EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

TÉRKÉPTUDOMÁNYI ÉS GEOINFORMATIKAI TANSZÉK

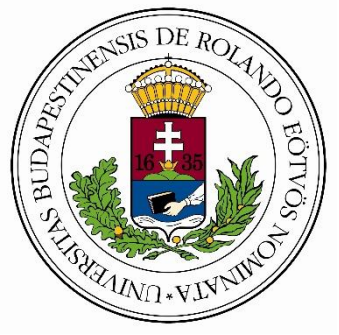
TÉRKÉPÉSZ MESTERSZAK

# Épületek generalizálása a gyakorlatban

DIPLOMAMUNKA

*Készítette:*
Kocsis Viktória
térképész hallgató

*Témavezető:*
Ungvári Zsuzsanna
tanársegéd
ELTE Térképtudományi és Geoinformatikai Tanszék




Budapest, 2017

Témabejelentő

# Tartalomjegyzék

# 1. Bevezetés

A térkép a valóság absztrakt megjelenítése, amelyben törekszünk a valóság minél pontosabb ábrázolására az adott méretarányban. A méretarány csökkenése a térkép felületének fogyatkozásához, és a térképi objektumok szimbólumainak növekedéséhez vezet, így gyakran konfliktust okoz az elemek között. Ahhoz, hogy a térkép alapvető tulajdonságait és olvashatóságát is megtartsuk, egy irányított egyszerűsítési folyamatra van szükség, amit generalizálásnak nevezünk.

A generalizálás során a térkép információtartalma csökken, egyszerűsödik. A vonalvezetések kevésbé összetettek, a kis utcák eltűnnek, és az épületek sokszögei is egyszerűbbé válnak. A folyamat nem csak a térképszerkesztésnél, hanem a térinformatikában és grafikában is fontos, mivel csökkenti a kirajzolandó elemek számát, így javítva annak idejét. Olyan módszerre van szükség, ami maximálisan automatizált, gyorsan elvégezhető, kevés manuális javítás szükséges utána és mégis képes megfelelő kartográfiai minőséget nyújtani.

Az épületek generalizálása a generalizálási módszerek között sajátos helyet foglal el, hiszen egyszerre két különböző hozzáállásról szükséges említést tennünk. Egyrészről szükség van az épületek alakjának egyszerűsítésére, emellett pedig az épületek számának csökkentésére is ahhoz, hogy a megfelelő olvashatóságot elérjük.

Dolgozatom célja, hogy bemutassam a jelenleg rendelkezésre álló módszereket. Bemutatom a különböző, az épületek alakjának egyszerűsítésére, illetve mennyiségük csökkentésére kidolgozott elméleteket. A széles körben használt térinformatikai programok alak generalizálásra használható algoritmusait és azok eredményeit vizsgálom, hasonlítom össze, végül pedig egy általam készített algoritmus működését és elméletét is bemutatom.

# 2. Épületek ábrázolása különböző térképeken

Az épületek ember által létrehozott önálló objektumok, építmények. Használatuk célja sokféle lehet, beszélhetünk lakóépületekről, kulturális vagy igazgatási célokra alkotott középületekről, különböző ipari és mezőgazdasági létesítményekről. Felületi elemként ábrázolandóak, de méretükből adódó sajátosság, hogy csak nagy és közepes méretarányokban jelennek meg. Formájuk jellegzetes, többnyire egyenes falakból állnak, amit csak ritkán szakít meg egy-egy íves elem, így térképi ábrázolásukkor ezekre a jellegzetességekre, megtartásukra és kiemelésükre szükséges odafigyelni. Másik fontos jellemzőjük, hogy az egyenes falak általában derékszög formájú sarkokban futnak össze, így erre is nagy figyelmet kell fektetni a szerkesztés során.

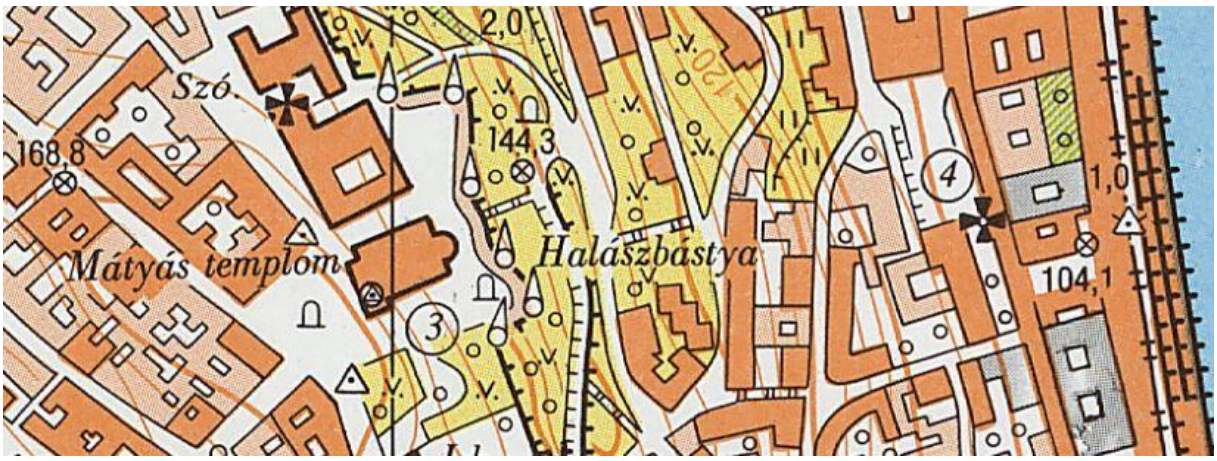
Az épületek megjelenítése nagy számban általában a kataszteri és topográfiai térképekre, illetve a tömegtérképek közül a várostérképekre, turistakiadványokra jellemző. Ezeken méretaránytól függően megjelenik az épületek túlnyomó része, vagy csak fontosabb építmények (látnivalók), alaprajzszerű ábrázolással, vagy pedig az épület formáját jellemző szimbólummal. A kisebb méretarányban történő megjelenítés esetén is törekednek az épület jellegzetes formáinak megtartására, hogy szükség esetén helyjelzőként való használatuk könnyebb legyen. Így az épületek szimbólumai is az eredeti formát jól közelítő poligonok. Megjelenítése a nagyobb méretarányokban jellemző, az 1:100 000 méretaránnyal bezárólag. Megjelenítésük vizsgálatához olyan térképeket választottam ki, amelyeken az általam későbbiekben vizsgált négy épület, a Mátyás-templom, Halászbástya, Hunfalvy János Szakközépiskola és a Vízivárosi Szent Erzsébet-templom valamelyike megtalálható.

## Épületek topográfiai térképeken

A topográfiai térképek topográfiai vagy fototopográfiai felmérés alapján készített és ezekből levezetett térképek, melyeken a térképi jelkulcs a felszín egyensúlyi ábrázolására törekszik (Faragó, 2014). Így megtalálható rajta a síkrajz mellett a domborzatrajz is. Méteraránya az 1:10 000-től 1:250 000-ig terjed, de önálló épületeket 1:100 000-ig találhatunk rajtunk.

Az 1:10 000 méretarányú EOTR szelvényeken minden önálló épületet megtalálunk, aminek mérete eléri a minimálisan olvasható térképi méretet a méretarányban. Az épületeket magasság szerint négy kategóriába osztva láthatjuk. A 12 méter alatti magasságúak (földszintes vagy egyemeletes) halvány narancssárga színezésűek, míg a legmagasabb építmények fekete

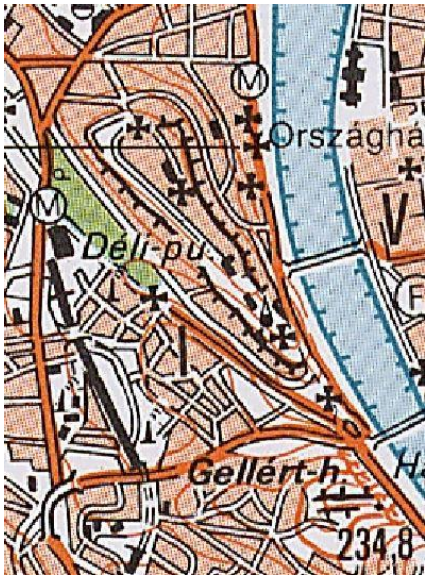
színt kaptak. A fontosabb épületek kiemelése vastag fekete körvonallal történt. A méretarány jellegzetessége, hogy a belvárosi társasházak sok esetben tömbönként egy poligonnal kerültek ábrázolásra, a külön kapualjakat összevonták, ha ugyanabba a magassági kategóriába tartoznak. A bennük található belső udvarok, átriumok viszont minden esetben láthatóak. Az ábrázolás jobb olvashatósága érdekében csak kisebb változtatásokat eszközöltek. A kicsi kiugró részeket eltávolították (ez látható például a szakközépiskola példájában) vagy eltolták a sarkok felé (Mátyás-templom). A Halászbástya bástyáit nem látjuk alaprajz szerint, a *toronyszerű építmény* szimbólumát láthatjuk helyükön.




*1. ábra: A Mátyás-templom, Halászbástya, Hunfalvy János Szakközépiskola és a Szent Erzsébet templom 1: 10 000 méretarányú EOTR szelvényen*

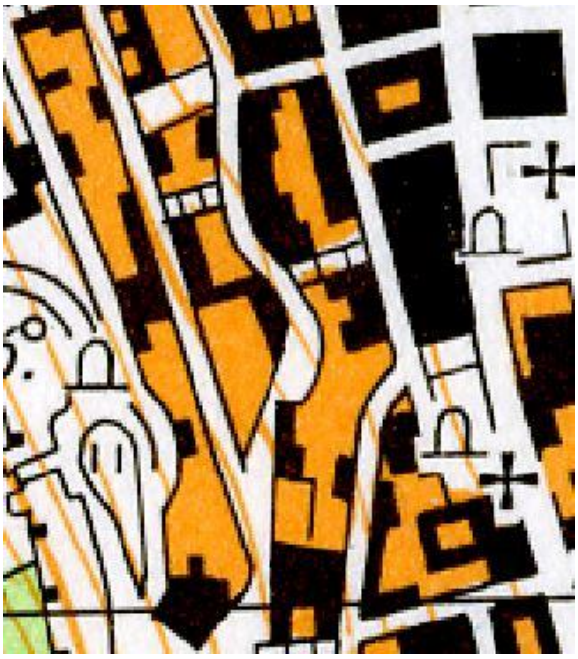
Az épületek falainak egyenessége nem minden esetben teljesül. Mivel az épületek összevonva láthatóak, az íves utcát körülölelő, egyenes homlokzatokból egy íves vonal lesz. Ezért a derékszögek sem pontosak mindenhol.

Az 1: 100 000 méretarányú EOTR szelvényeken már kevés épületpoligont láthatunk, azokat is jelentősen leegyszerűsítve. Ezek többnyire fontosabb, kiemelt funkciójú építményt jelölnek, iskolákat, templomokat vagy kórházakat. Ezek a poligonok többnyire egyszerű téglalapok. Sokszor azonban ezeket az épületeket sem láthatjuk felületként megjelenítve, hanem egy szimbólum jelzi elhelyezkedésüket. Ilyenek a templomok, a Mátyás-templomot illetve a Szent Erzsébet templomot is csak a szimbólum jelzi. A második ábrán látható, hogy a Budai Vár épületei közül csupán hármat láthatunk felületként megjelenítve.



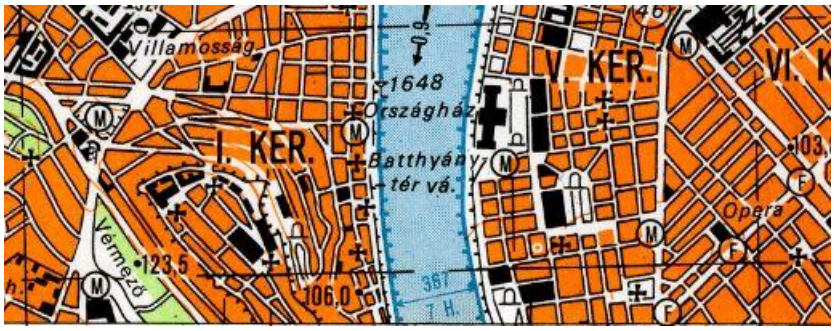

*2. ábra: A Budai Várnegyed és környéke az 1: 100 000 méretarányú EOTR szelvényen*

Az 1:25 000 méretarányú Gauss-Krüger szelvényeken a beépített terület ábrázolása kezd megjelenni az épületek poligonjai mellett. A különálló épületek közül a kisebbeket (főleg lakóépületek) már minden esetben téglalap formájúvá generalizálva láthatjuk, és számuk is csökkent az 1:10 000 EOTR szelvényen láthatóhoz képest. A társasházak tömbönként összevonva jelennek meg egy jelzésszerű belső udvarral. Ami még sajátossága ennek a térképnek, hogy a templomokat jelölő kereszt szimbólum mellett nem láthatunk poligont (ennek oka a szimbólum és a poligonok fekete színe). Az épületek oldalai sokszor nem egyenesek, inkább az utca vonalát követik, és így egészen érdekes formák is kialakultak, ez látható a harmadik ábrán.



*3. ábra: Eltorzult épületek az 1:25 000 méretarányú Gauss-Krüger szelvényen*

Az 1:50 000 méretarányú Gauss-Krügeren már sokkal kevesebb épület látható. A beépített terület színezést kapott, a fontosabb épületek találhatjuk csak meg rajta. Ezek közül a kisebbek erősen generalizáltak, téglalapként jelennek meg, a jellegzetes épületek láthatjuk alaprajszerű ábrázolással. Az épületek szögletessége jól követhető, az eredeti vonalvezetést még jól közelítő poligonokat láthatunk a térképeken.



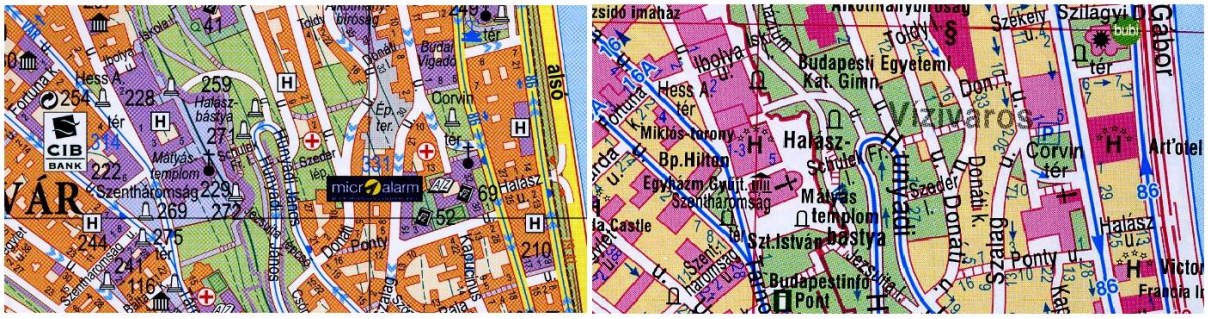

*4. ábra: Az I. és V. kerület az 1:50 000 méretarányú Gauss-Krüger szelvényen*

## Épületek megjelenítése várostérképeken

A várostérképek célja az adott településen belüli közlekedés, tájékozódás segítése. Ábrázolása eltolódik a mesterséges elemek kiemelése felé, legfontosabb elemei az utak, utcák és azok nevei, valamint a különböző épületek. Kiadványtól függően megtaláljuk rajta a fontosabb tömegközlekedési csomópontok jelzéseit, a kisebb-nagyobb településrészek neveit, kiemelt épületek neveit, vagy típusát szimbólummal jelezve. A térkép háttéranyaga többnyire a felszín fedettsége, hol találunk erdőket, parkokat, füves területek, és melyek a sűrűn lakott, beépített részek.

A várostérképek méretaránya a nagy és közepes méretarányok közé sorolható, többnyire többezrestől néhány tízezerig terjed. Méretarányából adódóan lehetőség van az egyedálló épületek ábrázolására, nagyobb méretarányban akár az összesre is, a térkép célja ezt nem engedi meg. Az épületpoligonokkal zsúfolt térképen nehézkes lenne a tájékozódás, ezért a jelentősen kiemelt utcahálózat mellett csak a fontosabb illetve nagyobb épületek megjelenítésére van lehetőség.

A legnagyobb méretarányú kiadvány az 1:7500-as Budapest belváros térkép. A térképen az úthálózat nincs túlzottan kiemelve, és a méretarányból adódóan elég sok épület található rajta. A fontosabb épületek kiemelése lila színnel történt, sötétebb naranccsal láthatjuk a magasabb épületeket, világosabbal pedig az alacsonyabb építményeket. Az egyedülálló épületek nem kerültek összevonásra, viszont a Belváros társasházai tömbönként egy poligonként jelennek meg. Ezeken legszembetűnőbb a generalizálás, nem csak összevonásra kerültek az épületek, hanem a közöttük levő kisebb átriumokat, belső udvarokat is összevonták, vagy eltávolították. Az egyedül álló épületeknél is észrevehető, a sokszögeknél a kisebb kiugrások törlésre kerültek, azokat téglalapokkal helyettesítették. Az épületek ívei mindenhol megmaradtak, de látszólag minden poligon egyszerűbben jelenik meg, a Halászbástya bástyái például nem láthatók.




*5. ábra: A 1:7500 és a 1:12 000 méretarányú várostérképek*

A 1:12 000 méretarányú Belváros kivágaton csak a fontosabb épületek jelennek meg, míg a nagyobb méretarányú térképen a lakóépületeket is láthattuk. Különböző színnel láthatjuk a beépített területeket, és a parkokat. Az épületek generalizáltak, de nagyon jól jellemzik az eredeti formákat. Sokszor a belső udvarokat elhagyták, vagy a jobb megjelenítés érdekében nagyították. Az 5. ábrán látható, hogy a nagyobb háztömbök épületei nem összevonva jelennek meg, a legnagyobb méretaránnyal ellentétben. A falak egyenesek (néhány kivételtől eltekintve), de a derékszögű sarkak meghagyása helyett az épületek az utcák ívét követik. A Halászbástya ezen a térképen már nem poligonként, hanem a várfal vonalának részeként jelenik meg.

Az előbbi két méretarányban viszonylag nagy számban fordulnak elő épületek, míg a 1:15 000 méretarány közeli térképen számuk már jelentősen csökkent. Ezen csak néhány fontosabb épületet találhatunk. A kiemelt épületek barna színezéssel szerepelnek, ezek többnyire a híres turistalátványosságok, templomok, múzeumok és pályaudvarok. Az ezeken kívül megjelenő épületek az adott kerület beépített területének színével láthatók. Ilyenek például az egyetemi kampuszok épületei, és a kórházak komplexumai. Az épületek generalizálva, de a méretarány által megengedett maximális részletességgel láthatók. A derékszögek és az egyenes falszakaszok nagyon jól láthatóak, nincsenek eltorzulva.

6. ábra: A Vár 1:15 700 méretarányú várostérképen

A 1:20 000 térképen az épületek már valódi méretükön felül ábrázolva láthatók. Csak a fontosabb épületeket, kulturális és igazgatási központokat láthatjuk megjelölve. Poligonjaik téglalapként jelennek meg, jelentősen leegyszerűsítve, és kihasználják az utcák által körbezárt tömbök területeit. A központi épületek mellett láthatjuk a háztömböket is, a társasházak poligonjait is néhány helyen.

