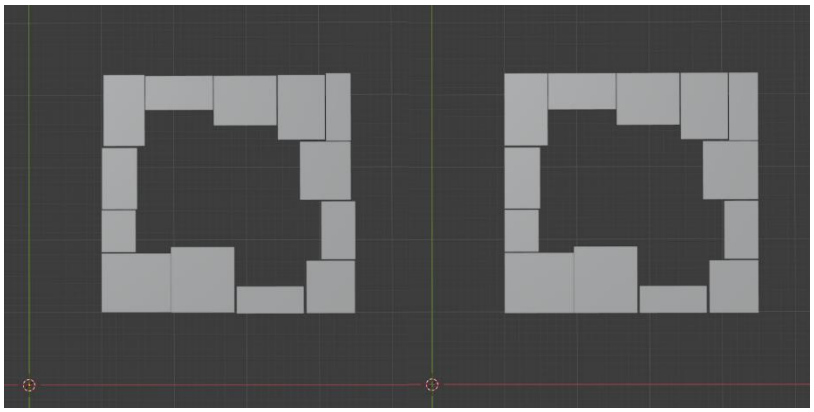
Obr. č. 6: Postavení půdorysů vůči souřadnicovým osám (osa x – červená, osa y – zelená)

get_global_extremes (vertices)

Zjistí minimální a maximální hodnoty souřadnic vertexů (*x_min_global, x_max_global, y_min_global, y_max_global*) na osách x a y. K těmto hodnotám se budou zarovnávat jednotlivé stěny. Tomuto účelu slouží funkce:

align_outer_boundary (x_min_global, x_max_global, y_min_global, y_max_global, object_parts, vertices, neighborhood, roof_types)

,která iterativně vyšetřuje, jakému ze souřadnicových extrémů na obou osách se hrana každé vnější stěny budovy nejvíce blíží. Parametr *neighborhood* je nahrazen slovníkem, jehož hodnotami jsou opět slovníky obsahující obj_id sousedící budovy a její pozici vůči hlavní budově. Obr. č. 7 zobrazuje geometrii půdorysu před vstupem do funkce (případ vlevo) a po výstupu z funkce (případ vpravo).



Obr. č. 7: Změna geometrie půdorysu zarovnáním vnějších stěn bloku

get_aggregate_parametres (aggregation_parts, counter, volume_dict, aggregation_parts_dict, objects, vertices, roof_types)

Spočítá objemy těla a střechy agregátu. Obě hodnoty jsou výsledkem součtů dílčích objemů budov agregátu před první úpravou jejich geometrie. Parametr *aggregation_parts* je seznam seznamů budov v agregátu, *counter* značí pozici agregátu v předchozím seznamu, *volume_dict* ve slovníku uchovává hodnoty objemů střech a těl budov, *aggregation_parts_dict* odpovídá slovníku *object_parts*, ale jen pro budovy, které jsou součástí agregátů.

Pro výpočet nové šířky je tedy ještě potřeba znát délku a výšku těla budovy.

calculate_aggregate_length (x_parallel, side, x_min, x_max, y_min, y_max)

Vypočítá velikost délky (*length*) z minimálních a maximálních hodnot agregátu v souřadnicovém systému. Argument *x_parallel* říká, jestli je společná stěna agregátu rovnoběžná se souřadnicovou osou x, pokud ano, délka se vypočítá jako rozdíl maximální a minimální x-ové souřadnice vertexu v agregátu. V opačném případě se provádí rozdíl souřadnic na ose y. Parametr *side* potom specifikuje pozici agregátu v rámci bloku.

calculate_new_width (wallcoords, length, volume_body, neighbor_relation, object_parts, lower_vertex, x_parallel, side)

Nejprve vypočítá výšku těla budovy z listu souřadnic stěn kolmých na půdorys (*wallcoords*). Nyní lze vypočítat novou šířku (*new_width*) ze vztahu:

$$\text{šířka} = \frac{objem}{délka * výška}$$

Kde objem těla budovy udává parametr *volume_body*. Za parametr *lower_vertex* jsou dosazeny souřadnice vertexu půdorysu na vnější stěně budovy. Pomocí sousedského vztahu

neighbor_relation funkce zjistí, zdali nedošlo k přesažení maximálního povoleného překrytí sousední budovy, pokud ano, je šířka zkrácena na povolenou mez, nakonec se aktualizuje hodnota výšky (*height*) podle upraveného vztahu:

$$výška = \frac{objem}{délka * šířka}$$

compute_new_body_coordinates (x_parallel, lower_vertex, higher_vertex, new_width, height, side)

Vypočítá souřadnice těla agregátu a vrátí je v seznamu (*points*), parametr *higher_vertex* se od parametru *lower_vertex* liší vyšší hodnotou jedné z horizontálních souřadnic v závislosti na orientaci vůči souřadnicovým osám.

compute_new_rooftop_coordinates (roof_type, roof_orientation, volume_roof, length, new_width, x_parallel, side, roof_up, points)

Vypočítá souřadnice nejvyšších bodů střechy, a to pomocí nově zjištěné výšky střešní části. Parametr *volume_roof* udává velikost objemu střechy, *roof_up* potom souřadnice nejvyšších vrcholů střech budov agregátu.

create_geometry (roof_type, points, obj_id)

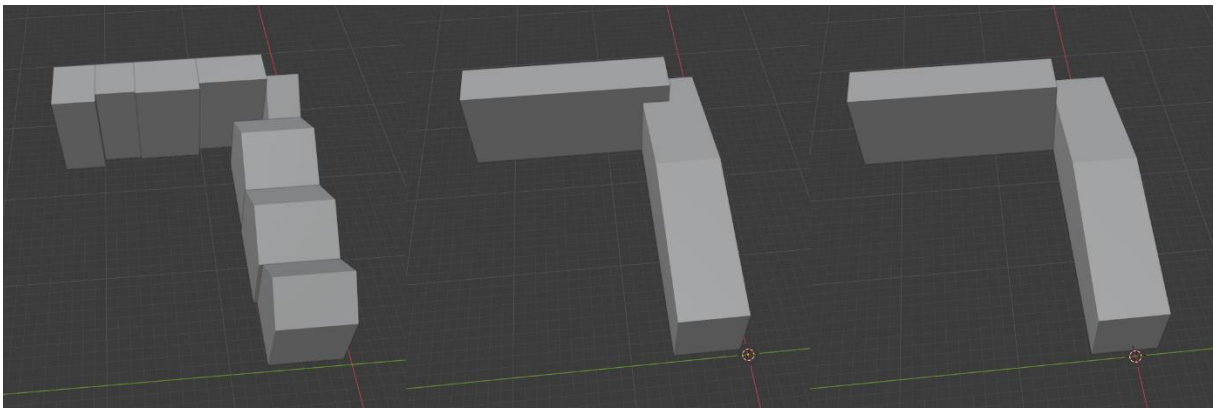
Vytvoří seznamy souřadnic vertexů budovy, indexů těla budovy a indexů střešní části.

detect_colision (object_parts, aggregation_parts)

Z nově vzniklých agregátů vyšetří, zda mezi nimi nedošlo k porušení topologie. Vrací dva seznamy dvojic obj_id agregátů, mezi nimiž došlo ke kolizi. Jeden seznam tvoří agregáty s odlišným centrem, druhý seznam potom části agregátu se společným centrem.

move_object (collision, x_parallel, side, vertices, object_parts)

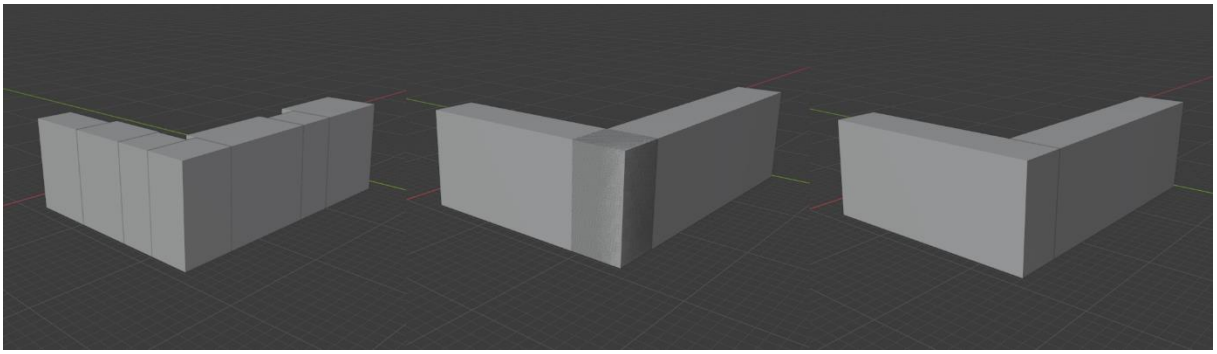
Posune jeden z dvojice agregátů s odlišným centrem, mezi nimiž došlo ke kolizi. Parametr *collision* obsahuje seznam obj_id dvou překrytých budov. obsahuje Na obr. č. 8 si lze prohlédnout směrem zleva doprava vývoj agregace od vstupního modelu přes vznik kolize až po její následné vyřešení.



Obr. č. 8: Vznik kolize dvou agregátů a její odstranění

remove_overlap (identical_object, object_parts, vertices, x_parallel, side)

Odstraní překryv agregátů se společným centrem odečtením jejich průniku od jednoho z nich. Parametr *identical_object* uchovává obj_id obou nových budov. Parametry *x_parallel* a *side* jsou použity od jedné z budov. Průběh agregace je k vidění na obr. č. 9.



Obr. č. 9: Agregace objektu tvořícího roh bloku budov

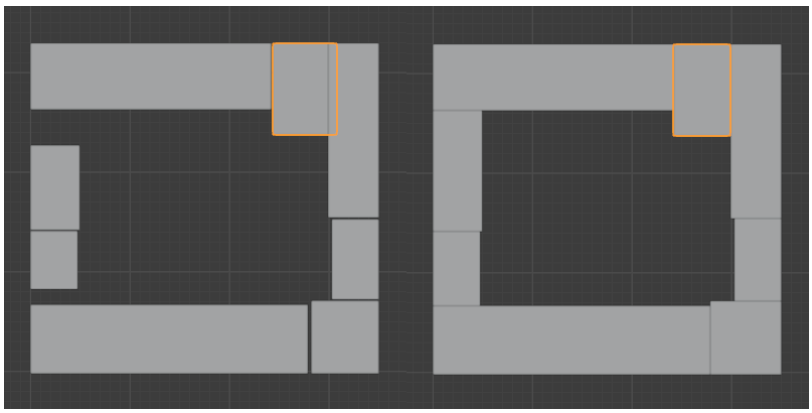
change_length_of_single_buildings (vertices, neighborhood, objects, object_parts, parts_list)

Poupraví délky samostatných budov tak, aby na sebe vnější stěny bloku plynule navazovaly. Jako parametr *part_list* se zadá seznam obj_id budov, které tvoří agregáty. S odstraněním mezer mezi budovami pomáhá vnořená funkce:

fill_gaps (obj_id, obj_id1, parts_list, new_objects, new_vertices, objects, vertices, object_parts, direction, EPSILON)

která pro samostatnou budovu vyhodnocuje změnu délky. Parametry *new_objects* a *new*_vertices odpovídají po řadě parametrům *objects* a *vertices* pro nově vytvořené agregáty. Parametr *direction* určuje směr budovy *obj_id* od budovy *obj_id1*. Na obr. č. 10 můžeme

zaznamenat vliv předchozích dvou funkcí na vstupní data. Půdorys budovy, u níž se funkce musela zbavit překryvu, je pro lepší názornost zvýrazněn oranžovým obrysem.