7. ábra: A Vár 1: 20 000 méretarányú várostérképen

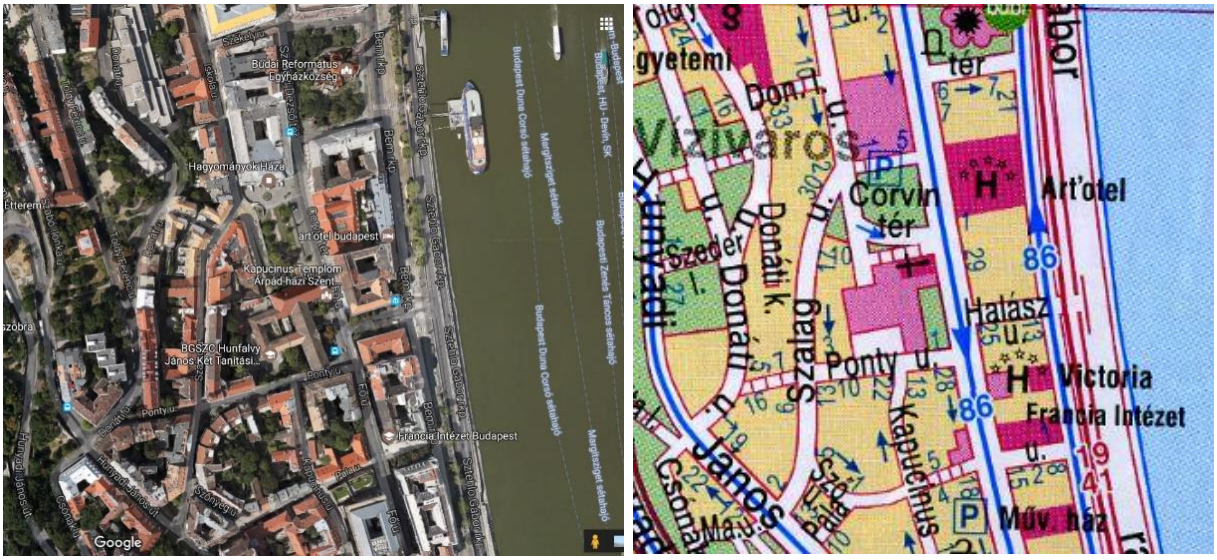
Kisebb (1:30 000) méretarányban már jóval kevesebb épületet láthatunk a térképen, a fontosabb épületek is kisebb számban láthatók. A beépített terület nem csak tömböket jeleníti meg, hanem ahol ritkábban állnak épületek, ott formájukat követve láthatjuk azokat. Ilyenek például a lakótelepek, irodaépületek és belvárosi társasházak. Kiemelt épületként láthatjuk a kórházakat, nagyobb egyetemi központokat, turistalátványosságokat. Ezek generalizálva, de a részleteket jól mutatva jelennek meg. A kisebb, köríves részek, például a Mátyás-templom épületén megmaradtak.

A várostérképek jellemzője, hogy az utcák hangsúlyozása miatt a méretaránynál megengedhető épületmennyiségnél jóval kevesebbet láthatunk. Azt tapasztaltam, hogy csak a turisztikai, igazgatási és közlekedési szempontból fontos épületek, kórházak (rendelőintézetek) és nagyobb iskolák jelennek meg alaprajzszerű ábrázolással. Ezek generalizálásának mértéke változó, hiszen a kisebb (1: 15 000–1: 20 000) méretarányú térképen is részletes rajzolattal láthatjuk őket. Ezen kívül a beépített területet láthatjuk, ami a parkok melletti területen többnyire az épületeket jól követő poligonnal, míg a város más részein a háztömbök egészét lefedő poligonnal jelenik meg.


*8. ábra: 1: 30 000 méretarányú várostérkép*

# 3. Az épületek generalizálása

A városi területek épületek egyszerűsítéséhez több generalizálási folyamat használatára is szükség lehet. Ezek az elhagyás, eltolás, összevonás, kiemelés, tipizálás, illetve a poligonok egyszerűsítése részletek elhagyásával és a sarkok derékszögesítésével (Damen et al, 2008). Az, hogy ezek közül az operátorok közül melyikek használatát létesítjük előnyben, attól függ, hogy az épületek generalizálásának melyik lépését nézzük, hiszen nem egyszerűen az alak egyszerűsítése a cél. Az ábrán jól látható, hogy nem csak az épületek nagy részét hagyták el, de a megmaradt épületek rajzolatát is egyszerűsítették. A templom belső udvarát nem ábrázolják, a hotel épületét (templomtól jobbra fel) egyszerűen egy téglalappá generalizálták.




*9. ábra: Az Árpád-házi Szent Erzsébet templom és környéke műholdképen és egy 1:12 000 méretarányú turistakalauz térképén (nem méretarányos)*

A generalizálás három alfolyamatra osztható:

- az épületpoligonok generalizálása
- épületek számának csökkentése
- átfedések kezelése, és az utakhoz való igazítás.

Az első két lépés csak tisztán egyszerűsítési folyamat, melynek elsődleges célja a térképi tartalom csökkentése, és végrehajtásuk sorrendje felcserélhető. Az utolsó módszer a tartalom olvashatóságát, minőségét javítja, az előző kettő folyamat végrehajtása után létrejövő térbeli konfliktusokat oldja fel. Magától értetődő, hogy a poligonok generalizálását a részletek elhagyásával és derékszögesítéssel hajtjuk végre, az épületek számát elhagyással, összevonással, tipizálással, de szükség lehet egyes fontosabb poligonok kiemelésére is. Az átfedések kezelésére pedig az eltolást, és szükség esetén az elforgatást használhatjuk.

## Az épületpoligonok generalizálása

A poligonok generalizálása a legegyszerűbb folyamat a generalizálás során. Nem szükséges nagy hangsúlyt fektetni a térbeli kapcsolatok megtartására, az átfedések lehetőségére, hiszen a cél csak tisztán az alak egyszerűsítése. A cél a kis kiugrások, sarkok, belső udvarok, vékony folyosók eltávolítása vagy szükség esetén kiemelése, hogy a lecsökkent térképi területen is jól elkülöníthetőek és olvashatók legyenek az épületek.

A poligonok generalizálásának legegyszerűbb megközelítése a vonalegyszerűsítésre használt algoritmusok sokszögre való implementálása. Mivel itt is a kisebb kiugrások, futásbeli fluktuációk eltávolítása a cél, elméletben ezek megfelelhetnek a célnak. A legismertebbek a Douglas–Peucker, Visvalingam–Wyatt, vagy a Wang–Müller algoritmusok, melyek mind vonalgeneralizálásra lettek kifejlesztve, de poligonok egyszerűsítésére is előszeretettel használtak. A kizárólag épületgeneralizálásra létrehozott módszerektől eltérően ezek nem veszik figyelembe azokat a plusz feltételeket, amik az eredmény megfelelő minőségét garantálják. Fontos szempont, hogy a generalizálás után az épületpoligonok területe ne változzon jelentős mértékben, csak egyszerűsödjön a rajzolata, fő formáját és a derékszögeket tartsa meg 90°-osnak.

Ezek fordulnak elő a térinformatikai programokban leggyakrabban, a végrehajtásuk eredményét a dolgozatom következő fejezetében mutatom be.

A 2000-es évek elejétől kezdve több, nem a vonalegyszerűsítésen alapuló poligongeneralizálási módszert fejlesztettek ki, ami nem csupán a csomópontok számát csökkenti, hanem figyelembe veszi az épületek jellegzetességeit, a derékszögeket, egyenes oldalakat, és belső udvarokat is.

### Épületek alakjának generalizálása minimumkereső eljárással (Sester, 2000)

Az épületek generalizálásának ezen megközelítése három szabályt vesz alapul. Az eredmény tartsa meg a derékszögeket, a párhuzamosságot, és az vonalra illesztett épület térbeli kapcsolatait. Az egyszerűsítés kiváltója az algoritmus futtatásához kiválasztott minimális homlokzatméret, ami a célméretarányunkban ábrázolható. Az ennél kisebb falakat ki kell cserélni, vagy pedig elhagyásuk szükséges.

Az algoritmus egyenként vizsgálja a poligonokat, és lokálisan működik, egyszerre csak egy oldalt elemez, azt cseréli ki. Ez megadott szabályok alapján történik, amik a szomszédos oldalak kapcsolatát veszik alapul. Az eredmény egyszerű, de nem minden esetben veszi

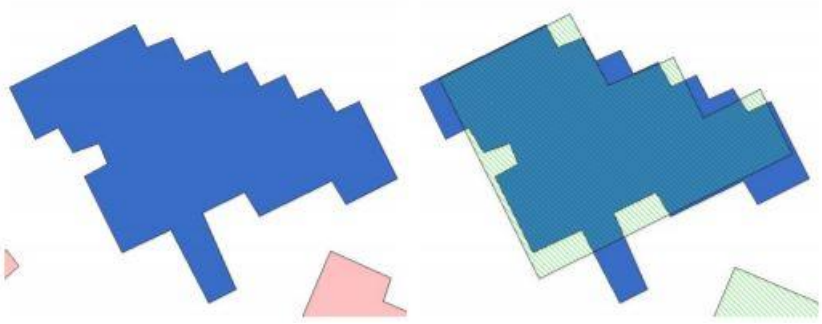
figyelembe az eredeti épület jellemző részeit, így egy plusz fázisra is szükség van. Ez az igazítás, amely során a létrehozott modellt az eredeti adatsorral összehasonlítják. Az algoritmus előnye, hogy a szabályok triviálisak és durván közelítenek, de az igazítással eredmény jól finomítható.

Az első lépés (maga az algoritmus) az alaprajz egyszerűsítéséhez vezet, amit modellnek nevezünk. Az épületek jellemző tulajdonságait fejezi ki, de csak az igazítás fázisa után tükrözi megfelelő minőségben az eredeti állapotot.

A rövid homlokzati elem – fal – egyszerűsítése a szomszédos elemek egymással való kapcsolatától függ, három lehetőség van:

- Benyomulás vagy kitörés: a vizsgált falat megelőző és követő falak által bezárt szög megközelítőleg 180°. Ilyenkor a falat a főbb futású fal vonalába mozgatjuk.
- Falkiugrás: a megelőző és a következő oldal által bezárt szög 0°-hoz közelít. A szomszédos oldalak közül a hosszabbat (tehát jellemzőbbet) megnyújtjuk, a rövidet pedig elhagyjuk.
- Sarok: a két falrész nagyjából derékszöget zár be, a szomszédos oldalak metszik egymást.

A poligon minden oldalát megvizsgáljuk, a legkisebbel kezdve, és végrehajtjuk rajta a három szabály közül a megfelelőt. Az algoritmus nagyon jól egyszerűsíti a jellemző épületrészeket, ahogy a x. ábrán is látható, de van néhány eset, amikor szükség lehet a forma kihangsúlyozására. Ha hosszú, keskeny épületrészek (pl. kinyúló folyosók) keskeny oldala kisebb, mint az egyszerűsítéshez kijelölt küszöbérték, az egész forma törlésre kerül. Fontos ezért bevezetni még egy küszöbértéket, ami a minimális területnagyságot adja meg. Így a megfelelő méretű poligonelemek fontos részként kezelhetők, és a keskeny oldal minimálisra növelésével megőrződik.



*10. ábra: Példa az algoritmus működésére*

Az egyszerűsítés után létrejött modellt paraméteres formában adjuk meg: az épületet tengelyekkel reprezentáljuk. A paraméterekkel könnyen megadható az épületek formája, például egy téglalap esetén két értékre van szükség. Ezeket méretparaméternek nevezzük, és a koordinátákkal, vagyis a pozíció paramétereivel kiegészülve alkotják azokat az ismeretleneket, amiket az algoritmus felhasznál.

A forma paramétereiből függvénnyel megadhatók az eredeti épületoldalak. Ezeket a függvényeket kiegészíti egy sztochasztikus modell, ami a pontosságot írja le, hogy az eredeti illetve egyszerűsített élek milyen kapcsolatban vannak. A modellel való egybeesést vizsgálja, a szakasz kezdő és végpontjának pontosságát, valamint a két él pozícióját.

A rövidebb élek többnyire kevésbé pontosak, mint a hosszabbak, hiszen az egyszerűsítés után a hosszabbak jobban közelítik az eredetit (lásd a falkiugrás egyszerűsítését). Emellett bevezethető egy újabb változó, aminek használatával megadhatunk egy tetszőleges pontosságot a szakasznak. Erre abban az esetben van szükség, ha egy paramétert meg kell erősíteni, amikor kiemelés történt. Ezt a változót ilyenkor nagy értékkel kell ellátni, mely általában megegyezik a minimális élhosszal.

Az igazítás fázisa során az elemek helyzetét kell javítani, egy meghatározott szabályrendszer alapján. A használt változók az épületek koordinátái, így minden szabály leírható a koordináták függvényeként, és pontosságot is rendelhetünk hozzájuk. A szabályok kétoldalúak, tartaniuk kell a minimális távolságot két különálló objektum között, és emellett a poligon minimális belső távolságait, belső szabályait is figyelembe kell vennie. A bevezetett szabályok:

- Az alak paraméterei: oldalak, irányok és szögek
- Minimális táv: a minimális távolság, amire törekedni kell, illetve bevezetendő egy kritikus távolság, ami alatt a távolság nullává válik, vagyis a poligonok összeolvadnak.
- További paraméterek: koordináták.

Az algoritmus egyszerű és vonzó eredményt ad, de a szerző szerint vannak hiányosságai. A belső udvart külön elemként kezeli, így előfordulhat, hogy az egyszerűsítés után metszeni fogja a külső poligont. A belső részek (mint például egy U alakú épület két szárnya által bezárt rész) minimális nagyságára vonatkozó szabályt nem veszi figyelembe, így azok összezáródhatnak. Ennek megoldására egy új változót kell bevezetni az algoritmusba. A

módszer legnagyobb előnye, hogy a derékszögek megtartását figyelembe veszi, ellenben, ha az épület kettőnél több fő tengellyel rendelkezik, a derékszögek megmaradása nem garantálható.

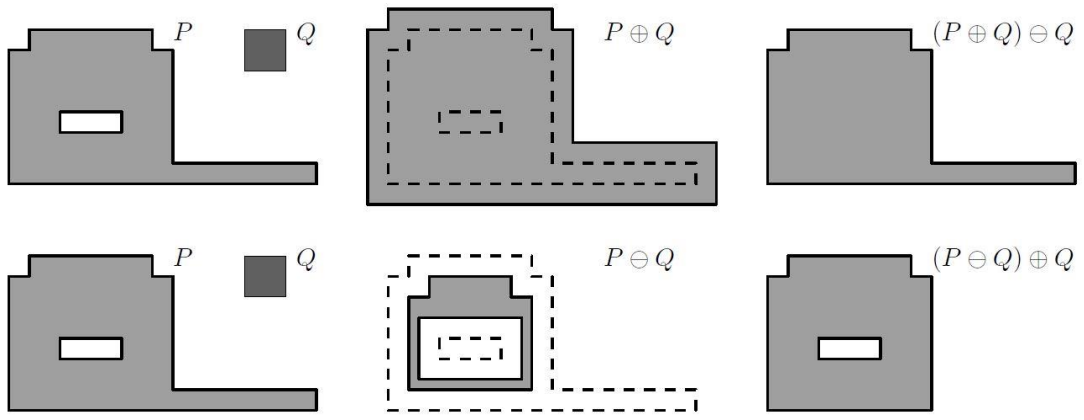
## Morfológiai operátorok és kiterjesztésük (Damen et al. 2008)

A morfológiai (alaktani) operátorok használata először a raszteres adatok egyszerűsítését segítette. A célja, hogy egy matematikai modellt építsen fel a két, matematikai alaktan alapján fejlesztett módszerre, az úgynevezett tágulásra (dilation) és pusztulásra (erosion). Ez a két művelet az alaktanban hasonló alapműveletnek számít, mint az összeadás vagy kivonás az algebrában. A matematikai alaktan alapjait két francia geostatisztikus, G. Matheron és J. Serra hozta létre. Lényege a geometriai struktúrák elemzése és feldolgozása, a halmazelmélet, háló (matematikai értelemben), topológia alapján. Azóta a digitális képfeldolgozásban és felületi hálók (surface mesh) vizsgálatához használják elsősorban. Később megpróbálták az alaktani eszközöket vektoros adatmodellhez igazítani, a térkép generalizálásához, és a digitális domborzatábrázoláshoz is felhasználni. A két alapművelet, ami a generalizálás alapját is szolgálja, a következő:

- Tágulás (Minkowski összeadás): $A \oplus B = \{a + b | a \in A; b \in B\}$
- Pusztulás (Minkowski kivonás): $A \ominus B = Komplementer\{a + b | a \notin A; b \in B\}$,

ahol A a feldolgozandó bináris kép, B egy kernel. Vektoros adatrendszerben való használatánál A és B egy-egy poligont jelölnek.

A kernel kritikus része a morfológiai műveleteknek, kiválasztása leginkább a konvolúcióban használt kernelmátrixhoz hasonlítható. Bármilyen formát felvehet, lehet négyzet, kereszt, kör, illetve mérete is változó.




*11. ábra: Az alaktani műveletek eredménye P poligonra Q kernel használatával*

A két alapműveleten alapul a generalizálásra használt két művelet, a nyitó, illetve záró művelet. A nyitás $(A \oplus B) \ominus B$, a zárás pedig $(A \ominus B) \oplus B$. A két művelet eltérő hatással van a poligonokra. A zárás inkább a kisebb lyukakat távolítja el. Egy poligon esetén a belső udvarokat, lyukakat, míg két közeli épületet össze is olvaszthat, a közöttük levő „folyosó" egyszerűsítésével. Ezzel szemben a nyitás a bonyolultabb felépítésű poligonokat szétoszthatja, illetve a belső udvarokat kiemeli, ha több van, azokat egybeolvasztja.

Mivel a két művelet végrehajtása kis hibákat eredményezett, szükség van egy plusz lépéssel való kiegészítésre, amitől a generalizálás jobb minőségű eredményt mutat. A záró művelet hosszú, keskeny épületrészeket, míg a nyitó művelet hosszú, keskeny belső udvarokat, és keskeny ösvényeket hagy az épületrészek között. Az egyik megoldás a két művelet egymás után való alkalmazása. Az eredményben jelentős szerepet játszik a kiválasztott kernel mérete és formája. A legideálisabb forma a kör lenne, de mivel mesterségesen létrehozott elemekről van szó, amelyek sarkai derékszögek, a négyzet választása a legkézenfekvőbb. A négyzet orientációja sem mindegy, ezek mind más és más generalizálási eredményt hoznak. A legmegfelelőbb, ha az oldalak ugyanolyan irányultságúak, amilyenek a vizsgált épület oldalai. A négyzet elforgatható, így minden épületre a legideálisabb eredményt hozza.

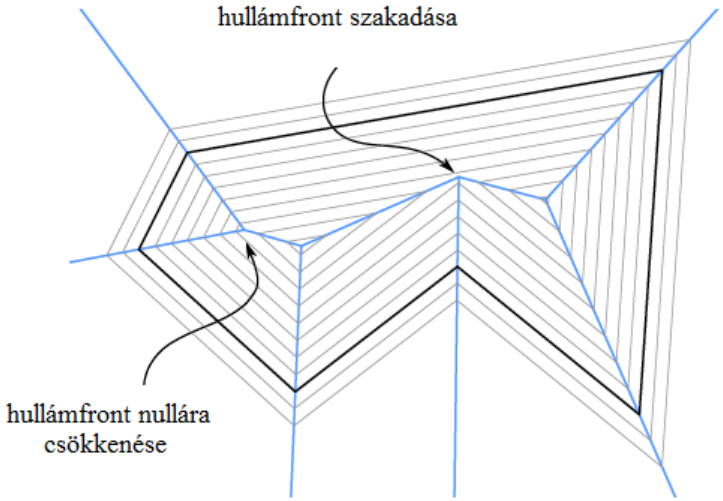
A generalizálás mértéke nagy részben a kernel méretétől függ, minél nagyobb, az eredmény annál erősebben generalizált. De ennek ellenére vannak kis élek, amik nagyobb érték esetén is megmaradnak. Nagy érték használatakor a nyitó műveletet követő záró művelet során sok poligon eltávolításra kerül, viszont kicsi érték esetén nincs jelentős különbség a megfordított sorrendben végrehajtott műveletek eredményi között. Azonban a két művelet kombinálása sem távolítja el a túl kicsi éleket. Ehhez szükség van egy él egyszerűsítő operátorra, ami a már ismertetett minimumkereső eljáráson alapul.

## Az egyszerű vázból létrehozott eltolt görbék módszere (Meijers, 2016)

A Martijn Meijers által kidolgozott módszer célja, hogy az előző tágulás és pusztuláson alapuló generalizáláshoz hasonló folyamatot hozzon létre (Meijers, 2016), de az egyszerű váz (straight-skeleton) alapján létrehozott görbék használatával. A váz nem csak poligonokból, hanem egyenes élű síkgráfokból is létrehozhatók. Az generalizálás inkább különálló épületek egyszerűsítésére született, de használható beépített területek egybeolvasztására is.