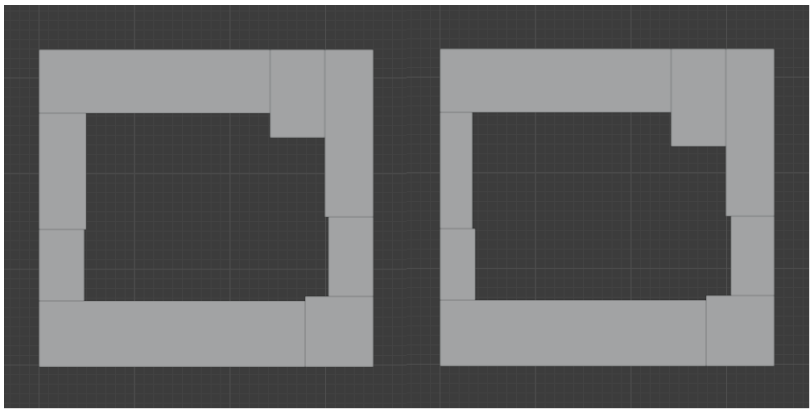


Obr. č. 10: Změna půdorysu bloku po vyplnění mezer mezi budovami

Pokud se nejedná o rohovou budovu, spustí se funkce:

modify_width_of_single_object (volume_body, length, body_height, wall_faces, x_parallel, x_min, x_max, y_min, y_max, x_min_global, x_max_global, y_min_global, y_max_global, vertices, EPSILON)

která nejprve, stejným způsobem jako při agregaci, vypočítá novou šířku. V parametru *body_height* je uložena výška těla budovy, *wall_faces* představují stěny kolmé na půdorys určené indexy vertexů, *x_min, x_max, y_min, y_max* obsahují minimální a maximální hodnoty budovy na horizontálních osách souřadnicového systému. Změna půdorysu je patrná z obr. č. 11.



Obr. č. 11: Změna půdorysu bloku po vypočítání finálních rozměrů půdorysů všech budov

ensure_about_alignment (vertices, neighborhood, obj_id, parts_list, object_parts, new_object_parts, roof_type, x_min, x_max, y_min, y_max, x_min_global, x_max_global, y_min_global, y_max_global, wall_faces, roof_faces, EPSILON, aligned_objects)

Vyšetří, zdali na sebe navazují vnější zdi agregátu a samostatné budovy. V parametru *roof_faces* jsou uloženy střešní polygony vymezené indexy vertexů. Do parametru *aligned_objects* se zadá seznam obj_id již v touto funkcí zarovnaných budov. V případě nenávaznosti vnější zdi agregátu v rámci bloku a samostatné budovy vnořená funkce:

define_alignment (footprintcoords, direction, x_min, x_max, y_min, y_max, x_min_global, x_max_global, y_min_global, y_max_global, roof_type, vertices, wall_faces, roof_faces, EPSILON)

určí směr posunutí vertexů vnější stěny a pomocí funkce:

align_object_boundary (object, vertices, change, x_min, x_max, y_min, y_max, x_min_global, x_max_global, y_min_global, y_max_global)

stěnu zarovná. Parametr *change* určuje, které ze stěn kolmých na půdorys se změní geometrie.

V případě rohové budovy dojde pomocí funkce:

modify_body_height_of_single_object (volume_body, length, width, roof_faces, corner, vertices, body_height, roof_top)

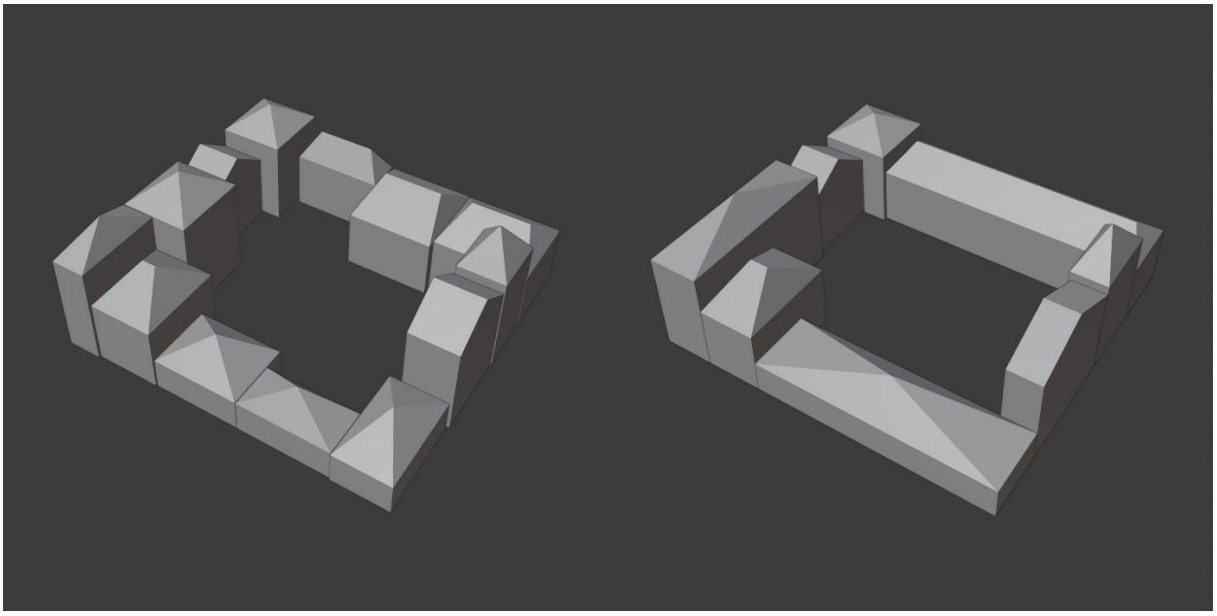
k vypočítání nové výšky těla budovy, a to poupravením předchozího vztahu pro výpočet šířky. Parametr *corner* pomocí booleovské proměnné určuje, zda se jedná o rohovou budovu.