Az egyszerű vázat Aichholzer és kollégái (Aichholzer et al. 1995) által kifejlesztett, úgynevezett hullámfront-terjedés folyamat során hozhatjuk létre. Ennek folyamata:

- egy önmagát nem metsző, egyszerű poligonból kell kiindulni
- a sokszög minden éle kibocsát két párhuzamos hullámfront-vonalat
- két szomszédos frontvonal metszéspontjában található a hullámfront-csúcs, ami a két frontvonal szögfelező egyenesén mozog
- a frontok terjedése során több esemény is történhet:
    - a hullámfront vonala nulla hosszúságúra csökken
    - a hullámfront vonala szétszakadhat, ahol találkozik egy másik frontvonallal
- az egyszerű vázat a terjedés során, a hullámfront-csúcsok nyomvonala rajzolja ki.




*12. ábra: Az egyszerű váz kialakítása*

A vázat felhasználva valós adatokat is egyszerűsíthetünk. Ennek egyik módszere, hogy eltolt görbéket hozunk létre. Poligonok esetén ezeket a görbéket mind a poligon belseje felé, mint pedig azon kívül megadhatjuk. Az adatsoron a váz generálása közben a hullámfront terjedésének t időpontjában megvizsgáljuk minden csúcs két szomszédos csúcsát. Az adott t időpontra vonatkozó csúcsok fogják megadni az eltolt görbét alkotó pontokat.

A másik módszer, hogy a hullámfrontok mozgását megállítjuk, amikor az előre megadott távolságot elérte. A frontvonalakból így egy kisebb poligon jön létre, aminek rajzolata egyszerűbb, mivel a mozgás eltávolított néhány részletet.

Meijers az előző három módszer alapján építette fel az épületek körvonalának egyszerűsítésére és több poligon összeolvasztására is használható algoritmusát. Lépései a következők:

1. Az egyenes váz kiszámítása minden poligonra
2. Poligonon kívüli, egy megadott *e* távolsággal eltolt görbe létrehozása (ez egyszerűbben fogalmazva az eredeti poligon kitágítása)
3. A váz újraszámítása az új poligonra
4. A poligon eltolt görbéjének számítása ezúttal befele, a használt *e* küszöbértékkel

Az eljárás alapja ez a négy lépés, amiben felcserélhető a két görbe létrehozásának iránya. Ezen kívül további két lépés is hozzáadható, ha szükséges.

5. Váz újraszámítása
6. Görbe létrehozása (az elsőnek végrehajtott irányba)

A tanulmány kitért arra is, hogy az irányok felcserélése és a plusz lépés hozzáadása mennyire hoz különböző végeredményt. Ennek vizsgálatához egy templom körvonalát vették alapul. Az alapmódszer végrehajtásakor először kifelé, majd befelé haladva, a kisebb részleteket sokkal jobban megőrzi, míg a fordított eljárás erősebben egyszerűsít, de az épület egy nagyobb részletét is eltűntette. Ennek kiküszöbölésére egy kisebb *e* küszöbérték választhatása a megoldás. A jelentős különbség miatt a kibővített eljárás során a második és hatodik lépésben a küszöbérték felét alkalmazzák, tehát először *e* felével tolják el befelé, majd *e* értékével kifelé, majd ismét az érték felével befelé ismét. Ezeket az irányokat felcserélve is végrehajtották. A bővebb folyamat már jobb eredményhez, megfelelőbb generalizálási minőséghez vezet. A nagyobb, jellemzőbb részleteket megtartja, míg a kisebbeket eltávolítja. Viszont a „ki-be-ki” módszer létrehozott egy új beugrást, ami korábban nem tartozott a körvonalhoz. A kisebb éleket viszont egyik módszer sem egyszerűsítette, de ez mind manuálisan, mind egy algoritmus használatával javítható. Az eltolt görbék létrehozásának sorrendje egy több poligonból álló adathalmazra is hasonló eredményeket ad: az első esetben a kisebb részletek megmaradnak, de egybeolvadnak a poligonok, míg a fordított eljárás során nem keletkezik egy poligon.

A módszer előnye, hogy a végeredmény teljesen független az épületek tájolásától, nem szükséges azok elfordítása. A nehézség az, hogy a használt *e* eltolási távolságot előre szükséges

megadni, és nehezen lehet megbecsülni, hogy milyen generalizálási erősséget és minőséget hozhat.

## Épületek számának csökkentése

Ha célméretarányunkban csupán az alaprajz generalizálása nem hoz megfelelő minőségű végeredményt, további generalizálási módszer végrehajtására lehet szükség. Mivel az épületek poligonja további egyszerűsítés során már nem lenne elég informatív, így az épületek számának csökkentéséhez kell fordulnunk. Ennek két eleme van, az egyszerű eltávolítás, illetve az épületek tipizálása. A tipizálás során több, azonos tulajdonsággal rendelkező elemet egy csoporttá alakítunk, és egy jellel helyettesítjük ábrázolásukat, úgy hogy helyzetiket megfelelően bemutassa.

### Hálózat egyszerűsítése (Burghardt és Cecconi, 2007)

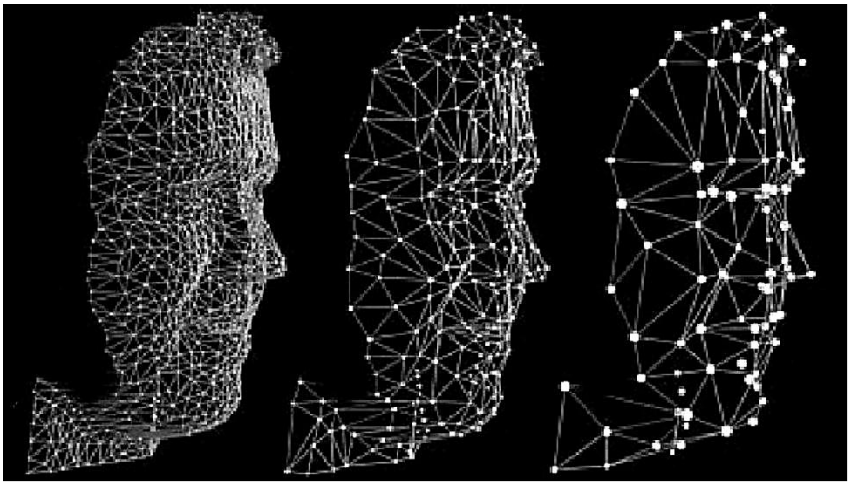
A számítógépes vizualizációban használt hálózatoptimalizálási technika elmélete ihlette a generalizálási módszert, ami legfőképpen a dinamikus térképkészítéshez kifejlesztett, gyors és rugalmas folyamat. A módszerek az felmért mintaadatok alapján rekonstruálják a különböző felületeket, és több tudományos ágazatban használják. A cél mindenhol ugyanaz, az adatmennyiség csökkentése. Ennek eredményeképpen gyorsabbá válik az adatok interpretálása, kevesebb tárhely szükséges, és könnyebbé válik az adathalmaz szerkesztése.

A hálózatoptimalizálás fő problémája a sűrű hálókban – többnyire háromszöghálókban – található felületek számának csökkentése. Ehhez egy olyan módszer fejlesztettek ki, ami interpolációval alkotja meg a különböző részletességi szintekhez tartozó modelleket. Ezek a módszerek viszont görbe felületekre lettek kifejlesztve, és túl bonyolult algoritmusokat használnak a webes térképeknél való használathoz.

A hálózatoptimalizálásnak két nagyon fontos felhasználási területe van, a felszín rekonstrukciója rendezetlen pontokból és a hálózatok egyszerűsítése. Az egyszerűsítés alapja, hogy optimalizációs folyamatnak tekinti az algoritmust és bevezetésre kerül egy E erőfüggvény, ami közvetlenül megadja, hogy a generalizált hálózat mennyire tér el az eredetitől. A megfelelő minőség érdekében E értékét minimalizálni kell (Hoppe et al 1993).

A két korábbi módszert ötvözve alkotott meg Burghardt és Cecconi egy hálózatoptimalizálási módszert, amiben az E egyenlet kiszámítása kevéssé időigényes. A megoldandó problémát következőképpen fogalmazták meg:

„Vegyünk X eleme $R^2$ adatpont gyűjteményt és $M_0$ kezdeti háromszögelési hálót, és keressük $M_f$ hálót kevesebb metszésponttal, ami megfelelően illeszkedik az eredeti adatokhoz.”



*13. ábra: Metszéspontok számának csökkentése egy térhálóban*

Az illusztrációban három hálózat látható, amelyik megfelelő minőségben és olvashatósággal reprezentálja az eredeti alakot, jelen esetben egy fejet. A baloldali ábrán az eredeti adatokból álló háromszög látható, a középső egy egyszerűsített változat, ami még jól megtartja a kezdeti körvonalakat, és jellemzőket. A jobb oldalon látható fej már erősen generalizált, az eredeti töréspontok töredékével. Ezekből is látható, mennyire fontos a megtartandó pontok kiválasztása, azok elhelyezkedése, hogy jellemzően leírják az eredeti adathalmazt. Ha ezt az arcot a térkép generalizálás témájával párhuzamosítjuk, tekinthetünk rájuk úgy, mint különböző méretarányban történő ábrázolások. Elképzelhető, hogy az új csomópontok nem egy eredeti pontot reprezentálnak, hanem például több elem tömegközéppontját.

Vegyünk egy $M_0$ hálót, ami n csomópontból áll – minden csomópont az épület tömegközéppontját mutatja –, és keresünk egy $M_j$ (n-1) csomópontból álló hálót. Ennek a lehető legjobban kell reprezentálnia az eredeti rácsot, kis változtatásokkal. Hoppe et al. (1993) az M rácsot (K,V) párként írták le, ahol K egy szimpliciális komplexum. ami leírja a csomópontok, élek és felületek kapcsolatait, $V=\{v_1, \ldots, v_m\}$ $v_j \in R^3$ pedig a csomópontok helyzetének gyűjteménye, ami leírja a hálót. Ahhoz, hogy egy olyan hálót kapjunk, ami megfelelően reprezentálja az eredetit, a következő E függvény végrehajtása szükséges:

$$E(K,V) = E_{dist}(K,V) + E_{rep}(K) + E_{spring}(K,V)$$

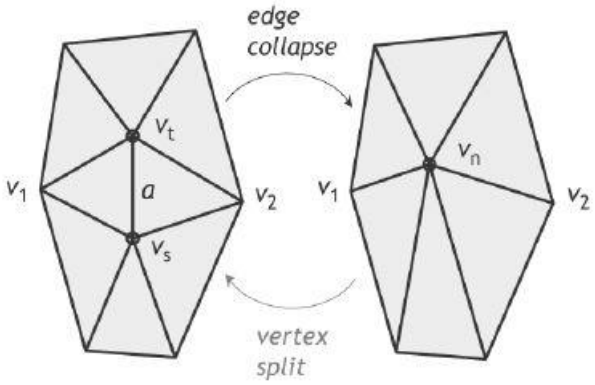
A cél a függvény eredményen minimalizálása, a különböző változók használatával. $E_{dist}$ a távolságok négyzeteinek össze, $E_{rep}$ egyenesen arányos a csomópontok számával. $E_{spring}$ pedig

egy szabályozó kifejezés, ami az élhosszakat írja le. A függvény használata elméletben nagyon sok lehetőséget ad a generalizálás folyamatának szabályozására, a távolságok, vagy irányok beállításával. Mivel ez a számítás rendkívül időigényes, és egy szabály, az elemek közötti távolság figyelembevétele is elég, egy egyszerűbb függvényt vezettek be az egyszerűsítéshez.

A tipizálás két fő lépésből áll, amik nem függetlenek és kommunikálnak egymással:

- pozíció: az új elemek számának és helyzetének meghatározása a célméretarány figyelembevételével
- ábrázolás: a meghatározott helyekre új épületek létrehozása

A tipizálás – vagyis a hálózategyszerűsítés – egy iteratív folyamat, ahol a kilépési értéket az előre meghatározott $m_r$ célméretarány adja meg. Az érték az elemek új száma, amit a kezdeti elemszám és a méretarány függvényéből számolunk ki. A két, egymás mellett fekvő objektum helyettesítésére szolgáló új elemet „helyőrzőnek" nevezzük. Az „edge collapsing" algoritmus használata egyszerű, és rendkívül eredményes hálók egyszerűsítésénél. Két szomszédos pontot von össze egy új helyzetbe, a 6. ábrán látható módon.




*14. ábra: Az „edge collapsing" algoritmus működése*

Az összevonandó pontok kiválasztása a köztük levő távolságtól függ, mindig a hálóban egymáshoz legközelebb fekvő pontpár kerül összevonásra. Az így létrejövő vertex az él közepének tekinthető, és a két eredeti pont között fekszik. Ez a változtatás csupán lokális hatású, a háló egészének struktúrájára nincsen hatással. Ha kettőnél több pont helyettesítésére van szükség, az összes érintett vertex tulajdonságait figyelembe kell venni, így az új helyzet az eredeti vertexek tömegközéppontja lesz.

Ahhoz, hogy kiszámolhassuk, hány csomópontból fog állni a generalizálás után létrejött háló, többféle számolási módszert használhatunk. Az egyik, hogy megtartjuk a térképen az épületek összterületének, és a háttérnek az arányát, vagy használjuk a Töpfer-féle gyökszabályt (7. ábra).

$$n_f = n_a \sqrt{\frac{M_a}{M_f}}$$

$n_f$ = Objektumok száma a levezetett méretarányban
$n_a$ = Objektumok száma az eredeti méretarányban
$M_f$ = Levezetett méretarány
$M_a$ = Eredeti méretarány

*15. ábra: Töpfer-féle gyökszabály*

Az egyszerűsítés végrehajtása után minden csomópont egy vagy több korábbi pontot képvisel. Nem csak a számuk, de a helyzetük is fontos, hiszen minden helyőrző pontnak a lehető legjobban kell képviselnie az eredeti elemeket.

A legkisebb távolság használata ahhoz a problémához vezethet, hogy a sűrűbb területek erősebben generalizáltabbá válhatnak. Ez előnyös lehet, hogy a sűrűn beépített területeken se legyen túl sok elem, de sajnos a terület eredeti arculatát jelentősen torzítaná. Ahhoz, hogy ez ne történjen meg, egy $f_a$ korrekciós együtthatót is bevezetett Burghardt és Cecconi a távolság kiszámításába. Ez megnyújtja az eredeti élek hosszát, hogy elkerülje az erős ritkulást az adott területen és a reprezentált elemek számától függ.

$$f_a = s_u * (r - 1) + 1$$

ahol $s_u$ a korrekciót, $r$ a távolságot jelenti.

Az algoritmus végrehajtása volt a tipizálás első lépése, a következő a megjelenítés, hiszen minden helyőrző vertex-nek tudnia kell azt, hogy melyik eredeti pontokat képviseli. Az eredeti pontok geometriai tulajdonságai alapján az kell az új épület formáját létrehozni az új méretarányban. A két tulajdonság az A alapterület és az α irány, ami jól leírja az elemet. Olyan új formát kell létrehozni, ami megfelelően utánozza a legnagyobb reprezentált elemet, de emellett a kisebbeket is figyelembe veszi.

A jelentősen egyszerűsített térképeken az épületek alaprajza többnyire téglalapot vesz fel. A módszer célja is az, hogy az új elemeket téglalapként ábrázolja, így a területet a vonatkozási pontok eredeti területéből kell kiszámolni. Ahhoz, hogy a minimális térképi méretet megtartsuk, $f_a$ arányt is szükséges bevezetni a számításba.

Az arány első fele kompenzálja a méretarány-változás közben fellépő méretváltozást, míg a második rész nélkül megtartanák a kezdeti alapterületeket.

Az így kiszámolt, és javított értékből kiszámolható az új elem szélessége és hossza, de a két oldal arányának meg kell egyeznie a helyettesített legnagyobb elem oldalhosszainak arányával.

A helyőrző irányának kiválasztásához a legnagyobb eredeti elem elhelyezkedését kell figyelembe venni. Ennek oka, hogy ez az elem képviseli legjobban a helyettesített csoportot, és a környezetének tulajdonságait sokkal erősebben meghatározza. Ennek okán a legnagyobb elem és a helyőrző α irányát egyenlőnek tekintjük.

A módszer könnyen szabályozható, és egyszerűen használható épületek számának csökkentésére, de van néhány hiányossága. A bemutatott verziója az épületsorokat nem tudja megtartani, ennek vizsgálatára egy további szabály bevezetésére lenne szükség. Ez egy feldolgozó lépés eredményként jöhetne létre, ami felismeri az épületek térbeli helyzetének jellemzőit. Mivel a helyjelző vertexek helyei csakis a korábbi objektumok tömegközéppontjaitól függenek, a különleges formájú épületek megtartása, formájuk reprezentálása előfordulhat, hogy nem megfelelő minőségű. Ehhez egy újabb változó bevezethető, mint például az épületek fontossága. A helyjelzők alapterülete a helyettesített épületek alapterületétől függ, azok átlagából kerül kiszámításra. Így a több, rendkívül eltérő méretű épület esetén azok méretének reprezentálása jelentősen torzulhat.

Emellett a minimális térképi távolság megtartását nem veszi közvetlenül figyelembe az algoritmus, így előfordulhat az elemek túlzott torlódása, azok fedésbe kerülése.

## Eltolás a váz és rugalmas sugárelmélet alapján (Liu et al. 2014)

A módszer célja, hogy olyan megközelítést dolgozzon ki, ami felismeri és megoldja azokat a közelségi problémákat épületek és utak között, amik a térkép méretarányának csökkenése miatt adódnak. Mielőtt az eltolás folyamatát végrehajtaná, az adathalmazt fel kell bontani több kisebb részletre. Ezeken a részeken belüli objektumok semmilyen térképi konfliktusban nincsenek más részletbe tartozó épületekkel, sem az utakkal. A felosztás alapjául általában az út- vagy vízhálózat szolgál. Ehhez először minden épületre létrehozunk egy r távolságú pufferzónát (r többnyire a térbeli pontosság négyszerese), majd egy egymást fedő puffereket egy nagy poligonná olvasztjuk össze. Ezután megvizsgáljuk az utak fedését is, és ez

alapján újabb, kisebb poligonokat hozunk létre majd pedig minden poligon centroidját megvizsgáljuk.

A felosztás után elfajuló Delaunay háromszögelést (Chew, 1989, Iványi–Radó 2015) hajtunk végre az adathalmazon, és ebből egy váz kerül létrehozásra. Az elfajuló háromszögelés esetén lesz az adathalmazban legalább egy tartomány, ami nem háromszög. A szabad térképi területet háromszögek sorozatára osztja fel, úgy hogy a hosszabb vonalakba plusz pontok is kerültek, tovább finomítva a háromszöghálót. A háromszögek tengelyeiből egy vázat készítünk, ami még nem végleges. További struktúrák bevezetése szükséges ahhoz, hogy a térkép területi jellemzőit, kapcsolatait jól megtartsa a modell végrehajtása közben. Az épületek poligonjait egy-egy node jelenti (az épületek centroidja), míg az élek a közöttük levő kapcsolatot jelzik egy gráfon belül. Azokat az épületcsoportokat, amiken belül nem volt konfliktus, azokat a modell egy node-ként kezeli, nem hajtja végre rajtuk az áthelyezést.

Az eltolás a szomszédsági konfliktusok eredményeként hajtandó végre, ami miatt nem megfelelő a térképi objektumok közötti távolság. A korábban létrejött váz szomszédsági viszonyai megerősítik, hogy fennáll-e a probléma. A térképi elemek méretét figyelembe véve meg kell adni egy minimális térképi távolságot:

$$d_{min} = d_c + \frac{1}{2}(r_1 + r_2)$$

ahol $d_c$ a minimális térképi térköz, $r_1$ és $r_2$ az elemek nagysága.

Ha az elemek közötti távolság kisebb ennél a határértéknél, konfliktus van az elemek között, aminek $s_l$ súlyossága következőképpen adható meg:

$$s_l = \max[0, (d_{min} - D)$$

ahol D a két elem közötti legkisebb távolság.

Ezek a térképi konfliktusok „erőket” ébresztenek az elemek között, amik egymásra hatnak és így előidézik az eltolási folyamatot. Kétféle konfliktus ébredhet, két épület, illetve egy épület és egy utcaszakasz között. A két épület közötti ütközés során két erő ébred:

$$\overline{f_b} = \frac{A_j}{A_b + A_j} s_l \frac{\overrightarrow{P_j P_b}}{|\overrightarrow{P_j P_b}|} \qquad \overline{f_j} = \frac{A_b}{A_j + A_b} s_l \frac{\overrightarrow{P_b P_j}}{|\overrightarrow{P_b P_j}|}$$

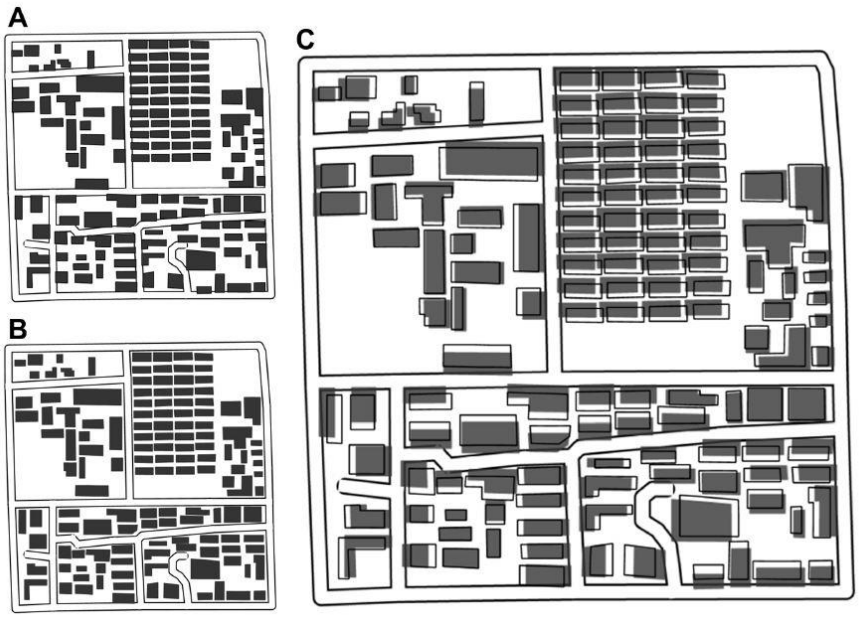
ahol $A_j$ és $A_b$ a két poligon területe, $P_j$ és $P_b$ pedig a poligonok legközelebbi pontjai.

Az épület és utca között fennálló konfliktusban ébredő erő az épületre hat, hiszen az utca nem mozgatható. Az erő a következő:

$$\overline{f_B} = s_l \frac{\overrightarrow{P_L P_B}}{|\overrightarrow{P_L P_B}|}$$

ahol $P_L$ és $P_B$ azok a pontok, amik a legkisebb távolságra vannak egymástól az útszakaszon és az épületpoligonon.

Ezek az erők idézik elő a poligonok eltolását. Bevezetendő egy helyzeti pontosság, ami alapján vizsgáljuk, hogy az áthelyezés meghaladta-e a szükséges pontosságot. Ez alapján megadható egy súlyosság, amiből kiindulva egy húzóerő ébred. Ez az erő hivatott visszahúzni a poligont, hogy a helyzeti pontossága ne haladja meg annak határértékét.

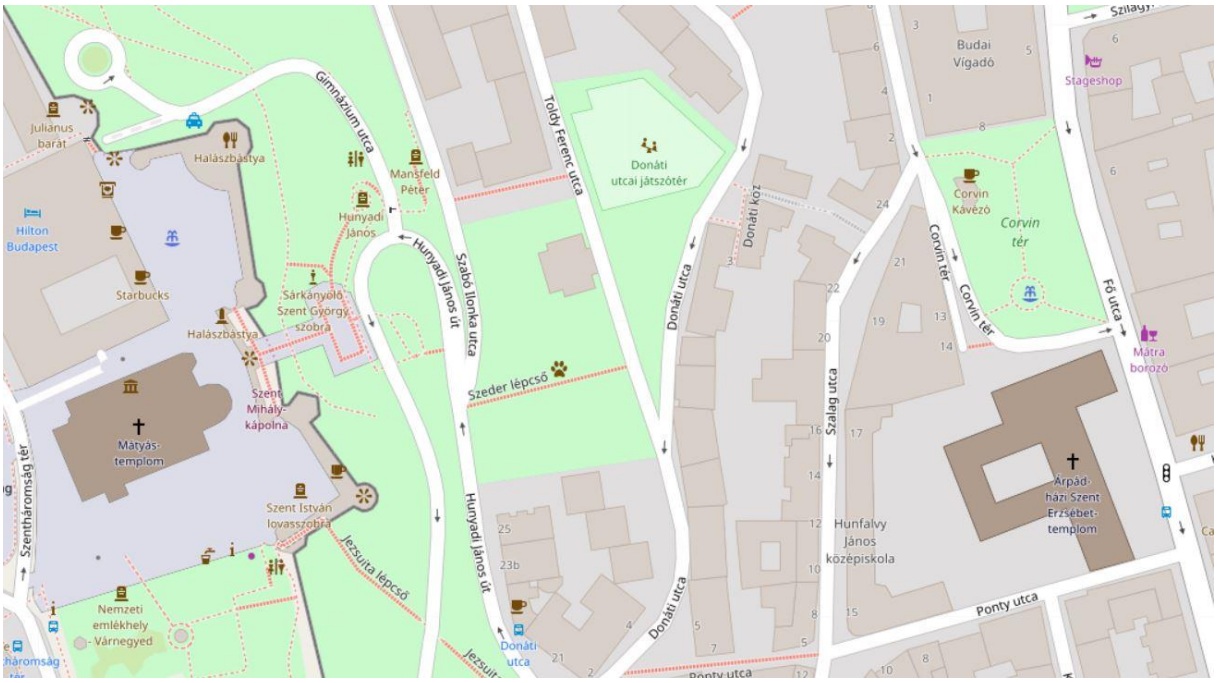



*16. ábra: A sugárelmélet alkalmazásának eredménye*

Ha az erőket nem egy-egy poligon, hanem több között vizsgáljuk, azok eredőjét kell venni, és kiszámolni az új helyzeteket. Az erők egyenkénti kiszámítása ahhoz vezethet, hogy az eltolás mértéke nem egységes az egész adathalmazon belül, és sérülhet az eredeti struktúra. A módszer nevét adó sugár a vizsgált poligonok között fut. Ennek hosszát is figyelembe véve számolható ki az elemek új helyzete egy mátrix-egyenlet végrehajtásával. A sugárelméletre egy algoritmust építettek, melynek célja az olvashatóság növelése volt úgy, hogy a kezdetben megadott helyzeti pontosságnak megfeleljenek az új elemek. A 16. ábrán látható az algoritmus eredménye. Az A ábrán az eredeti poligonok, a B ábrán pedig az eredmény látható. A C ábrán a szürke poligonok jelölik az eredeti pozíciókat, míg a körvonalak az eltolás utáni helyzeteket.

# 4. Épületgeneralizálás a térinformatikai programokban

Különböző térinformatikai programok generalizáló moduljainak teszteléséhez négy, különleges alakú épületet választottam ki. Mindegyik más-más szempontból különleges, így máshogy jelent kihívást a generalizáló algoritmusoknak. A választott épületek: Mátyás-templom, Vízivárosi Árpád-házi Szent Erzsébet-templom, Hunfalvy János középiskola és a Halászbástya (17. ábra). Ezek többnyire megjelennek különböző turista-kiadványokban, láthatók az EOTR szelvényen, az 1:25 000 és 1:50 000 méretarányú Gauss-Krügeren, illetve kataszteri térképen is. Így nem csak az algoritmusok összehasonlítására van lehetőségem, hanem a különböző méretarányú és kiadású térképeken látható generalizált változatokkal is összevethetem.




*17. ábra: Budapest I. kerülete az általam vizsgált épületekkel*

Az alapadatom forrása az OSM – OpenStreetMap– szabadon felhasználható internetes térkép adatbázisa. OSM-ről az Overpass Turbo nevű internetes adathalász alkalmazással vagy a QGIS-ben található OpenStreetMap menüvel lehet letölteni (Openstreetmap Wiki).

Az OpenStreetMap menü közvetlenül kapcsolódik a szerverhez, és egy tetszőleges kiterjedésű területről lehetséges az adatletöltés. Ezt megadhatjuk manuálisan földrajzi koordinátákkal, használhatjuk a meglevő adatsorunk vagy "térképvásznunk" kiterjedését. Ezzel minden, a területen található objektum letöltődik egy OSM fájlba. Ezt utána manuálisan kell a

projekthez hozzáadni. Ahogy a 18. ábrán is látható, a terület minden egyes eleme megtalálható ebben a fájlban.

*18. ábra: A térképterület minden eleme shape rétegekben*

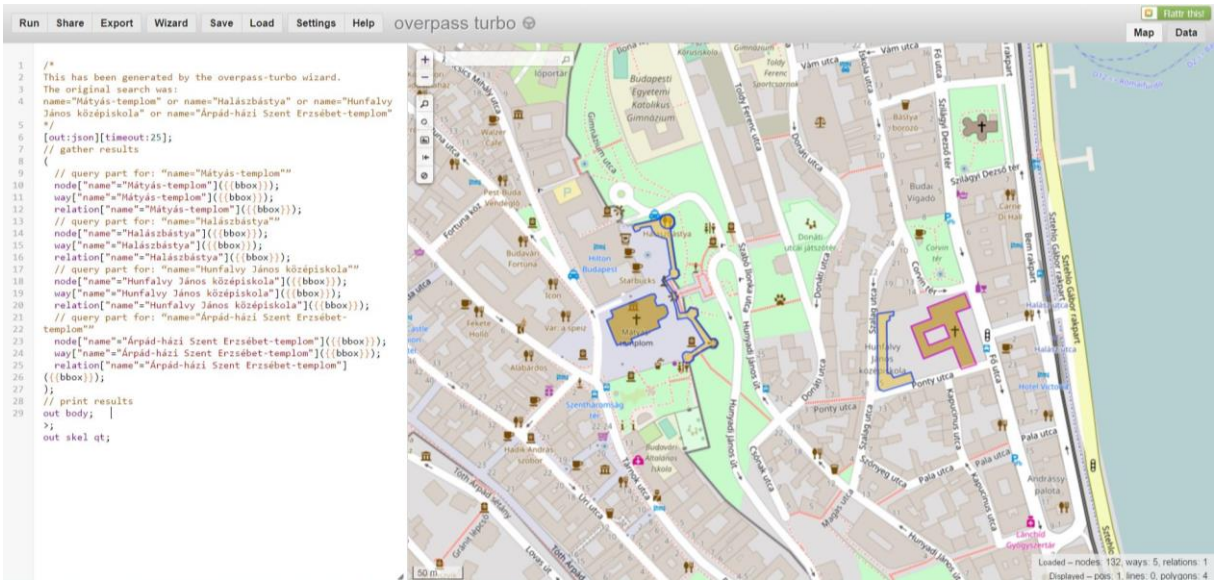
Mivel a programok tesztelése során a négy épületre koncentráltam, és a nagyobb fájl az algoritmusok futását is jelentősen lelassítja, mindenképpen a négy épületet tartalmazó adatsor használata előnyös. A szükséges elemeket ehhez egy új shape fájlba kell menteni. A szerkesztés bekapcsolása után az objektumokat egy új shape rétegre másoljuk majd elmentjük.

Az Overpass Turbo használatával viszont letölthetőek egyedülálló objektumok is, nem csak területek. A menüsorában található Wizard varázslóval különböző lekérdezések építhetők. Az elemek bármelyik attribútuma alapján lehet keresni, és bármekkora területen. A lekérdezések nagyon egyszerűek, csak az attribútum típusára és a keresett adatra van szükségünk. Én a lekérdezéshez az épületek nevét használtam. Ez nem mindig ad pontos eredményt, hiszen az adatbázisban eltérő lehet a nevük. Árpád-házi Szent Erzsébet-templom például több is található Budapesten, így vagy a keresési terület szűkítésével (én ránagyítottam az I. kerületre) vagy a lekérdezés bővítésével pontosíthatjuk az eredményt.

Az általam használt lekérdezés a következő volt:

*name="Mátyás-templom" or name="Halászbástya" or name="Hunfalvy János középiskola" or name="Árpád-házi Szent Erzsébet-templom".*

A több, különálló tagból álló neveket (vagy bármiféle tulajdonságot) mindig idézőjelek közé kell tenni, anélkül sajnos nem fut le helyesen a keresés. Az eredmény az alább található képen látható.




*19. ábra: A lekérdezésem eredménye*

A felhasznált adatsorom vetületének az EOV-t választottam. Mivel szögtartó, ezért nagyon jól megfigyelhető rajta, hogy a végeredmény elemeinél megmaradt-e a derékszög, ami a kevésbé összetett épületeknél alapvető pontossági követelmény. Az adatsorban ez leginkább a Szent-Erzsébet templomon figyelhető meg.

## QGIS

A QGIS egy nyílt forráskódú, mindenki számára hozzáférhető, ingyenes térinformatikai program, rendkívül széles spektrumú lehetőségekkel. Jelenleg a program 2.18.0 verziószámú kiadását használom. Alapvetően nincsenek generalizálásra használható lehetőségek a programban, de rendelkezésre áll egy plugin tárház, ahonnan bármilyen, számunkra fontos modul letölthető.

Jelenleg négy generalizálásra kifejlesztett plugin adható hozzá a programhoz: Generalizer, Advanced geometry simplification (SimpliPy), Cartographic Line Generalization és Simplify poligons. A négy modulból egy, a Generalizer csak vonalak egyszerűsítésére alkalmazható, így épületek esetében nem használhatók, ennek használatát nem mutatom be. Azonban egyik plugin lehetőségei között sincs jelenleg kizárólag épületek egyszerűsítésére szolgáló lehetőség, ezért a leggyakrabban használt poligonegyszerűsítő módszereit alkalmaztam.

## SimpliPy

A plugin két algoritmust, a Douglas–Peucker- és a Visvalingam-algoritmusokat használja. A generalizálást végrehajthatjuk egy egész rétegen vagy akár csak néhány kiválasztott elemen is. Az elkészült adathalmazt egy rétegbe menti, amit tetszés szerint hozzáadhatunk a projektünkhöz. A rétegből nem generálódik automatikusan shape fájl, így ha az eredményt biztosan szeretnénk tárolni, akkor szükséges új fájlként elmenteni.

Néhány szabály is beállíthatunk, hogy a generalizálás minőségén javíthassunk. Automatikusan kijavítja az átfedéseket, metszéseket, ha kiválasztjuk a ’Repair intersections’ szabályt. A ’Prevent shape removal’ (tartsa meg a kis poligonokat is) kikötés rendkívül hasznos az épületek generalizálása során. Beállíthatjuk, hogy minimum hány vertex-ből álljanak az egyszerűsített elemek. Így lehetőség van arra, hogy minden épület négyszög maradjon, és így megfeleljen legalább az egyik kikötésnek a generalizálás után. A topológia megtartására is van lehetőség (Use topology), de mivel az épületeknél többnyire egymástól külön álló objektumokról van szó, ennek használata, vagy kikapcsolása nem jelentett az adatsoromban változást.

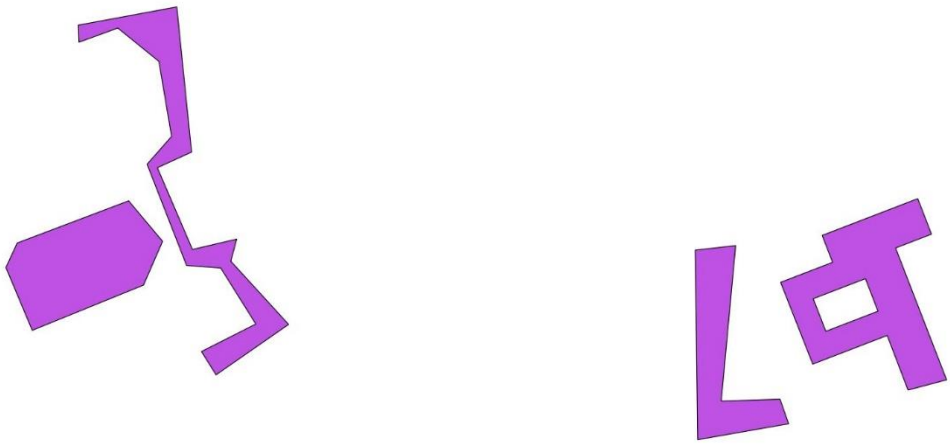
Kiválaszthatjuk, hogy az algoritmus hogyan ossza fel a poligont egy vonalláncra. Három lehetőség van: First and last, First and furthest és Diameter points. A First and last az első és az utolsó pont között választja szét, a First and furthest az első, és a tőle legmesszebb található pontnál, míg a Diameter points a poligon köré írható kör két átellenes pontjánál választja szét a poligont. Ennek a szabálynak a változtatása nem okozott semmilyen változás az algoritmus eredményében, így az alapbeállítással, a ’First and last’-tal dolgoztam.

A Douglas–Peucker az egyik legelterjedtebb vonalegyszerűsítő algoritmus, ami merőleges távolságon alapul. A megadott határérték ezt a távolságot vizsgálja. Ha az első és utolsó pont összekötésekor a legtávolabbi pont merőleges távolsága nagyobb, mint a határérték, a pontot az új vonalláncba felveszi az algoritmus. Az iteráció az új pontokat összekötve ismétlődik, egészen addig, amíg az összes szükséges pont belekerült a vonalláncba (Douglas és Peucker, 1973).

Természetéből adódóan az algoritmus először a kisebb éleket egyszerűsíti, minél nagyobb belső szöget zárnak be a szomszédos éllel, annál gyorsabban. A modul első megnyitáskor a 0,01 méteres tolerancia jelent meg automatikusan. Ekkora küszöbértéknél még semmilyen változás nem történt, így növeltem azt. Az első érték, ami változást hozott, 0,5 méter

volt. Ez először a legkisebb éleket kezdte eltávolítani, így a Halászbástya tornyain, valamint a középiskola bejárati oldalánál. Itt fontos megjegyezni, hogy az iskola OSM-en található poligonja sajnos hibás. A bejárat frontja a valóságban egy egyenes fal, ami a két utcafronti fal által bezárt sarok „levágásával” keletkezett, míg az OpenStreetMap-en a bejárati oldal 3 élből áll, kicsit lekerekítve a sarkokat. Először a kisebb élek eltávolítása történt meg, egészen az 1 méteres küszöbérték használatáig. Itt egy olyan él került eltávolításra a Mátyás templom rajzolatából, ami az egyik főbb oldal elmozdulását jelentette. Innentől kezdve a tolerancia értéket fél méterrel növelve futtattam le az adatsoron.

Mivel ezek az értékek elég kicsik, továbbra is csak a kis élek eltávolítása figyelhető meg, leginkább a Halászbástya és a Mátyás templom rajzolatában. Két méteres határértéknél a bástyák már kezdenek eltűnni, helyette vastagabb falszakaszok jelennek meg, így eltorzítva a formát. A szakközépiskola formája ebben a változatban közelíti rendkívül jól az eredeti rajzolatot, az utcákkal párhuzamos falak még nem mozdultak el. A 3,5 méteres értéknél már az összes lekerekített falfelület eltűnt a generalizálás eredményeképpen. Az iskola bejárati oldala is eltávolításra került, így megszűnt az utcákkal való párhuzamossága. Ennek ellenére még rendkívül jól közelíti az eredeti rajzolatot, ennél a küszöbértéknél nagyobb használata nem lenne az épületre szükséges. A Mátyás templom boltozatos oltárrésze, és a Halászbástya minden bástyája egy háromszöggé redukálódott. A 4 méteres érték eltávolítja a Mátyás-templom déli oldalán található kiugrást, ezzel megerősítve a derékszögű sarkakat, illetve a bástya közepén található kiugró részt szintén a másik oldallal párhuzamos vonallá egyszerűsíti. Ennél nagyobb toleranciaérték használata erre a három épületre már rendkívül nagy torzulás okoz, ami sok manuális javítás után lenne csak értékelhető minőségű. Ezzel szemben, a csak derékszögű sarkokból álló Szent Erzsébet-templom generalizálása nem indult el.