modify_rooftop_of_single_object (roof_type, x_min, x_max, y_min, y_max, length, width, body_height, volume_roof, roof_top_vertices, vertices, x_parallel, EPSILON)

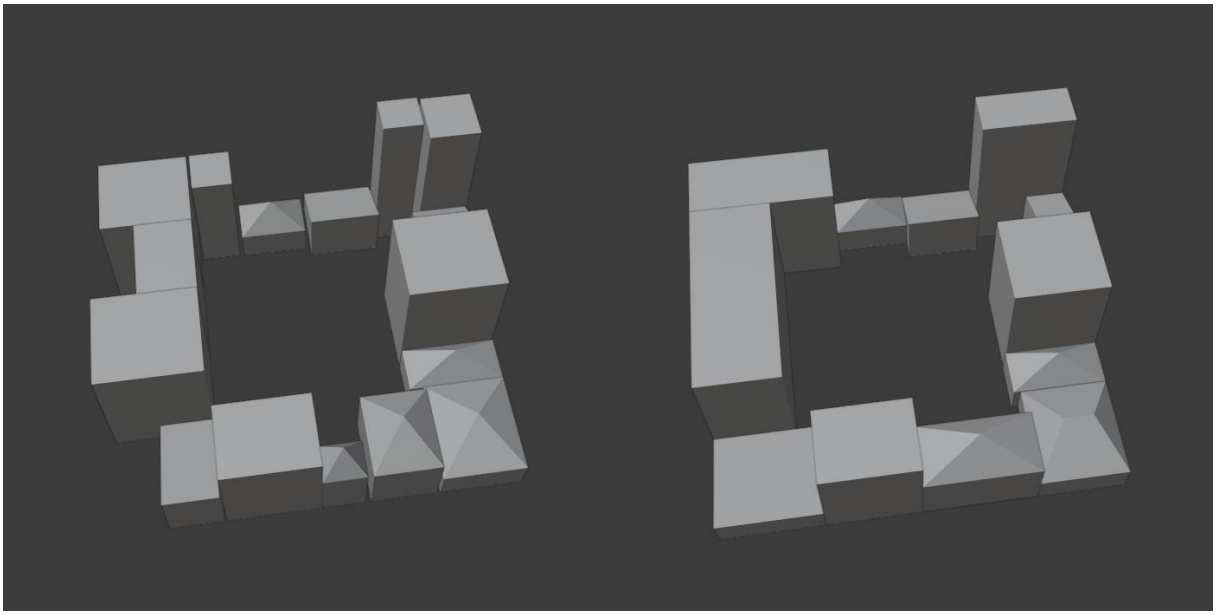
Přepočítá hodnotu souřadnic střešní části budovy, vrátí seznam vertexů.

7. Experimentální výsledky

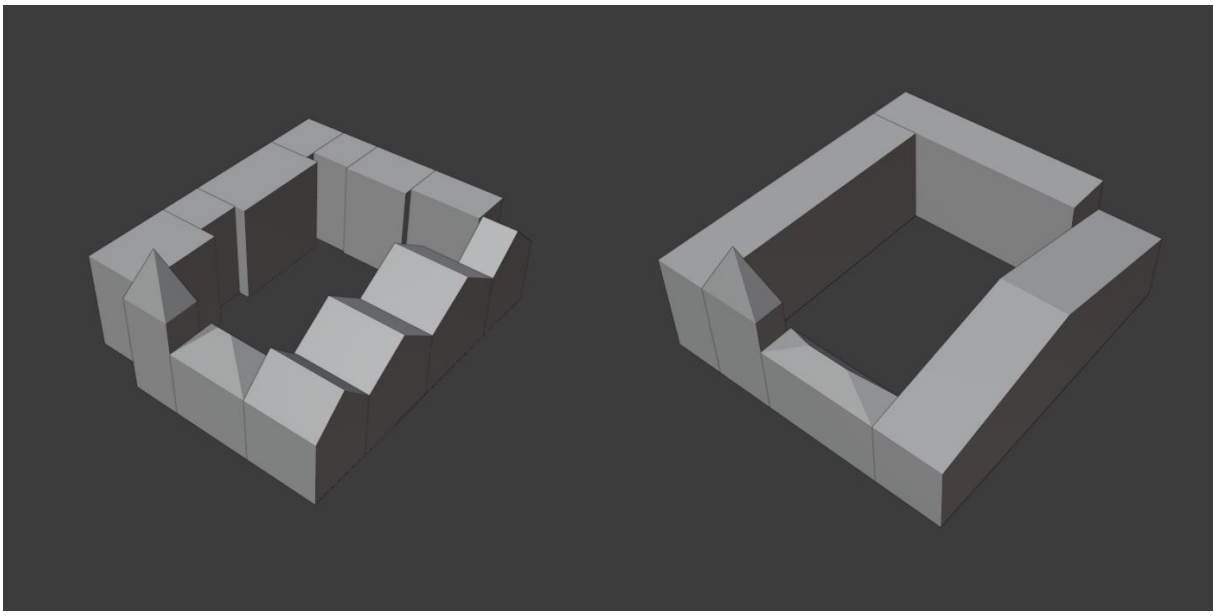
Program byl testován na všech výstupech práce (Měchurová 2020). Níže budou prezentovány spolu s výstupy této práce, a to tím způsobem, že vstup této práce bude zobrazen nalevo a k němu vztažený výsledek napravo.