*20. ábra: A Douglas–Peucker algoritmus eredménye 6 méteres küszöbértékkel*

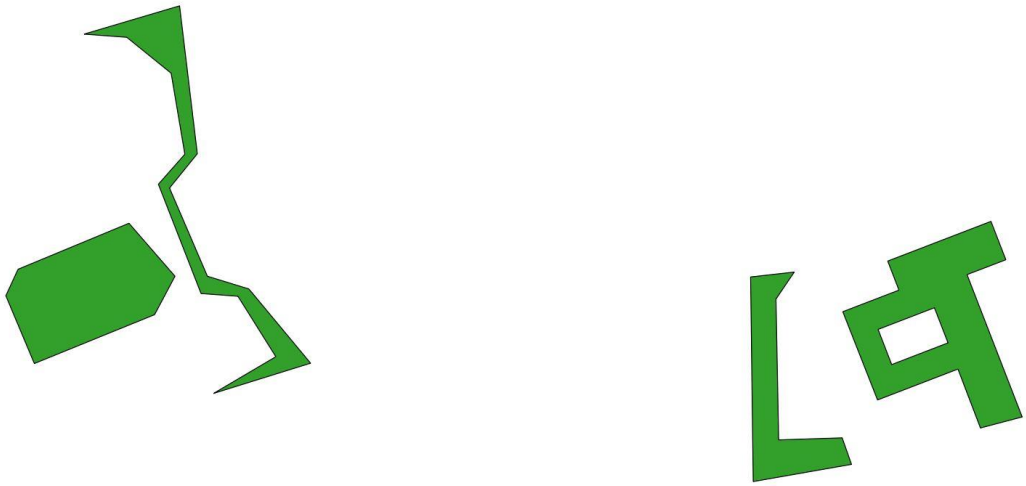
Ahogy a 22. ábrán is látható, 6 méteres küszöbértéknél az iskola egy L alakú poligonná válik, ami az eredeti oldalakhoz való igazítás után megfelelő lenne az eredeti forma helyettesítésére a kisebb méretarányokban. A Mátyás-templom egy hatszöggé egyszerűsödött, ami már nem hasonlít a kiindulási anyagunkra. A Halászbástya az egyszerűsödés eredményképpen egyre nagyobb és nagyobb alapterületűvé válik, ahogy a bástyákat összekötő épületszakaszok folyamatosan vastagodnak. Innen egészen a 8,5 méteres értékig nem történik változás az adatsorban. Az iskola L alakjának egyik sarka eltűnik, a Halászbástya déli fele pedig teljesen eltorzult. 10,5-ös küszöbértéknél a Halászbástya területe újra elkezd csökkenni, de a fő futási vonala már teljesen eltorzult. 11 méternél a Mátyás-templom egy paralelogrammává alakult, elérte a maximális generalizálási értéket, hiszen korábban kikötöttük, hogy az épületeknek minimum négy oldallal kell rendelkezniük. Elindult a Szent Erzsébet-templom egyszerűsödése is, az északkeleti hajó négyszögből háromszöggé változott.

Ezután ismét megállt az egyszerűsödés mértéke, egészen a 15 méteres küszöbértékig. A Szent Erzsébet-templom tovább egyszerűsödött, az alapvető formáját teljesen eltorzítva. A Halászbástya ismét felvette a kezdeti vékony falakat követő poligonformát. A 20 méteres értéktől kezdve már egyik épületnél sem volt megfelelő a generalizálás, nagymértékű torzulás miatt értékelhetetlen, és manuálisan nehezen javítható eredményt hozott. Az algoritmus futtatásának eredményei további küszöbértékek használatával az 1. mellékletben találhatók.

A Visvalingam-algoritmus azokat a pontokat távolítja el a poligonból, amelyek eltávolítása a legkevesebb területvesztéssel jár. A futtatása során megadunk egy küszöbértéket, ami azt a területnagyságot jelöli, aminél kisebb területű háromszögeket kitöröljük. A háromszög a vizsgált pont, és a pontot megelőző, valamint utána következő pont által kerül kijelölésre. Természetesen kezdetben a nulla területtel rendelkező pontok kerülnek törlésre. (Visvalingam és Whyatt, 1993)

Az előző módszerhez hasonlóan ez is a kört közelítő részek generalizálásával kezdte az egyszerűsítést a kisebb küszöbértékek használatakor. Legelőször a Mátyás-templom beugró sarkának részletei, illetve az iskola bejárati oldala egyszerűsödött. 5 $m^2$-es toleranciaértéknél már a Mátyás-templom szentélye és a Halászbástya bástyája elkezdett egyszerűsödni. Mivel itt nem távolságok, hanem területek vizsgálata adja az egyszerűsítés alapját, jóval lassabbak a változások a küszöbérték növelésekor. A Douglas–Peucker algoritmus használatakor 5 tizeddel való növelés hozta azt az egyszerűsítési szintet, mint ennél az algoritmusnál 5-tel való növelés. Ezzel szemben az adatsoromon az az előnye, hogy a kiugró részeket egyszerűsíti először, és a

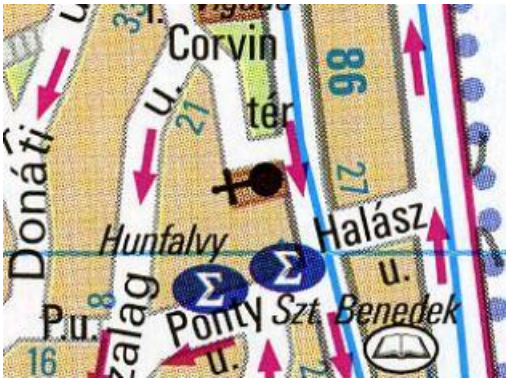
fő formát megtartja. Ez főleg a Halászbástya esetében lehet előnyös, hiszen többnyire a bástyák nélkül kerül ábrázolásra különböző térképeken. A Mátyás-templom kiugró oldalhajója is kisebb küszöbértéknél kerül eltávolításra ennél az algoritmusnál, mint ahogy a szentély kört közelítő formája teljesen eltorzulna. Ez a 35 $m^2$-es érték, ahol az iskola formájában található lépcsős falak is eltávolításra kerültek. Ez egészen 150 $m^2$-es küszöbértékig nem változik, eddig Mátyás-templomon valamint a Halászbástyán is csak a kisebb részletek eltávolítása folytatódik. Ennél az értéknél viszont egyszerűsödik az iskola bejárati oldala is, megszűnik az utcákkal való párhuzamossága. Ez az eredmény használható lenne a kisebb méretarányokban való megjelenítéshez, az utcákhoz való igazítás, és a párhuzamosság javítása után. A Mátyás-templom poligonja egy hatszög, ami már nem reprezentálja megfelelően az eredeti formát. A Halászbástya eredménye viszont nagyon jól közelíti a térképen általában látható formát, ahogy a bástyák nélkül láthatjuk. Némi javítás szükséges lenne, a háromszög formájú végződéseket, és a nem párhuzamos falakat kell manuálisan korrigálni.



*21. ábra: Visvalingam algoritmus eredménye 150 m2-es toleranciaértékkel*

A változás az eredmények között ismét lelassult, így a küszöbértéket 50 $m^2$-rel növeltem. 250 $m^2$ toleranciaértéknél elkezdődött a Halászbástya egyes falrészeinek eltávolítása. Az iskola egyik szárnya háromszöggé egyszerűsödött, a Mátyás-templom pedig ötszöggé. Ezzel szemben, az eddig nem említett Szent Erzsébet-templom egyszerűsödése ennél az értéknél kezdődött el. Először az északnyugati szárny négyszöge egyszerűsödött háromszöggé, vagy 400 $m^2$-nél a déli is. A Halászbástya 350 $m^2$ toleranciaértéknél elérte a maximális egyszerűsödést, egy háromszög alakját vette fel. 400-as értéknél a Mátyás-templom négyszög alakot vett fel. A Szent Erzsébet-templomnak először az északkeleti hajója egyszerűsödik. 500 $m^2$-es értéknél csak egy háromszöggé válik, 750-es toleranciánál pedig teljesen eltűnik. Ezzel az algoritmussal elég jól egyszerűsíthető ez a forma is, mivel a két kiugró szárnya kerül

eltávolításra, a központi négyszög marad meg. A térképeken általában téglalapként láthatjuk, amelyről néhány esetben a belső udvar ábrázolása is elmarad.



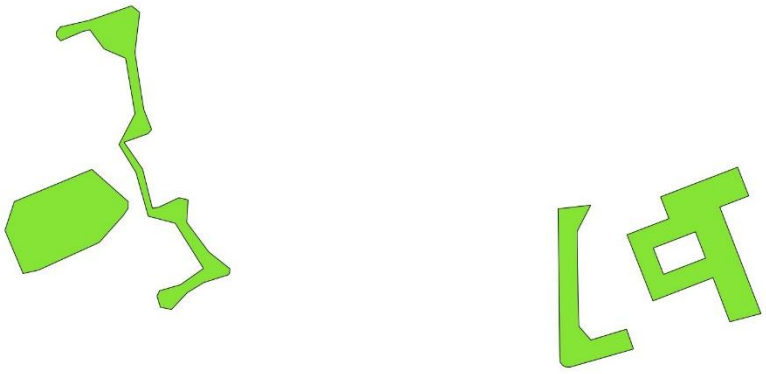

*22. ábra: A Szent Erzsébet-templom várostérképen*

Az algoritmus futtatásának további eredményei a 2. mellékletben láthatók.

## Cartographic Line Generalization

A plugin generalizálja a vonalakat nyomtatott vagy képernyőn megjelenített térképen megfelelő ábrázoláshoz. A generalizálás mértéke térképi méretarány kiválasztása alapján történik, hogy egyszerű legyen a használata. Minél kisebb méretarányt adunk meg (vagyis minél nagyobb számot) annál egyszerűbb poligon lesz az eredmény. Megőrzi az elemek alapterületét, nem engedi a poligonok vonallá vagy ponttá alakulását, viszont megőrzi az eredeti vonal fő futását. A topológiai hibákat, önmetszéseket, fedéseket nem tudja kezelni. (QGIS Python Plugins Repository)

A kézi generalizálás folyamatát utánozza, először egyszerűsíti a vonalat, majd pedig simítja, úgy hogy az eredeti területet közelítően megtartsa. A generalizálás paraméterei a különböző méretarányú, kézzel készítette térképek vizsgálatából származnak (Tutić, 2009).



*23. ábra: Az Egyszerűsítés algoritmus eredménye 1: 50 000 célméretarány beállításával*

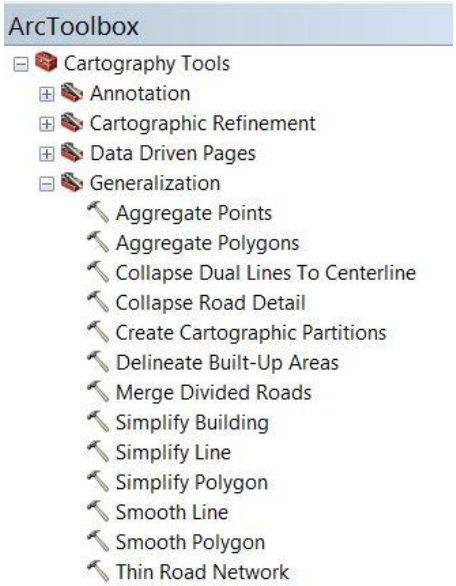
Négy különböző típusból választhatunk: Egyszerűsítés (Simplification), Egyszerűsítés+Simítás (Simplification+Smoothing), Simítás (Smoothing) és Derékszögű

szakaszok (Orthogonal segments). A Smoothing kivételével a másik három módszer eredményeit vizsgáltam az adatsoron. A generalizálás mértékeként egy tetszőleges méretarányt szükséges megadni, és a vizsgált térképek alapján tanult módszerrel egyszerűsíti az elemeket. Mind a simítással kiegészített, mind a sima egyszerűsítés törli a kiugró részeket, a lépcsős sarkokat, és a bástyákat. A simítással kiegészített egyszerűsítés elveszi az eredeti épületjellemzőket, az egyenes szakaszokat, szögeket. A derékszögű szakaszokat megtartó módszer lassabban generalizál, megtartva a szögletességet. Kismértékű generalizálásra alkalmas ez a program, nagyobb változásokat nem tud megfelelően végrehajtani. További probléma pedig, hogy nagyobb számok megadása esetén nem hajtja végre a generalizálást, így az eredmények vizsgálata ennél a modulnál nem teljes körű. A legnagyobb érték, amit meg tudtam adni, 50000 volt, ami nem eredményezett jelentősebb változást az adatsorban (Ungvári, 2017).

## ArcMap

Az ESRI ArcGIS szoftvercsaládjának elsődleges térinformatikai szoftvere, ahol a geoinformatikai műveletek nagy tárházának végrehajtására van lehetőségünk. Az adathalmazunkat térképpé alakíthatjuk, elemezhetjük, és különböző adathalmazokba rendezhetjük. Munkám során az ArcMap 10.4.1-es kiadását használtam. Elemzésre alkalmas eszközei között többféle generalizálásra kifejlesztettet találhatunk. A *Generalization* eszköztárban többféle generalizálási módszerre is találunk példát, egyszerűsítésre, simításra valamint összevonásra is. Az épületek generalizáláshoz használható eszközök a következők:

- Aggregate Polygons
- Delineate Built-Up Areas
- Simplify Building
- Simplify Polygon.



*24. ábra: A Generalization eszköztár lehetőségei*

Az Aggregate Polygons eszköz használatával több poligon összevonható, a Delineate Built-Up Areas egy poligonnal helyettesíti az egyedülálló kisebb épületpoligonokat. Mivel csak az épületpoligonok egyszerűsítésének módszereit és minőségét vizsgálom, a Simplify Building és Simplify Polygon eszközöket fogom részletesebben bemutatni.

A Simplify Building speciálisan épületek poligonjainak generalizálására lett kifejlesztve, ezért összehasonlítom a poligonegyszerűsítő modul eredményeivel.

### Simplify Polygon

A poligonegyszerűsítő két különböző vonalegyszerűsítő algoritmus sokszögekre optimalizált változatával dolgozik. Eltávolítja a túlzott kiugrásokat, felesleges íveket a vonal futásából úgy, hogy megtartja az alapvető formáját. A két algoritmus a Point Remove és a Bend Simplify. A Minimum area paraméterrel beállítható, hogy a végeredmény poligonjainak mi legyen a minimális területe, az ennél kisebbeket eltávolítja. A topológia figyelembevételénél három lehetőség van: ne vizsgálja, jelölje a hibákat, vagy javítsa a hibákat. Az ellenőrzés nélkül a generalizálás folyamata gyorsabban zajlik, de az esetleges hibákat –önmagukat vagy egymást metsző poligonokat – manuálisan szükséges megkeresni. (ArcMap Help)

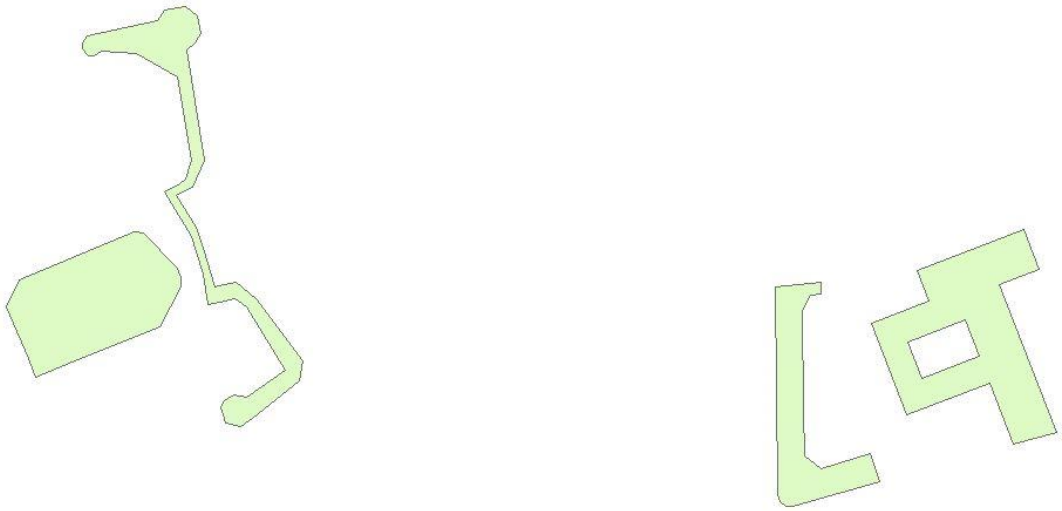
A Point Remove a Douglas–Peucker-algoritmust használja. A két lehetőség közül ez a gyorsabb, de kevésbé őrzi meg az eredeti formát. Célja inkább az adatmennyiség csökkentése, durva egyszerűsítés, mint minőségi generalizálás. Az algoritmust lefuttattam néhány küszöbértékkel, de pontosan ugyanazokat az eredményeket hozta, mint az elsőként említett, QGIS SimpliPy Douglas–Peucker algoritmusa, így ebben a részben nem részletezem az eredményeket.

A Bend Simplify a Wang–Müller algoritmus használatával hoz létre az eredeti vonalfutást jól követő eredményt. Lassabb, mint a Point Remove, kevésbe generalizál, de szebb eredménnyel. A kevésbé fontos íveket távolítja el, így tartja meg az eredeti formát. (ArcGIS Help)

Az algoritmusban a vonalalak ívek sorozata, amiket inflexiós szögek változása határoz meg. A forma és a méret az ívek jellemzője, így több típus különböztethető meg. Az algoritmus küszöbértéke egy félkörív minimális átmérője. Egyenként vizsgálja az íveket, és helyi minimumokat, vagyis olyan íveket választ ki, amik a környezőknél kisebbek, és ezeket a toleranciaértékhez hasonlítja. Ha nagyobb az ív, meghagyja, ha pedig kisebb a küszöbértéknél, egyszerűsíti. Az egyszerűsítésnek három módja van: törlés, összevonás és nagyítás. (Wang és Müller, 1998)

A Bend Simplify a többi algoritmushoz hasonlóan a bástyák generalizálását kezdi el először. Nagyon jól kiszűri a kisebb kiugró részeket a Mátyás-templom illetve az iskola poligonjában is, a kiugrásokat eltávolítja. 5 méteres küszöbértéktől kezdődik el a generalizálás,

kisebb érték nem hozott változást az adatsoron. Ezután fokozatosan csökkennek a részletek a bástyákon, méterenként növelve a toleranciát. A 10 méteres tolerancia eltávolítja a Mátyás templom szentélyének egészét, és a déli oldalon található kiugrást is. A kiugrás levágását leszámítva, egy konvex burkot hozott létre körülötte. A 25. ábrán látható, hogy 15 méteres küszöbértékkel végrehajtva hozta az eddigi legjobb eredményt a Halászbástya generalizálására, levágta a bástyákat, de a főbb vonalfutást továbbra is meghagyta. A szélső bástyák azonban itt nem egyszerűsödtek, ez az egyetlen hátránya. Nagyobb értékek használatakor ismét eltorzul a poligonja, 30 méteres tolerancia használata önmetsződést okoz. A többi három poligonon nem hoz nagy változást, egészen a 30 méteres érték használatáig, amikor az iskola északi sarka levágásra kerül. 35 méteres értéknél a Szent Erzsébet-templom egyik csomópontja törlődik. A Halászbástya már jelentősen eltorzult. Az 50 méteres küszöbérték már jelentősen eltorzítja az épületeket. Az iskola a háromszög formát kezdi közelíteni, a Halászbástya egy torz poligonná egyszerűsödött, a Mátyás-templom továbbra is konvex burokra hasonlít. A Szent Erzsébet-templom belső udvara törlésre került, emellett a külső körvonal újabb sarka egyszerűsödött.



*25. ábra: Bend Simplify eredménye 15 méteres toleranciaértékkel*

Ez az algoritmus kisebb küszöbértékkel való használat esetén nagyon jó eredmény hoz, hiszen pont azokat a kisebb részleteket távolítja el, amelyek csupán kiugrások a főbb formából. Így nem torzítja el az eredeti vonalfutást, nagyon jól közelíti azt. Ahogy viszont a Szent Erzsébet-templomon és az iskolán látható, nem törekedik az alapforma megtartására. A többi algoritmus a kiugró részeiket, kisebb oldalakat távolította el, míg ez fontosabb pontok törlésével eltorzította az eredeti formát. Az algoritmus eredményére további példák a 3. mellékletben láthatók.

## Simplify Building

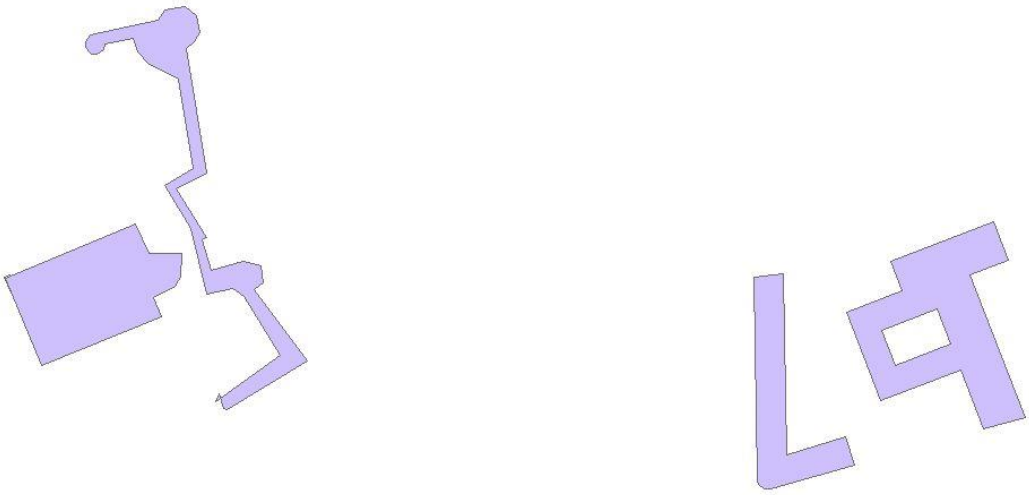
A Simplify Building algoritmusa kifejezetten különálló épületek egyszerűsítésére van optimalizálva. Figyelembe veszi a derékszögeket, azokat megtartja, szükség esetén pedig javítja. Felismeri, hogy az épületek különállóak, egyenes szomszédos vonalakkal rendelkeznek, vagy pedig bonyolultabb módokon kapcsolódnak egymáshoz. A különálló épületeket egyszerűsíti a bennük található belső oldalakkal együtt, és az egyszerűen egymáshoz kapcsolódó épületeket is generalizálja. A bonyolultabb, összetett épületcsoportokat nem tudja egyszerűsíteni (ArcMap Help).

A megadott küszöbérték az a minimális hossz, amit a falak felvehetnek. Az ennél rövidebb oldalakból álló kiugrásokat levágja, kiegyenesíti az oldalakat, és egyszerűsíti a poligont. Az épületet alkotó pontok száma csökken, de az algoritmus törekszik az eredeti alapterület közelítésére. Megadható egy minimális alapterület, az ennél kisebb épületeket, épületcsoportokat kihagyja az algoritmus. A legerősebben generalizált elemek téglalap alakot vesznek fel. Bár nem találtam meg, milyen algoritmus alapján működik, a folyamat rendkívül hasonló a minimumkereső eljárás (épületek alakjának generalizálása minimumkereső eljárással) folyamatához.

Az algoritmus problémája, hogy önmagukat metsző poligonokat is létrehoz. Nem javítja ezeket a problémákat, de ha kipipáljuk a *Check for spatial conflicts* szabályt, külön poligonokat hoz létre, egy önmagát metsző helyett.

A legkisebb toleranciaérték, amivel jól látható változások jöttek létre az 5 méter volt. Már itt látható, hogy ezt a módszert az épületekre optimalizálták, a kisebb kiugrásokat úgy egyszerűsíti, hogy az objektumok fő formája a lehető legkisebb mértékben változzon. A Mátyás-templom egyszerűsítése nem a szentély körívén kezdődött, hanem a kisebb kiugró részek eltávolításával. A Halászbástyának a bástyái kerültek egyszerűsítésre először, de a hosszabb falszakaszok meghosszabbításával vágódtak le a bástyák részei. Az iskolának szintén a kisebb kiugró falszakaszai kerültek először egyszerűsítésre. Ez tovább folytatódik nagyobb küszöbértékek használatakor, de a Mátyás-templom szentélyének eltorzulása már 10 méteres küszöbértéknél megtörténik. 11 méteres toleranciánál (26. ábra) már a Halászbástya elkezd önmagát metsző poligonná alakulni. Először a déli csücske, majd pedig 13 méteres küszöbértéknél egészen eltorzult, háromszor metsződő poligonná alakult. 15 méteres küszöbértéknél ez a nagyobb metsződés letűnik, de a poligon már erősen eltorzult, nem reprezentálja megfelelően a Halászbástya eredeti formáját. Itt látszódik legjobban, hogy az

algoritmus a párhuzamosításra, és a derékszögesítésre törekszik, de ezzel eltorzítja az objektumot. A Mátyás-templom szentélye tovább egyszerűsödik, az iskola poligonjáról minden kiugró részlet eltűnt. A Szent Erzsébet-templom északi fele eltorzul, nem a kiugró részek kerültek eltávolításra, hanem a többi hozzáigazítva.



*26. ábra: Simplify Building eredménye 11 méteres toleranciértékkel*

20 méteres toleranciánál a Mátyás-templom téglalappá egyszerűsödik, a Szent Erzsébet-templom északi kiugró hajója pedig eltűnik. Ez után jelentősebben csak az utóbbi változik, a másik három poligonban nem hoznak jelentősebb változást a nagyobb küszöbértékek. A 25 méteres tolerancia használatának eredményeképp egy nagy négyzet jött létre a templomból, melynek a belső udvarában semmilyen változás nem történt. A Mátyás-templom, és az iskola egyszerűsödése ennél a küszöbértéknél megáll, nem folytatódik tovább. Az iskola egyszerűsített rajzolata rendkívül jól képviseli az eredeti poligont, arról eltűntek a kisebb kiugrások, csak a főbb falak maradtak meg. A Halászbástya és a Szent Erzsébet-templom tovább változik. Míg az előbbi poligonja egyre torzultabb, és nagyobb lesz, az utóbbinak csak mérete változik, formája nem. Az 50 méteres küszöbérték jelentős változást hoz, a Halászbástya ismét felveszi a kezdeti alakzatot jól közelítő formáját, amiben néhány bástya már nem látható. Az 55 méteres értéknél már a Mátyás-templom sem téglalap formájú, megjelenik az eredetinél csak kevéssel egyszerűbb formája. Ezek nem egyeznek meg egyik korábban használt küszöbérték eredményével sem, de a Halászbástya esetében elfogadható eredményt ad. Az algoritmus futtatásának további eredményeiről ábrák a 4. mellékletben találhatók.

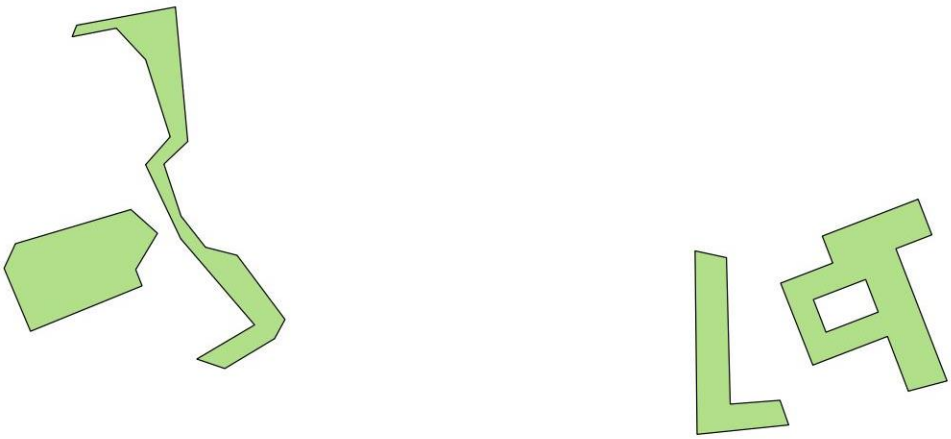
## GRASS GIS

A GRASS (Geographic Resources Analysis Support System) egy több mint harminc éves, ingyenes, nyílt forráskódú térinformatikai szoftvercsalád, ami térbeli adatok kezelésére és elemzésére, képfeldolgozásra, grafika– és térképszerkesztésre, térbeli modellezésre és megjelenítésre használható (GRASS GIS manual). Használata széles körben elterjedt. Amellett, hogy önálló térinformatikai programmal rendelkezik, beépülve megtalálható a QGIS-ben is.

A v.generalize nevű modulja több, vektoros generalizálásra alkalmas algoritmust tartalmaz. Található benne több vonalegyszerűsítő, simító algoritmus, eltolás és hálózat generalizálás. Sajnos, az eltolás csak vonalas elemekre alkalmazható, így csak az egyszerűsítő algoritmusok eredményeit vizsgálom. Négy különböző módszer közül választhatunk: Douglas–Peucker, Douglas–Peucker Reduction, Lang, Reumann–Witkam algoritmusok (GRASS GIS manual).

A Douglas–Peucker Reduction algoritmus az eredeti algoritmus továbbfejlesztése, aminél egy plusz *reduction* paraméter jelenik meg, ami megadja, hogy az új poligont hány csomópont alkossa, százalékban viszonyítva az eredeti mennyiséghez.

A Lang-algoritmus merőleges távolságok alapján vizsgálja a vonalakat. Megadandó egy *search region* (a modulban *look_ahead*) *paraméter*, amivel beállítható, hogy egyszerre hány pontot vizsgáljon. A kezdőpont és a paraméter által megadott pont összekötésre kerülnek, és ettől a vonaltól vizsgáljuk a többi vertex merőleges távolságát. A legtávolabbi pont, ha messzebb van, mint a beállított küszöbérték a kezdőponttá (*key point*) válik, és újra folytatódik az algoritmus futása. Ha kisebb, a *search region* területén található összes pont törlésre kerül. Ez addig folytatódik, míg az összes pont kulcsponttá nem válik. (Psimpl)



*27. ábra: Lang algoritmus eredménye 6 méteres küszöbértékkel, a „search region-t" 5-nek választva*

Az algoritmus futtatása során a *search region*-t 5-nek (27. ábra) választva kaptam a legjobb eredményeket. A Mátyás-templom és a Halászbástya nagyon eltorzul az algoritmus alkalmazása során, de az iskolát nagyon jól generalizálja. A Szent Erzsébet-templomot nagy küszöbérték választásakor sem egyszerűsítette. Azoknál a toleranciaértékeknél, amik már túl nagyok voltak az adott épületen való alkalmazáshoz, nem futtatta le az algoritmust, a végeredmény az eredeti poligonnal tökéletesen megegyezett.

A Reumann–Witkam algoritmus is a merőleges távolságok alapján működik. Itt a vizsgálati vonalat a kulcspont és a sorban következő jelöli ki, a merőleges távolságok ettől mérjük. Mindig a kulcspont utáni második pontot vizsgáljuk. Ha a pont közelebb található, mint a küszöbérték, töröljük az összes pontot a kulcspont és a vizsgált pont között, így a legnagyobb merőleges távolságú pont marad meg. Ha messzebb található a tesztpont, mint a toleranciaérték, hozzáadjuk a csomópontot a vonalhoz, majd újrakezdjük az algoritmus futtatását (Curve Simplification Turk Project).

Az algoritmus jelentősen eltorzította mind a négy épületpoligont, azok egyszerűsítésére nem alkalmas. A 28. ábrán látható eredményt egy nagyon kicsi, 2 méteresen küszöbérték használata eredményezte.

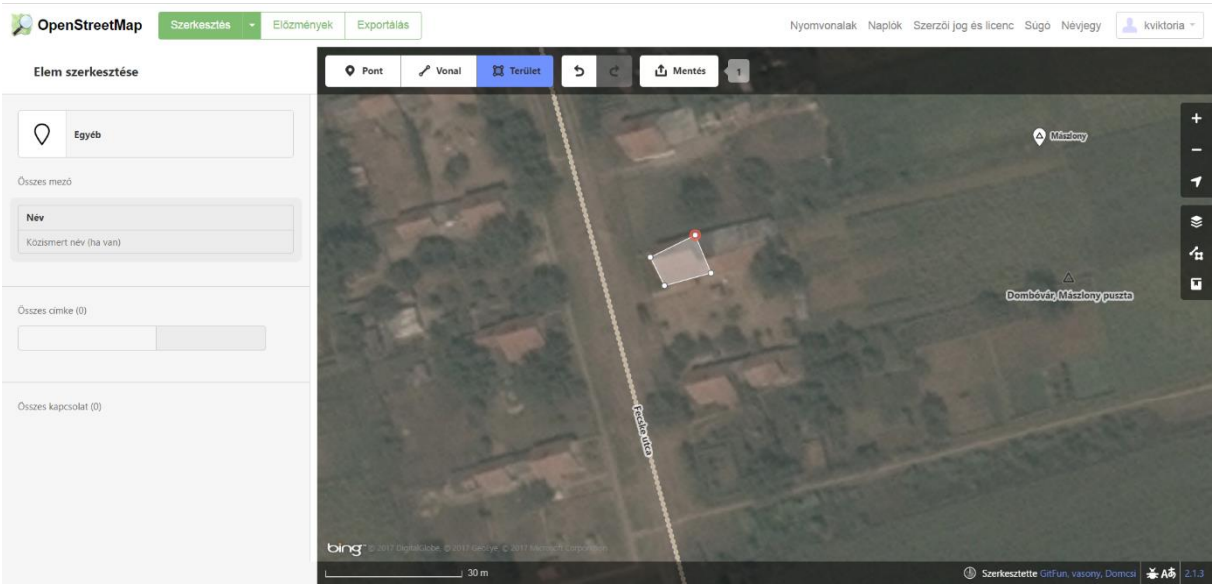
*28. ábra: A Reumann–Witkam-algoritmus eredménye 2 méteres küszöbérték használatával*

## Adatfelvétel során felmerülő kisegítő lehetőségek

A térképi adatok minőségét jelentősen javítja, ha az épületek formája már az adatfelvétel során megfelel a korábban említett elvárásoknak. Fontos, hogy a valóságban derékszögű épületsarkok a térképen is derékszögűek legyenek, ne csak közelítsék azt, és utcafronti faluk párhuzamosan fusson az utca középvonalával. Emellett az is fontos, hogy ne fedjék egymás a különálló poligonok és az utakra se lógjanak rá. Az utóbbi két feltételt kézi javítással tudjuk megoldani a felvétel során, a másik feltételek kielégítésére rendelkezésünkre állnak különböző segédeszközök.

### OpenStreetMap - böngészőben futó szerkesztők

Az OSM bárki által szabadon szerkeszthető internetes térkép. Bárki mégsem szerkesztheti, egy nagyon egyszerű regisztrációra szükség van hozzá. A szerkesztő mód nem jelenik meg azonnal, csak egy bizonyos nagyítás után, és az aktív Szerkesztés gomb melletti lenyíló menüvel kiválaszthatjuk, hogy milyen platformon szeretnénk a szerkesztést végrehajtani. Legegyszerűbb a böngészőben futó *iD* szerkesztő (29. ábra). Ezzel megmarad a térkép alapnézete, csupán kiegészül a szerkesztéshez szükséges eszköztárral. Kiválaszthatjuk, hogy pontot, vonalat vagy területet vinnénk fel és persze lehetőség van visszavonásra is. Az általunk végrehajtott változtatások nem mentődnek automatikusan, csak a mentés gombra kattintva, így elkerülhető a hibás elemek felvétele (Openstreetmap Wiki).



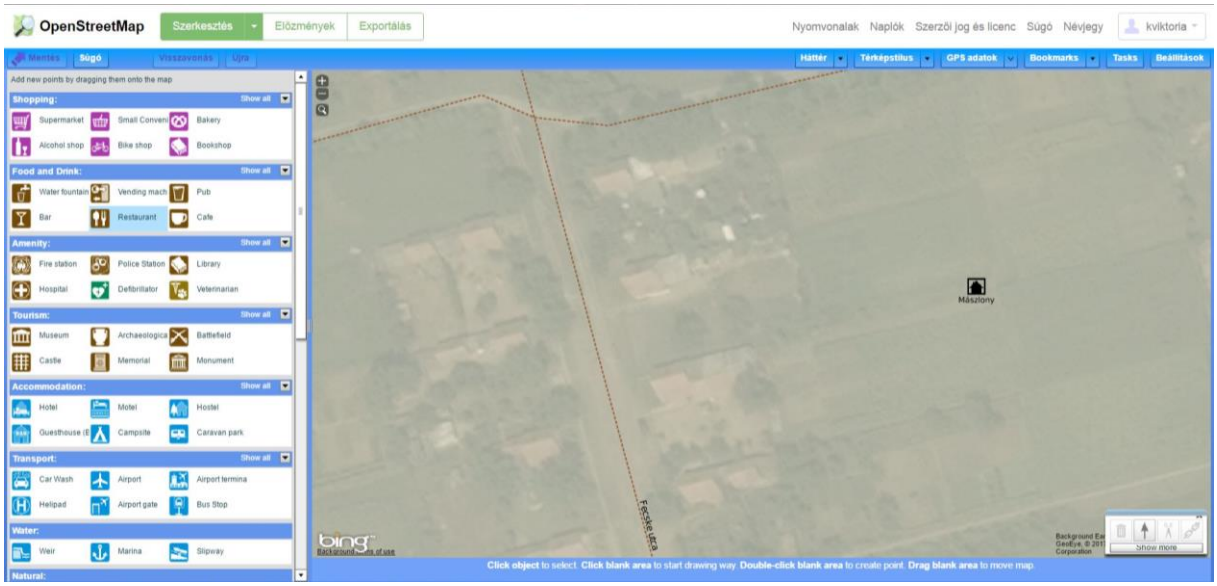

*29. ábra: iD OpenStreetMap szerkesztő*

Én egy külterületi lakott helyet választottam, ahol nincsenek épületek rajzolva még. Egy nagyobb épületet nagyjából körülrajzoltam. A poligon bezárásához az utolsó pontra kell duplán kattintani. Ezután kiválaszthatjuk az elem típusát. Az elemre duplán kattintva megjelenik egy

kisebb menüsor, különböző szerkesztési lehetőségekkel: elforgatás, tükrözés, szögletesítés, mozgatás, körré alakítás és törlés. A terület sarkainak szögletesítése a sokszög sarkait derékszögűvé alakítja, eredménye a 30. ábrán látható. Ez a parancs elérhető az S billentyű lenyomásával is.


*30. ábra: Az épületek sarkainak „derékszögesítése”*

A másik böngészőből használható szerkesztő a *Potlatch* (31. ábra). Ez kicsit lassabb, illetve jóval bonyolultabbnak is tűnik. Oldalt láthatjuk a térképi elemek típusait. Ezeket drag&drop módszerrel a térképre helyezhetjük, ha pontszerű elemet szeretnénk. A vonalak és területek rajzolásához elég csak a térképfelületre kattintanunk, és rajzolnunk. Ezután beállíthatjuk az elem típusát. Különlegessége, hogy olyan elem, hogy lakóépület, nem található a típusok között. Itt nincs megjelenő segédmenü a javításokhoz, csak a billentyűparancsokkal dolgozhatunk. A sarkok derékszögesítése itt a Q billentyű megnyomásával történik.