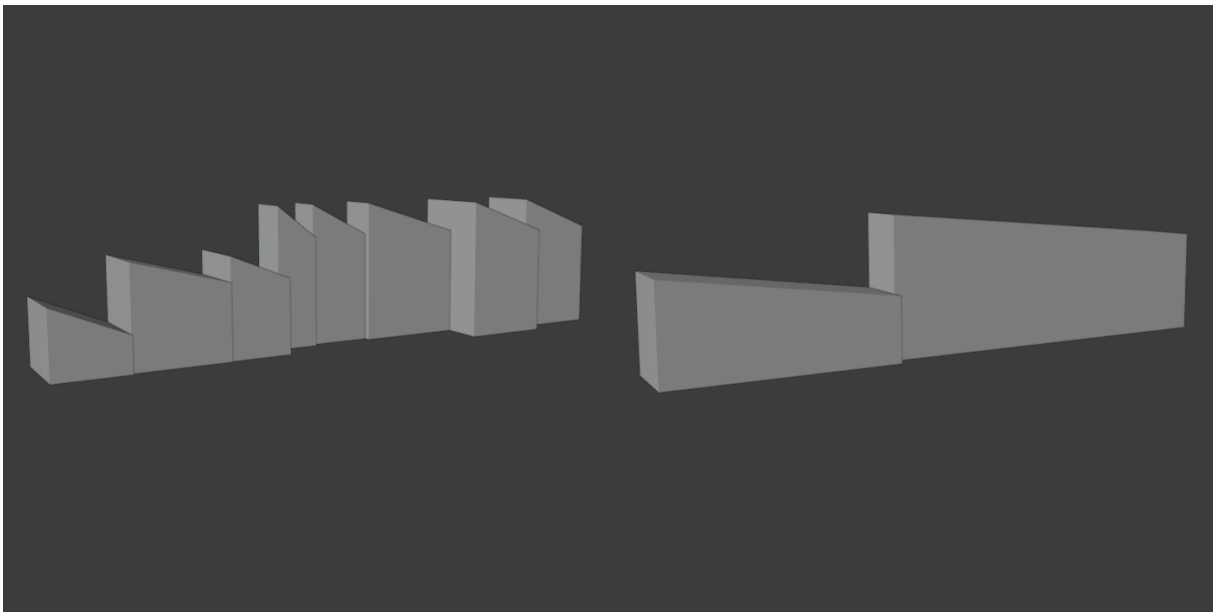
Obr. č. 12: Experimentální výsledek č. 1



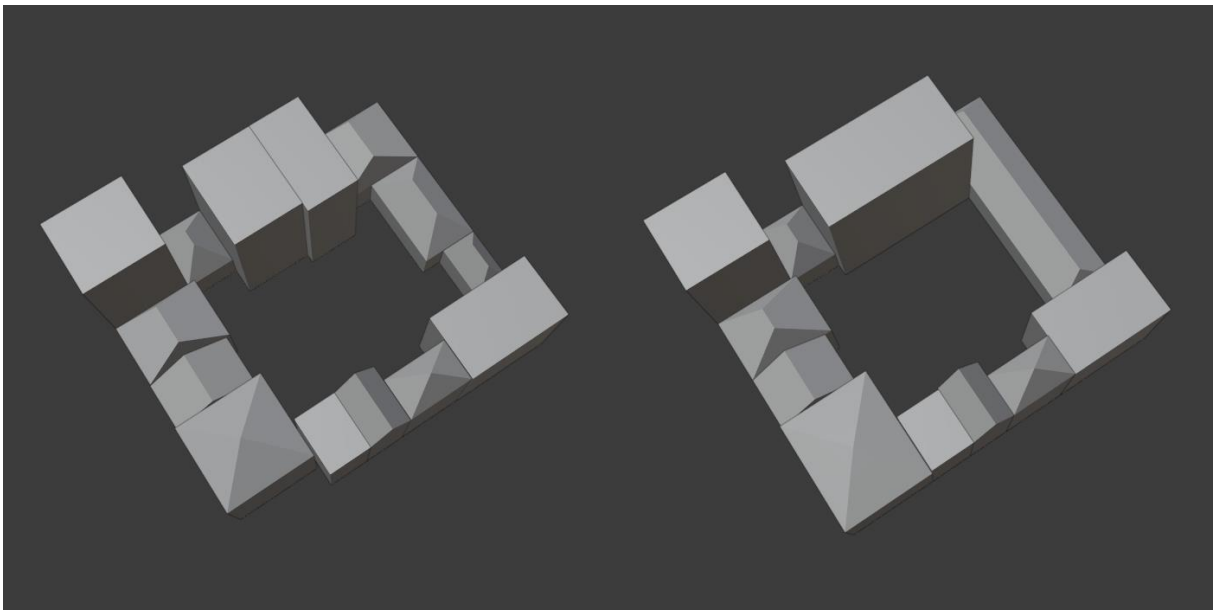
Obr. č. 13: Experimentální výsledek č. 2



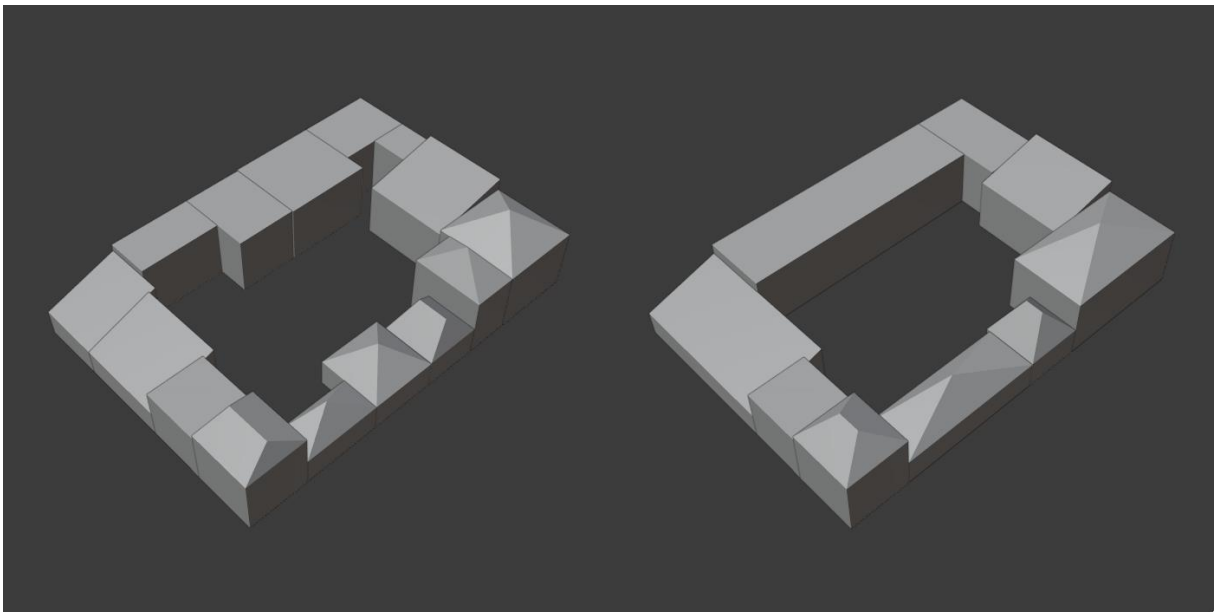
Obr. č. 14: Experimentální výsledek č. 3