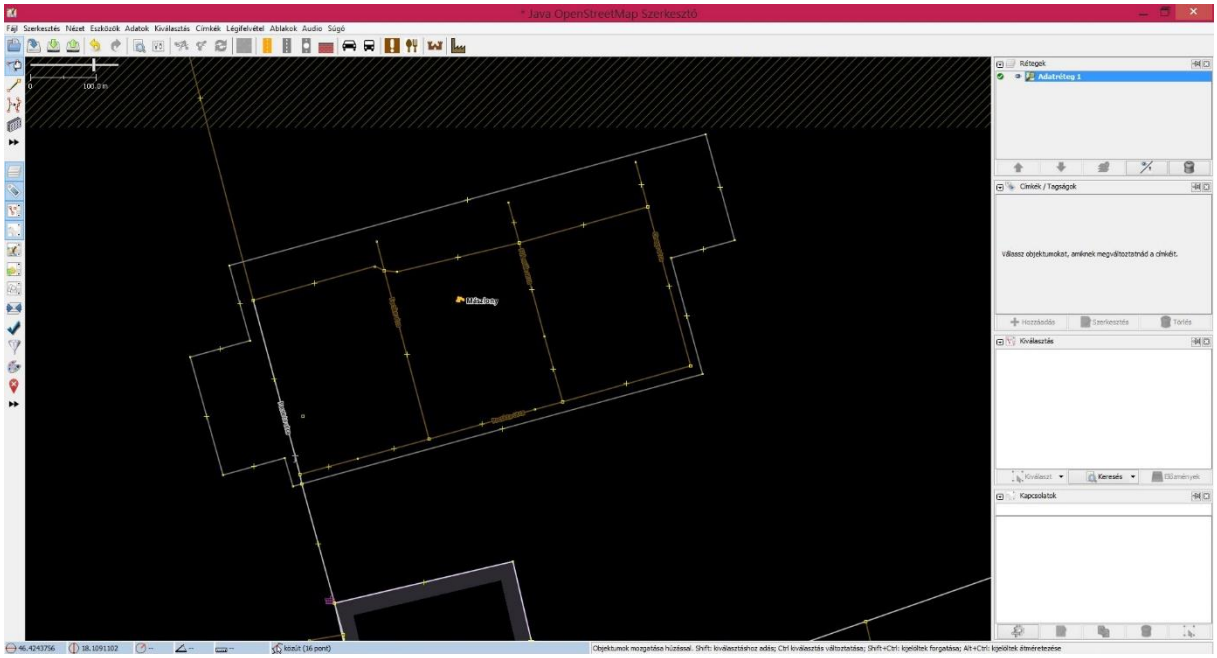

*31. ábra: A Potlatch szerkesztő*

Ebben az alkalmazásban két másik művelet is van, ami segítségünkre lehet adatfelvétel során, bár sajnos nem az épületek esetében. Létrehozhatunk egy korábbi vonallal párhuzamos vonalat, illetve vonal egyszerűsítést is végrehajthatunk.

A probléma ezekkel a két művelettel az, hogy bár a derékszögeket kijavítja, az elem térbeli helyzetét megváltoztatja. Mivel a szögek javításánál nem tudjuk kiválasztani, hogy melyik él maradjon érintetlen, sok esetben elforog a poligon, illetve néha el is torzul a műholdképen látotthoz képest. Ezt a műveletek között található forgatással, és a node-ok további igazításával persze könnyű kijavítani, de fontos, hogy a sarkok helyesbítése után ellenőrizzük az elemünk pontosságát.

## JOSM - Java alapú OSM szerkesztő program

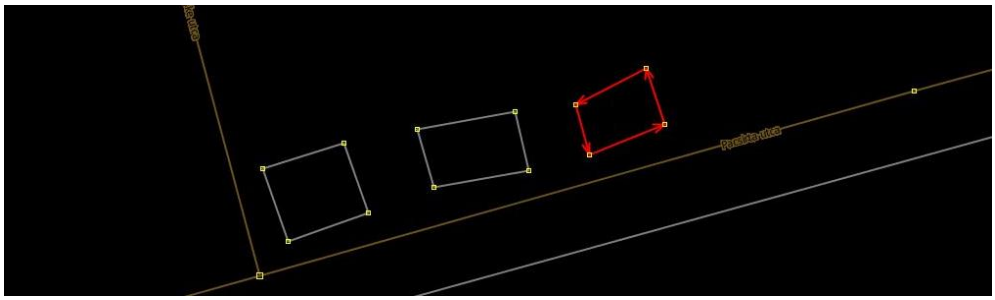
A JOSM (32. ábra) egy asztali alkalmazás, amit arra fejlesztettek ki, hogy tapasztaltabb szerkesztők jóval könnyebben tudják az OSM adatbázisát szerkeszteni. Rendkívül összetett, sok szerkesztést segítő módszere van, és több plugin is rendelkezésre áll, amik megkönnyítik és meggyorsítják a szerkesztést. Használata már nem olyan egyszerű, mint a korábban említett két lehetőség, de egy kis olvasgatás után könnyen elsajátítható.




*32. ábra: JOSM szerkesztő*

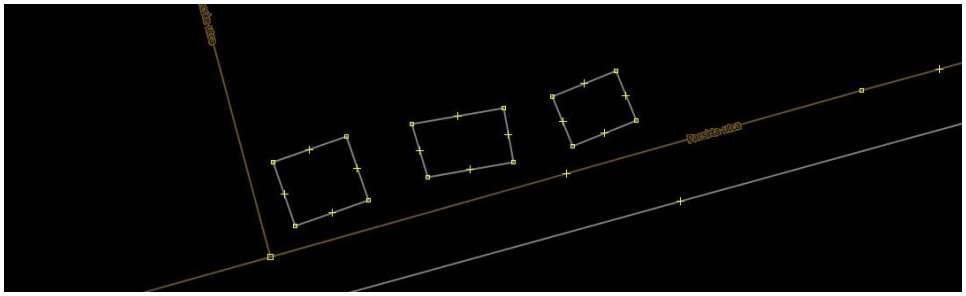
Alapvetően ugyanazokkal az eszközökkel rendelkezik, mint az iD és Potlatch szerkesztők, de az elemeket különböző rétegeken hozhatjuk létre. Importálhatunk gps nyomvonalakat, multipoligonok létrehozására is van lehetőség, és az OpenData plugin használatával shape fájlokat is importálhatunk és tölthetünk fel OSM-re (JOSM Wiki).

A pluginok a josm.openstreetmap.de/wiki/Plugins weboldalról tölthetők le, vagy a Beállítások menüben, a Bővítmények almenüből adhatók hozzá a programhoz. Az épületek generalizálására létrehozott modul a Building Generalization nevet viseli. A derékszöghöz közeli sarkokat derékszöggé alakítja, és a poligonok oldalait párhuzamosítja az utcával.



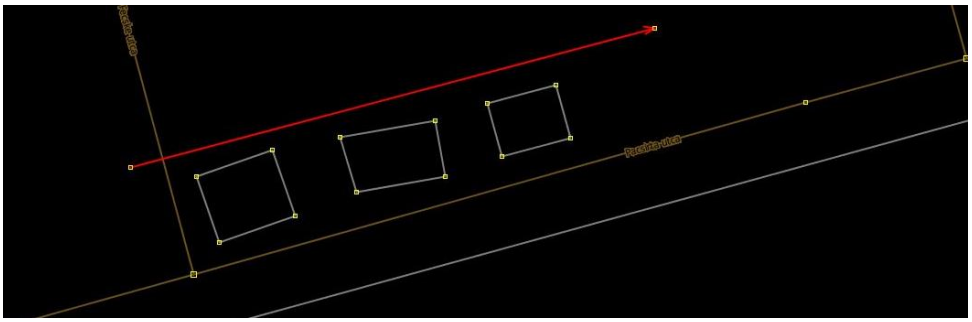
*33. ábra: A megrajzolt poligonok*

A javítás végrehajtásához nincs szükség kijelölésre, a térképlapon található minden elemre elvégzi a műveletet. A derékszöggé alakítás csak 84 és 96° közötti szögeknél működik, a párhuzamosításnál nincs ilyen kitétel. A 34. ábrán jól látszik, hogy a 90°-hoz közelítő sarkokat átalakította, míg a többit érintetlenül hagyta.



*34. ábra: Derékszögesítés eredménye*

Az úttal való párhuzamosítást csak általunk rajzolt vonal esetén hajtja végre, de a 35. ábrán látható, hogy itt is csak néhány fokos eltérés esetén hajtódik végre a javítás.



*35. ábra: Vonalhoz igazítás*

## OCAD

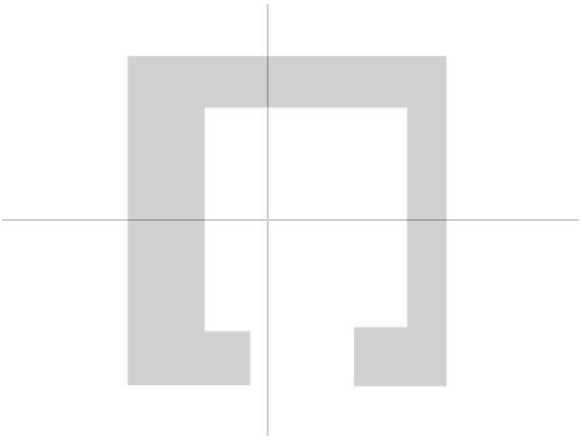
Az OCAD jelenleg egy svájci cég által fejlesztett térképrajzoló program. Egy szoftvercsaládot takar, amivel lehetőség van topográfiai térképek, internetes térképek és különböző tematikus állományok létrehozására. Importálhatunk és exportálhatunk adatokat gps-re, vagy shape fájlba, webes térképeket hozhatunk létre vele, a térképet publikálhatjuk több különböző állományba és az egyik legújabb lehetősége a ThematicMapper használata (OCAD Inc.).

A program erőssége az előre megtervezhető és már alapértelmezettként beépített jelkulcsok, amivel térképsorozatok is készülhetnek, illetve a tájfutó térképek készítése. Sok rajzolási módszer közül választhatunk (36. ábra):

- Görbe vonal
- Ellipszis
- Kör
- Szögletes vonal
- Szögletes poligon
- Lépcső
- Törött vonal
- Szabadkézi rajzolás

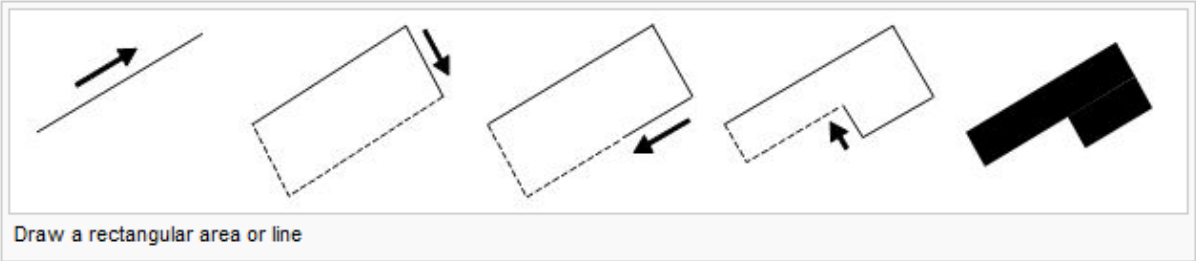
*36. ábra: Rajzolási lehetőségek OCAD-ben*

Épületek rajzolásához a szögletes poligon mód használata szükséges. Ha egyszerűbb, csak egymásra merőleges oldalakból álló házakat kell rajzolni, ez a módszer segít a térkép minőségének javításán úgy, hogy a derékszögű sarkok a rajzolás során valóban derékszögűek lesznek, ahogy a 37. ábrán is látható. Ezt a szabadkézi poligonrajzolással nehezebb és jóval lassabb megvalósítani.



*37. ábra: A szögletes rajzolási mód eredménye*

A művelet során egy alapvonal húzása után, arra merőleges vonal húzásával egy téglalap vázát építi fel. Ezután húzhatunk további vonalakat. A program minden egyes oldal rajzolása közben mutatja, hogy milyen lesz a poligonunk. A rajzolás folyamata a 38. ábrán látható.



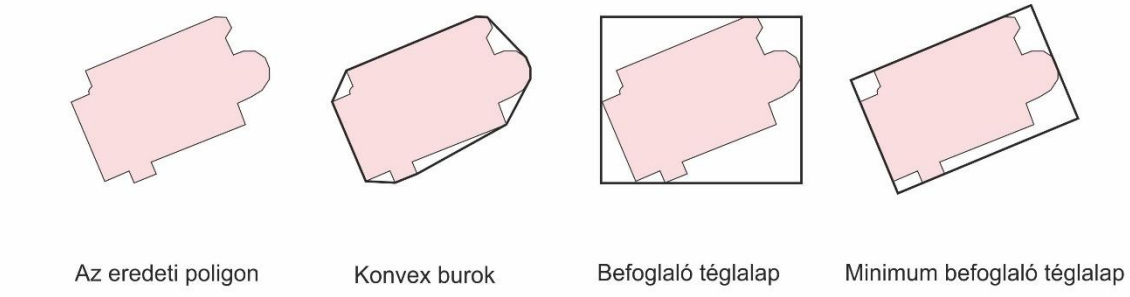

*38. ábra: A szögletes mód rajzolási folyamata*

# 5. Saját, az épületek alakját generalizáló program készítése

A korábban említett épületgeneralizálási folyamatok, mint például a minimumkeresés vagy az alaktani változókon alapuló egyszerűsítés mind a kisebb méretarány változást követő generalizálást vették célba. Leegyszerűsítik az épületpoligonokat úgy, hogy közben megtartják a főbb tulajdonságait. Ez kisebb méretarány-csökkenés esetén szükséges, amikor még ábrázolhatjuk a kisebb részeket.

Nagyobb méretarány-változás esetén már a minimális térképi távolságok miatt a kiugró részeket, vékony folyosókat nem tudjuk ábrázolni, ezért ezek eltávolítására van szükség. Minél erősebb a generalizálás, annál kevesebb oldallal (falfelülettel) fog rendelkezni a poligon, míg a poligon négyszöggé nem egyszerűsödik.

A poligont így reprezentálhatjuk egy négyszöggel, ha nagyobb méretarány csökkenés után szeretnénk megjeleníteni azt. A különböző lineáris egyszerűsítési megoldások alkalmazásával ezek létrehozása elég számításigényes és lassú folyamat, ezért helyettesíthetjük befoglaló négyszög létrehozásával is. Ez a négyszög megfelelő helyjelzőként funkcionál, de hátránya, hogy oldalai a koordinátarendszer oldalaival párhuzamosak, így sokkal nagyobb helyet elvehet a térképi helyből. Ez a probléma könnyen kiküszöbölhető a minimum befoglaló négyszög használatával. Ennek oldalai a forgatás közben létrehozott koordinátatengely-párok valamelyikével párhuzamosak, jobban mutatják az elemek irányultságát is. A 39. ábrán jól látható a különböző befoglaló poligonok közötti különbség.




*39. ábra: A Mátyás-templom befoglaló poligonjai*

A minimum befoglaló négyzet létrehozására a *rotating calipers* (forgó tolómérce) algoritmus (Toussaint, 1983) használata a legmegfelelőbb. Ez a módszer az eredeti poligon konvex burka alapján számolja ki annak minimum befoglaló négyzetét alapul véve. Emellett bevezethető egy fontos szabály, ami megkönnyíti a számítást:

„Egy konvex poligon minimum befoglaló téglalapja rendelkezik olyan oldallal, ami egybeesik a poligon egyik élével.” (Freeman and Shapira, 1975)

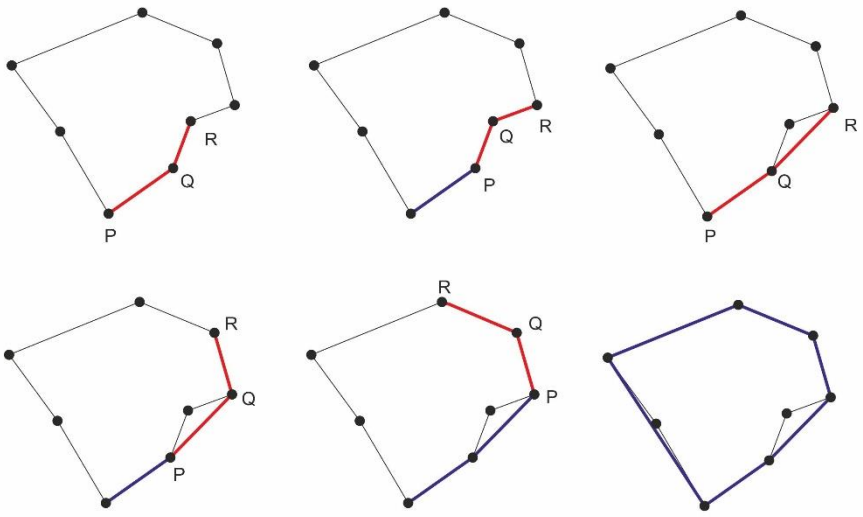
## A konvex burok létrehozása

Egy pontfelhő, vagy poligon konvex burka az a legkisebb területű konvex sokszög, ami mindegyik pontot tartalmazza (De Berg et al. 2000). A konvex burok létrehozására leggyakrabban használt algoritmusok a Graham pásztázó, és Jarvis march algoritmusok.

### Graham pásztázó algoritmusa (Graham, 1972)

Az algoritmus célja, hogy a pontfelhő szélén található pontokat (vagy a poligon csomópontjait) vizsgálja, eltávolítva a konkáv részeket. Egyszerre három pont segítségével halad végig a poligon szélén, óramutató járásával ellentétes irányban (40. ábra).

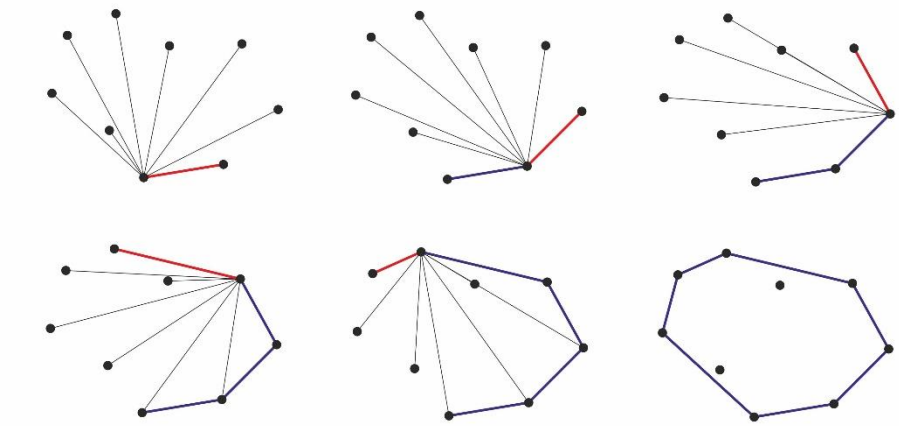
Elsőként megkeresi a legkisebb *y* koordinátával rendelkező *P* pontot. Ha több ilyen is van, azok közül a legkisebb *x* koordinátával rendelkezőt választja. Ezután egy rendező algoritmus kiválasztja a következő *Q* pontot, ami legközelebb található a kezdeti ponthoz. A pásztázó algoritmus ugyanezzel a rendező algoritmussal kiválasztja a harmadik *R* pontot is. Azt vizsgálja, hogy a *P*-ből *R*-be a *Q* ponton keresztül bal vagy jobb fordulattal jutunk el. Ha bal fordulat volt, *Q*-t eltávolítjuk, és eggyel hátrébb mozgatjuk a pontokat. Ha jobb fordulatot formálnak, vagy egy egyenesen fekszenek a pontok, felvesszük őket a burokba, és továbbléptetjük őket. Addig ismétlődnek ezek a lépések, amíg P pont újra a kezdőpontra nem ér.




*40. ábra: Graham pásztázó algoritmusa*

### Jarvis march algoritmus (Jarvis, 1973)

Az algoritmus kezdőpontjának tetszőlegesen választhatunk egy olyan $P_i$, $i=0$ pontot, ami biztosan a konvex burok része lesz. Négy ilyen pont lehet, a minimális és maximális $x$, illetve $y$ koordinátával rendelkező pontok. A kiválasztott pontból óramutató járásával ellentétesen indulva kiválaszt $P_{i+1}$ egy pontot úgy, hogy a kezdeti és a kiválasztott ponton átmenő egyenestől balra helyezkedjen el a poligon (vagy pontfelhő) összes többi pontja. $P_i$ mindig egy polárkoordináta-rendszer központja, és az összes többi pontot a koordinátái alapján vizsgálja (41. ábra).