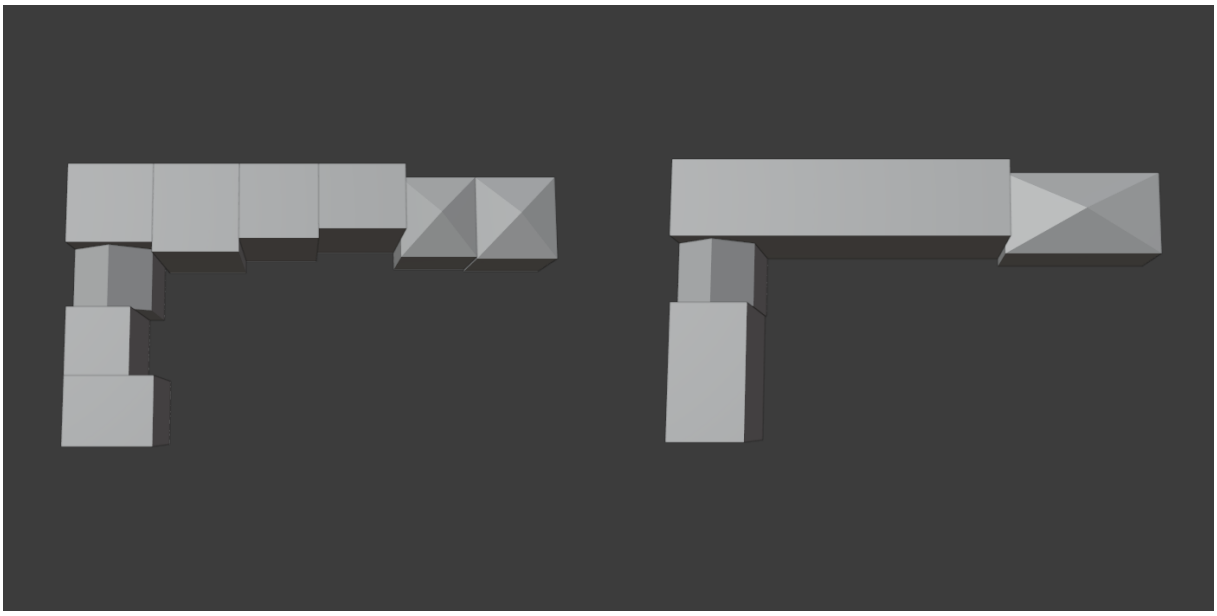
Obr. č. 15: Experimentální výsledek č. 4



Obr. č. 16: Experimentální výsledek č. 5



Obr. č. 17: Experimentální výsledek č. 6



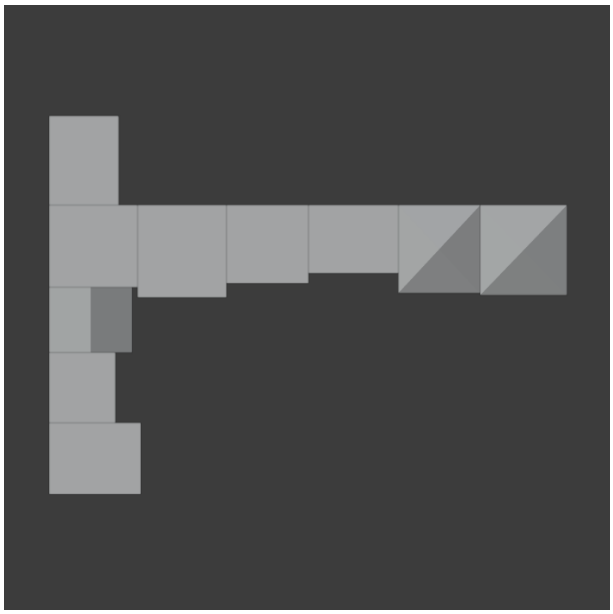
Obr. č. 18: Experimentální výsledek č. 7

Z výše prezentovaných experimentálních výsledků je patrné splnění předem definovaných cílů práce. Sousední budovy na sebe plynule bez výskytu mezer navazují. V případě obr. č. 15, kde jsou budovy poskládány do řady, tvoří společnou hranu půdorysu bloku, ke které jsou všechny budovy zarovnané, jedna strana stěn. V ostatních případech, ve kterých jsou budovy seřazeny do obdélníkového tvaru, tvoří tuto společnou hranu stěny na vnější straně bloku. Vnější strana půdorysu celého bloku je tedy vymezena pravoúhlým čtyřúhelníkem.

Výška těla budovy agregátu se změnila pouze v experimentálním výsledku č. 6 u budovy se sedlovou střechou. Výška těla samostatných (rohových) budov se měnila jen v případech, kdy původní model nebyl zarovnaný nebo obsahoval mezery mezi budovami. Tam kde se měnila výška těla budovy, měnila se i výška střešní části. Ke změně výšky střešní části došlo navíc i u všech budov s valbovým typem střechy.

S jedním výstupem práce Měchurová (2020) si však algoritmus neporadí. Problém spočívá v prostorovém uspořádání budov v bloku. Model je zobrazen na obr. č. 19. Funkčnosti algoritmu brání existence tří sousedních budov jedné z budov. Vždy na začátku jsou jednotlivé budovy zarovnávány v závislosti na pozici v bloku ke konkrétnímu souřadnicovému extrému. Budova na obr. nejvýše stanoví příliš vysoké maximum, ke kterému by měla být zarovnána část

bloku rovnoběžná s osou x. Odstraněním této budovy získáme validní výsledek viditelný na obr. č. 18.



Obr. č. 19: Model bloku s nevyhovujícím uspořádáním budov

8. Diskuze

Z výsledků bakalářské práce je patrné, že její metoda slouží pro vizualizaci výstupů optimalizační úlohy rozebrané v kapitole *3.3.2*. Díky implementované automatizaci pro specifikovaný typ vstupu v kapitole *4.1* není třeba programu dodávat další informace, či do jeho průběhu jakkoliv zasahovat.

Návaznost na Měchurová (2020) průběh metody této práce značně usnadnila. Vstupní 3D model zaručuje jistotu topologické korektnosti, a to díky metodě procedurálního modelování, jíž vznikl původní model. Zároveň nebyla potřeba provádět konverze mezi datovými formáty, neboť formát OBJ je k vizualizaci modelů vhodný. Dále byl díky předem známé informaci o typu střechy, potažmo o její orientaci usnadněn průběh skriptu. Vzhledem ke společnému kritériu objemové změny obou prací se naskytla možnost využít již existujících funkcí, viz kapitola *3.4*.

Úprava geometrie budov bloku byla provedena tak, aby snížila datovou zátěž výsledného modelu. Všechny dvojice sousedních budov mají na konci alespoň jeden společný vertex. Hlavní podíl na odstranění redundantních dat měla samotná aplikace agregace. V jednom případě se však drobná redundance nepodařila odstranit. Příklad je viditelný na obr. č. 9. Jde o situaci, kdy je agregát nacházející se na rohu bloku, rozdělen na dvě části. Tyto části se pak vyskytují zvlášť i ve výstupním souboru.

Úprava geometrie jednotlivých budov se neobešla bez změny výšek některých objektů. Předpoklad pro potenciální změnu výšek budovy nastává v situaci, kdy se její část nachází na rohu obdélníkového bloku. K modifikaci výšky těla agregátu dojde, pokud jeho rohová budova sousedí s dalším agregátem a po provedení agregace mezi nimi vznikne mezera. Dále ke změně výšky těla agregátu dojde v případě, že je šířka rohové budovy výrazně menší než šířky ostatních budov daného celku. U samostatných objektů dojde ke změně výšky těla rohových budov, které jsou od jejich sousedů odděleny mezerou nebo které nejsou zarovnány do výsledného pravidelného půdorysu. Tam, kde dojde ke změně výšky těla budovy, dojde i ke změně výšky střešní části. Algoritmus též nezachovává velikost výšky střešní části budov s valbovou střechou.

Skript je napsaný v programovacím jazyce *Python 3*. Využívá například základních matematických knihoven, jakými jsou *numpy* a *math*. Pro vizualizaci a interpretaci výsledků byl využit software *Blender*, který si poradí s načítáním OBJ formátu.

Zatímco metoda v této bakalářské práci usilovala o vizuálně vzhledné výsledky, další práce by mohla metodu rozšířit v rámci nové optimalizační úlohy a pokusit se o minimalizaci změny výšky objektů. K jedinému morfologickému operátoru – objemu, by se tedy mohla přidat šířka a výška. V případě nastavení volných podmínek pro změnu výšky by se výsledek shodoval s výsledkem této bakalářské práce. V opačném případě, kdy by došlo k minimální změně výšek, už by byl výsledný model vhodný pro jiné účely, jako jsou například letecké simulace.

Navržený algoritmus nevyhovuje všem výstupům práce Měchurová (2020). Proto by měla další práce ke vstupnímu modelu přistupovat nezávisle na jeho tvaru. Vzhledem k tomu, že optimalizační úloha (kapitola *3.3.2*) může provést agregaci budov s různým typem střech, bylo by potřeba definovat podobu střechy pro všechny možné případy. Počet případů se potom zjistí jako suma k-členných kombinací (pro k ležící v intervalu <2,5>) z 5 prvků.

Metoda prezentovaná v této práci se postarala o automatizaci posledního kroku práce Měchurová (2020). Jak bylo výše zmíněno, existují případy, ve kterých algoritmus nevrátí validní výsledek, nicméně pro většinu výstupů Měchurová (2020) program funguje spolehlivě.

9. Závěr

V této práci byla navržena vizualizace generalizace bloku budov metodou agregace. Vizualizací se rozumí úprava geometrie již provedené agregace na základě výstupu optimalizační úlohy definované v práci Měchurová (2020). Výchozí data představuje 3D model bloku budov s aplikovanými parametry vypočtenými v optimalizační úloze, viz kapitola *3.3.2*. Úprava geometrie v této práci byla provedena tak, aby minimalizovala změnu objemu výstupních dat. Výsledný model působí celistvě a zarovnaně. Metoda zcela zachovala velikost objemu původních objektů ze vstupních dat.

Z experimentálních výsledků je patrné, že metoda spolehlivě funguje pro výchozí blok budov uspořádaný do obdélníkového útvaru. Úprava geometrie v žádném z případů neporušila topologii žádné z budov. Provedené zjednodušení je na první pohled zřejmé.

Nakonec byly navrženy možné způsoby navázání na tuto práci. Další práce by mohla pro nové zpracování využít optimalizační úlohy formou dosažení minimální hodnoty účelové funkce. Dále by byl algoritmus třeba uzpůsobit pro všechny možné podoby výstupu práce Měchurová (2020).

Seznam použité literatury

DAMEN, J., KREVELD, M., SPAAN, B. (2008): High quality building generalization by extending the morphological operators. 11th ICA Workshop on Generalization and Multiple Representation, Montpellier, France, s. 1–12.

GE, L. (2018): Generalization of LOD2 buildings with different roof structures. Journal of Spatial Science, 4, s. 319–340.

GUERCKE, R., GÖTZELMANN, T., BRENNER, C., SESTER, M. (2011): Aggregation of LOD1 building models as an optimalization problem. IPRS Journal of Photogrammetry and Remote Sensing, vol. 66.

KADA, M. (2007): Scale-dependent simplification of 3D building models based on cell decomposition and primitive instancing. In: WINTER, S., DUCKHAM, M., KULIK, L., KUIPERS, B. (eds.): Spatial Information Theory. Springer Verlag, Berlin, s. 222–237.

KADA, M. (2011): Aggregation of 3D buildings using a hybrid data approach. Cartography and Geographic Information Science, 38 (2), s. 153–160.

LYSÁK, J. (2018): Generalizace: *činitelé, druhy generalizace, cenzální a normativní výběr, generalizační algoritmy* [přednáška]. PřF UK, Praha, 19. 11. 2018.

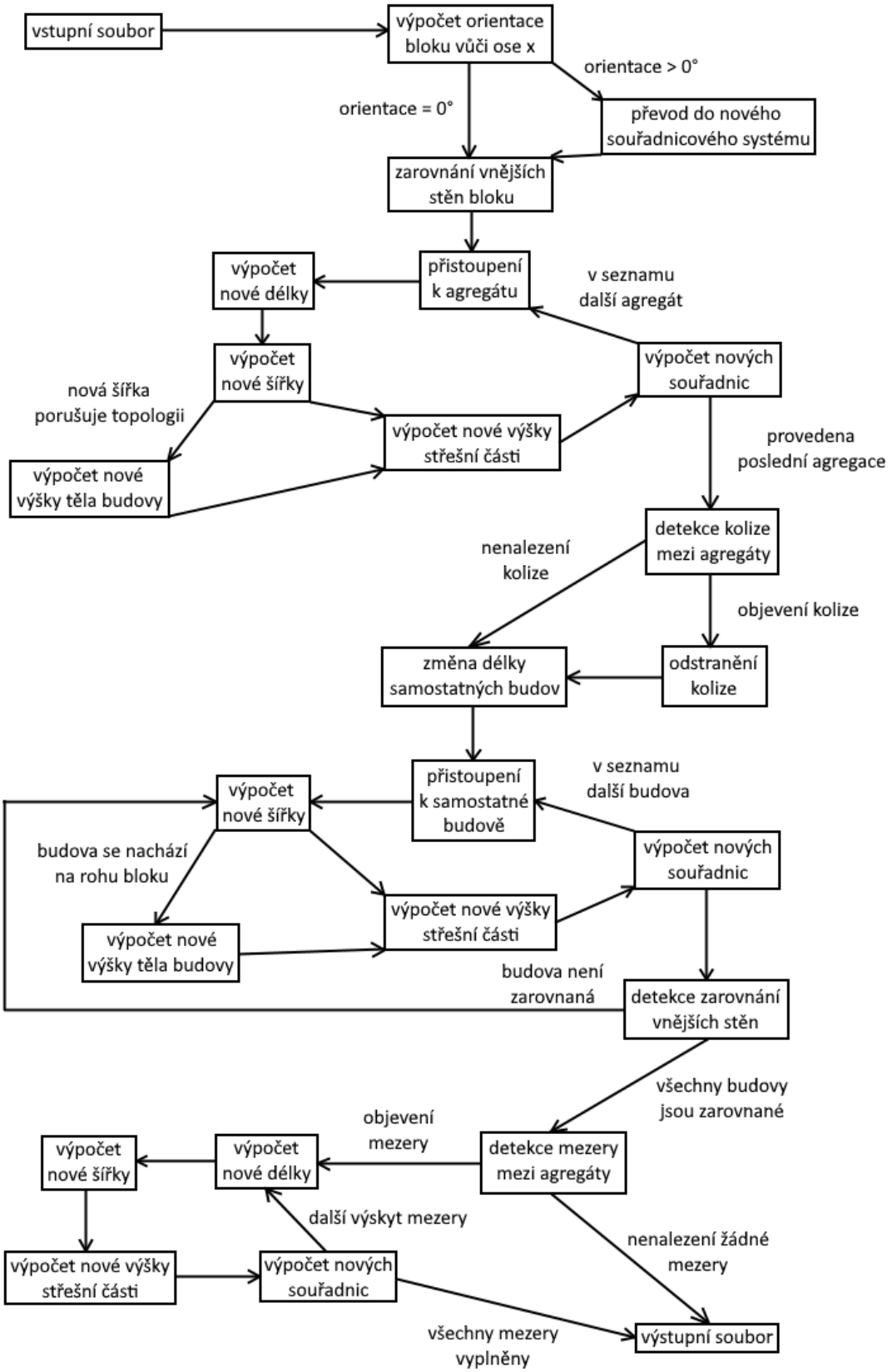
MĚCHUROVÁ, K. (2020): Generalizace LOD2 modelů budov metodou agregace. Diplomová práce. Katedra aplikované geoinformatiky a kartografie PřF UK, Praha.

SESTER, M. (2020): Cartographic generalization. Journal of Spatial Information Science, 21, s. 5–11.

XIE, J. a kol. (2012): Automatic simplification and visualization of 3D urban building models. International Journal of Applied Earth Observation and Geoinformation, 18, s. 222–231.

Přílohy

Příloha č. 1: Grafické schéma metody



Příloha č. 2: Skript automatizující vizualizaci včetně vstupních testovacích dat

https://github.com/sklibad/Bachelor_project