*41. ábra: A Jarvis march algoritmus működési elve*

## Forgó tolómérce algoritmus (Toussaint, 1983)

Az algoritmus lényege, hogy a konvex burok minden élére létrehozunk egy-egy befoglaló téglalapot, majd ezek közül a legkisebb területűt kiválasztva megkapjuk a minimum befoglaló téglalapot. Az algoritmus a következő:

1. keressük meg $X_{min}$, $X_{max}$, $Y_{min}$ és $Y_{max}$ pontokat, hogy ezek alapján létrehozzuk a befoglaló téglalapot
2. számoljuk ki a négy szöget, amit a téglalap oldalai bezárnak az őket óramutató járásával megegyező irányban követő poligonoldalakkal
3. a legkisebb szöggel rendelkező oldalt válasszuk ki a téglalap kezdeti irányának
4. hozzunk létre egy téglalapot, aminek egyik oldala párhuzamos a kiválasztott poligonéllel
5. méretezzük át a téglalapot, hogy befoglalja a poligont
6. számoljuk ki a téglalap területét
7. vizsgáljuk meg az egész poligont a 4-6. pontok ismétlésével.

## A program

A programom Python nyelvben készült, így szükség esetén kisebb változtatásokkal alkalmas lehet akár QGIS szkriptként való alkalmazásra. Célja, hogy shape fájlból beolvasott épületpoligonok minimum befoglaló téglalapját kiszámítsa, majd egy új, a program által létrehozott shape fájlba írja azokat. A minimum befoglaló téglalapok kiszámításához az előbb bemutatott folyamatot használtam, az alapadatoknak először a konvex burkát számoltam ki, majd a *forgó tolómérő algoritmus* implementálásával kiszámoltam a minimum befoglaló téglalapot.

Ahhoz, hogy a Python-ban térbeli adatokkal és az azokhoz használt beépített függvényekkel is tudjunk dolgozni, a GDAL/OGR és kiegészítésként a *NumPy* modulok telepítésére volt szükség. Az *IDLE Shell* ablakban láthatjuk, hogy melyik verzió fut, a verzióhoz tartozó telepítőket kell letölteni. Ezután a *PIP installer* segítségével parancssorból tudjuk ezeket telepíteni. A *home* könyvtárból kell elindítani:

```
pip install D:/letoltesek/ GDAL-2.1.3-cp36-cp36m-win_amd64.whl
```

A *NumPy* telepítése is ezzel a folyamattal történik.

Ahhoz, hogy a programunkban a vektorok adatokat tudjuk használni, a kód elején importálnunk kell a modulokat. Mivel matematikai számításokat, kifejezéseket is használtam a kódban, szükségem volt a *Math* függvénykönyvtár importálására is.

```
import ogr
import numpy
import math
```

Ezután a fájl beolvasásához szükségünk van egy úgynevezett *driver* definiálására, amiben megadjuk a fájl típusát, majd beolvassa azt.

```
driver=ogr.GetDriverByName('ESRI Shapefile')
pg = driver.Open('epuletek.shp',1)
layer = pg.GetLayer()
numFeatures = layer.GetFeatureCount()
```

Elsőként megadjuk a fájltípust, majd megnyitjuk azt. A fájlnév utáni *1* az írást megengedi, *0* használatával csak olvasni tudjuk a fájlt. Beolvassuk a réteget, majd megszámoljuk a térképen levő elemeket.

A fájlba íráshoz először megadjuk a létrehozandó fájl nevét, kiterjesztését. A kiíráshoz szükségünk van az *os* és *sys* modulok importálására is.

```
fname ='epuletek_mbr.shp'
import os, sys
os.chdir('D:\ELTE\MSC\SZAKDOLGOZAT\program')
driver = ogr.GetDriverByName('ESRI Shapefile')
```

Figyelni kell, hogy létezik-e már ez a fájl. Ha igen, a program felülírja.

```
if os.path.exists(fname):
    driver.DeleteDataSource(fname)
outDS = driver.CreateDataSource(fname)
```

Ha nem sikerül a fájl létrehozása, kilép a programból, és a következő hibaüzenetet írja ki:

*'Could not create file'*.

```
if outDS is None:
    print ('Could not create file')
    sys.exit(1)
```

Ezután létrehozzuk a fájlban található réteget (jelen esetben poligon) és az attribútumtábla mezőit megadjuk.

```
outLayer = outDS.CreateLayer('test', geom_type=ogr.wkbPolygon)
layerDefinition=outLayer.GetLayerDefn()
n=ogr.FieldDefn('NEV',ogr.OFTString)
n.SetWidth(100)
outLayer.CreateField(n)
```

Ezután következik a poligonok forgatására használandó függvény definiálása. A függvény négy változóval dolgozik. Megadjuk a forgatás szögét, a forgatandó elemet, az elem forgatási pontját (itt az elem centroidja) és az elemet alkotó pontok számát. A ciklusokon kívül definiáltam egy lineáris gyűrűt, amibe a függvény által kiszámolt pontok koordinátái kerülnek, és a ciklus lefutása után a pontokkal feltöltött gyűrűt poligonná konvertáljuk.

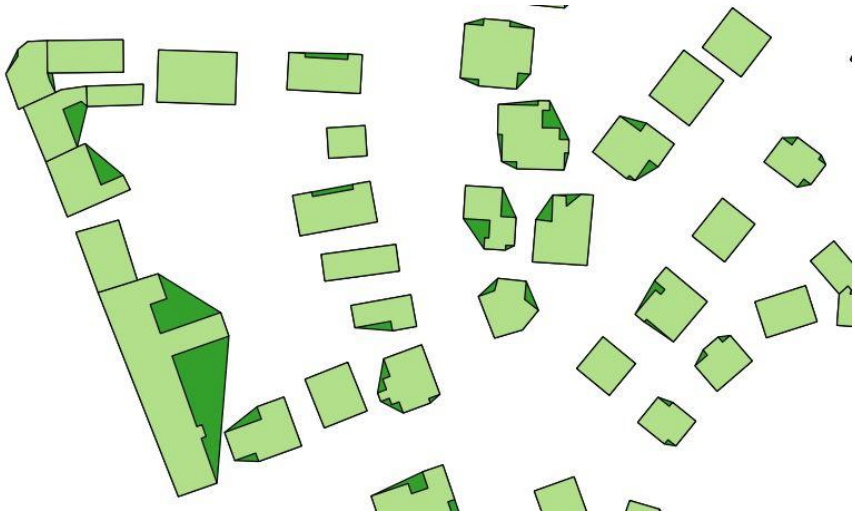
Az első *for* ciklus végig megy az összes elemen, míg a benne található második ciklus egy-egy elem pontjain halad végig, az első ponttól az utolsóig, és a koordináták alapján kiszámolja az elforgatott poligon új pontjait. Ezekkel a pontokkal töltjük fel a *ring* gyűrűket. A belső ciklusból kilépve pedig minden egyes gyűrűből létrehozunk egy-egy poligont. A függvény visszatérési értéke ez a *box* poligon lesz.

```
def forgat (szog,konvex,centroid,node):
    ring = ogr.Geometry(ogr.wkbLinearRing)
    for j in konvex:
        for k in range (0, node):
            kx = j.GetPoint(k)[0]
            ky = j.GetPoint(k)[1]
                        fx = centroid.GetY()+(math.cos(szog)*(ky-
            centroid.GetY())-math.sin(szog)*(kx-centroid.GetX()))
                        fy = centroid.GetX()+(math.sin(szog)*(ky-
            centroid.GetY())+math.cos(szog)*(kx-centroid.GetX()))
            ring.AddPoint(fy, fx)
        box = ogr.Geometry(ogr.wkbPolygon)
        box.AddGeometry(ring)
    return box
```

A konvex burok létrehozásához meg kell vizsgálnunk, hogy mi a fájlban található elemek geometriája. Erre azért van szükség, mert csak a poligonok konvex burkát szertnénk létrehozni. A *for* ciklus végighalad az összes elemen. Először megvizsgálja mindegyik geometriáját. Az *if* függvény feltétele, hogy a geometria poligon legyen. Ha ez teljesül, az OGR könyvtár *ConvexHull* függvényével létrehozzuk az elemek konvex burkát.

```
for i in range(0,numFeatures):
    feat = layer.GetNextFeature()
    geom = feat.GetGeometryRef()
    if geom.GetGeometryName()=="POLYGON":
        konvex = geom.ConvexHull()
```

Ezeket fájlba íratva a következő eredményt kapjuk (az 5. mellékletben látható teljes adatsor és a konvex burkok):



*42. ábra: Az épületek kiindulási állapota (világos zöld) és konvex burkuk (sötét zöld)*

Ezután a befoglaló téglalap létrehozásához szükséges változókat definiáltam.

```
centroid=konvex.Centroid()
minter = float("inf")
minszog = 0.0
mbr = ogr.Geometry(ogr.wkbPolygon)
```

Szükség van a poligon centroidjára, (centroid) melyet az OGR könyvtár *Centroid* beépített függvényének használatával kapok meg, a legkisebb területre (minter), aminek a végtelent adtam kezdő értéknek, hogy bármely később számolt terület kisebb legyen nála. Megadtam a minimális területű befoglalóhoz tartozó szöget (minszog), ami kezdetben 0, illetve létrehoztam egy poligont (mbr), ami később a minimális befoglaló téglalap lesz.

A *GetGeometryCount* függvényt a konvex burokra meghívva, megkapjuk, hogy hány elemből áll. Ezután a *for* ciklus végighalad az összes *j* elemen, gyűrűket hoz létre, majd ezek csomópontjainak számát megadjuk a *GetPointCount* függvény használatával.

```
konvexGeomSzama = konvex.GetGeometryCount()
for j in range(0,konvexGeomSzama):
  gyuru = konvex.GetGeometryRef(j)
  node = gyuru.GetPointCount()
```

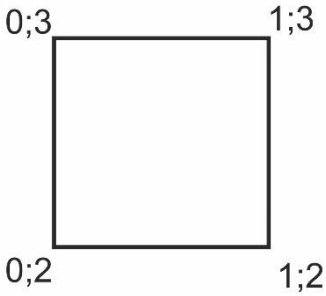
A következő *for* ciklusban az *i* csomópontokon haladunk végig. Minden *i* pontra kiszámoljuk az általa és a következő, *i+1* pont által alkotott poligonoldal szögét. A kapott szög mínusz 1-szeresével elforgatjuk a konvex burkot a korábban definiált forgat függvény használatával, majd a GetEnvelope beépülő függvény meghívásával kiszámoljuk a maximális kiterjedést.

```
for k in range (0,node-1):
      kix = gyuru.GetPoint(k)[0]
      kiy = gyuru.GetPoint(k)[1]
      kiix = gyuru.GetPoint(k+1)[0]
      kiiy = gyuru.GetPoint(k+1)[1]
      szog = math.atan2(kiix-kix, kiiy-kiy)

      forgkonvex = forgat(-1*szog,konvex,centroid,node)

      minmax=forgkonvex.GetEnvelope()
```

A GetEnvelope függvény visszatérési értéke egy tömb, benne négy koordináta, [0]=$X_{min}$, [1]=$X_{max}$, [2]$Y_{min}$, [3]=$Y_{max}$.

Ahhoz, hogy ezekből poligont tudjunk létrehozni, a kapott pontok segítségével meg kell adni a poligon koordinátapárjait. Ezek EOV-ben a 43.ábrán látható módon alakulnak.

A pontokat ezután óramutató járásával megegyező irányban a létrehozott lineáris gyűrűbe helyezem. Ahhoz, hogy zárt poligont alkosson, az első pontot kétszer, utolsóként is hozzá kell adni. Ezt a gyűrűt poligonná alakítjuk, így a *GetArea* függvény meghívásával kiszámíthatjuk területét.



*43. ábra: A GetEnvelope függvény visszatérési értékeinek EOV koordinátákká alakítása*

```
ring = ogr.Geometry(ogr.wkbLinearRing)
ring.AddPoint(minmax[0],minmax[2])
ring.AddPoint(minmax[0],minmax[3])
ring.AddPoint(minmax[1],minmax[3])
ring.AddPoint(minmax[1],minmax[2])
ring.AddPoint(minmax[0],minmax[2])

tegla = ogr.Geometry(ogr.wkbPolygon)
tegla.AddGeometry(ring)
terulet=tegla.GetArea()
```

Ezután már csak a legkisebb területű *tegla* befoglaló téglalap megtalálása következik. Ha a kiszámolt *terulet* kisebb, mint a korábbi, a minimális értéket jelölő *minter*, akkor a minimális érték felveszi az aktuális értéket, a *tegla* lesz a korábban definiált *mbr*, a hozzá tartozó forgatási szög pedig a *minszog*. Ez addig vizsgálja a területeket, míg meg nem találja a legkisebb területű befoglaló téglalapot.

```
if terulet < minter:
      minter = terulet
      mbr = tegla
      minszog  = szog
```

Az így létrejött téglalapok oldalai párhuzamosak a koordináta tengelyekkel (lásd 5. melléklet, c ábra), vissza kell őket forgatni az eredeti helyzetbe, hogy megfelelően reprezentálják az eredeti poligonokat.

```
visszaforg = forgat(minszog,mbr,centroid,5)
```

Mivel a korábbi forgatásnál a szög mínusz egyszeresét használtuk, itt az eredeti értéket használva kapjuk meg az eredeti elhelyezkedést. Az elforgatott téglalapokból létrehozzuk a *minimumboundingbox* nevű térképi elemet, ami felveszi azok geometriáját, ezután pedig a létrehozott fájl térképi rétegébe írjuk.

```
minimumboundingbox=ogr.Feature(layerDefinition)
minimumboundingbox.SetGeometry(visszaforg)
outLayer.CreateFeature(minimumboundingbox)
```

Végül kilépünk a programból.

```
pg.Destroy()
outDS.Destroy()
```

## A program futtatásának eredménye

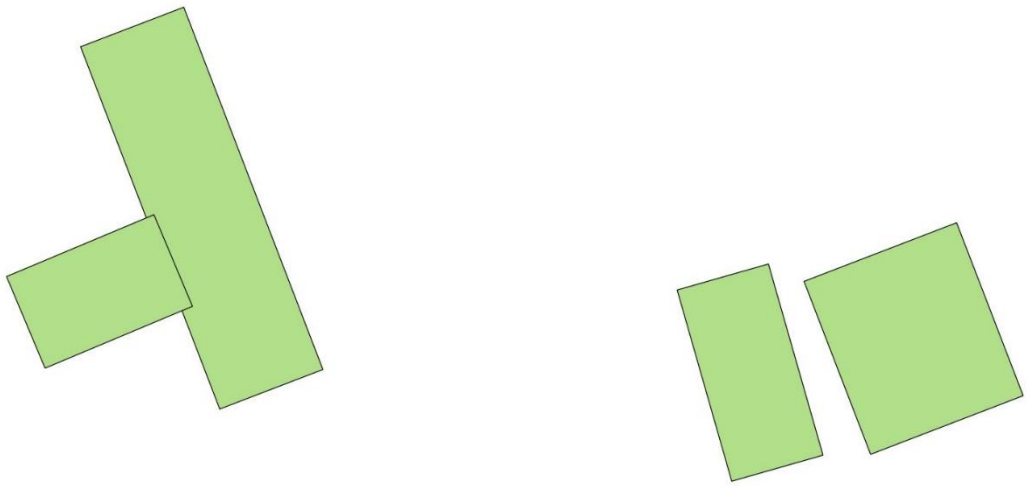
A programban megírt algoritmus létrehozta a térképi elemek minimális befoglaló téglalapját, ahogy ez a 44. ábrán is látható. Azonban ez az eredmény nem használható fel módosítások nélkül épületek generalizálására. Az eredeti, közeli objektumok befoglalói sok esetben metszhetik, fedhetik egymást, főleg hogyha nem egy vonalon, egy irányban helyezkednek el, vagy egymáshoz bonyolultabb formában, nem egy fal mentén csatlakoznak.

*44. ábra: A program eredménye, az épületek minimális befoglaló téglalapjai*

Ahhoz, hogy megfelelő térképi olvashatóságot biztosítson, további manuális generalizálásra van szükség. Néhány egyszerűsítési eljárás arra törekszik, hogy a létrehozott poligonok területe megegyezzen az eredeti poligonokéval, így az algoritmust szükség esetén ki lehet egészíteni a téglalapok kicsinyítésével, de jelentős változást ez nem hozna az eredményben.

Ahogy korábban említettem, az épületek generalizálása két irányból közelíthető meg, a poligonok egyszerűsítése illetve az épületek számának csökkentése. Mivel az algoritmus futtatásával elértük a maximális alaki generalizálást, így már csak az épületek számának csökkentése egyszerűsítheti tovább az eredményt. Először azokat a befoglalókat kell kiszűrni, amik mérete nem éri a térképen megjeleníthető minimális méretet, az általam tesztelt adatsorban is van néhány ilyen, kisebb téglalap. Ha a méret szerinti válogatás után is túl zsúfolt a terület, a fontosság alapján érdemes vizsgálni a megmaradt épületeket. A kisebb épületek, amiket fed egy nagyobb, jellegzetes fekvésű épület, eltávolíthatók, és a házsorok is ritkíthatók, ha jellegzetes struktúrájuk megmarad. A fontosabb épületeket (nagyobb, vagy pedig jellegzetes fekvésű) érdemes megtartani, hiszen azok jól jellemzik az eredeti struktúrát. Mivel sok esetben a poligonok így is egymásra lóghatnak, eltolásuk is szükséges lehet. Ha kialakultak olyan épületcsoportok, amik szorosan egymás mellett állnak, és hasonló méretűek, egy poligonná való összevonásuk is javít a térkép olvashatóságán.

A kiválasztott négy épületre is lefuttattam az algoritmust. A 45. ábrán is jól látható, hogy nem mindegyik épület generalizására megfelelő a legkisebb befoglaló téglalap használata.



*45. ábra: Az algoritmus futtatásának eredménye a kiválasztott épületekre*

A különleges formájú épületeket nem reprezentálja megfelelően, ez a Halászbástya esetében a legszembetűnőbb. A forma megtartása főleg kiemelt épületek esetén fontos, ezért azok generalizálását célszerűbb manuálisan elvégezni.

# 6. Összegzés

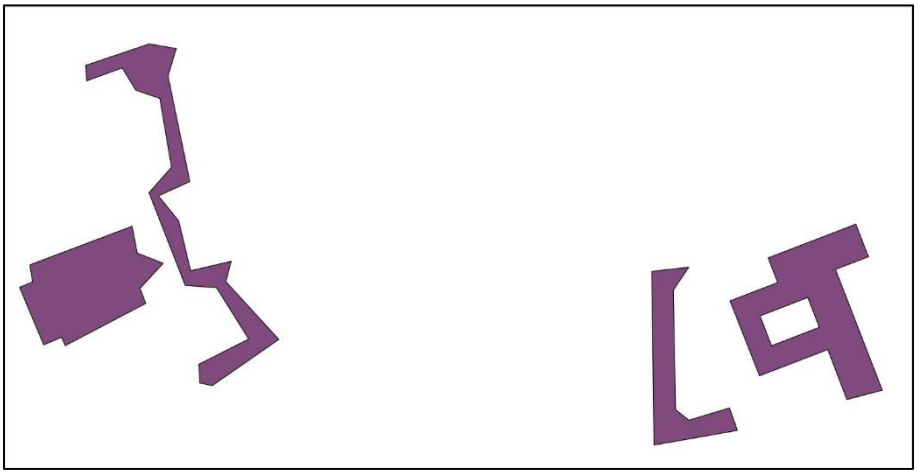
Az épületek generalizálásának problémája napjainkban is jelentkező probléma. Mivel ember alkotta objektumok, gyorsan változnak, így térképezésük, adatbázisba helyezésük folyamatosan szükséges. Ez látható a bemutatott módszereken is, többségük az elmúlt pár évben került kidolgozásra. A módszerek az épületek két fő tulajdonságát veszik alapul, miszerint az épületek poligonjait egyenes oldalak határolják, amik derékszögben metszik egymást. Ezek a módszerek és a térinformatikai szoftverekben található algoritmusok is e két tulajdonság megtartását és szükség esetén javítását, megerősítését tűzték ki célul. Dolgozatomban bemutattam, hogy ez a hozzáállás működik azokon az épületeken, amik megfelelnek a két fő felvetésnek. Ezeket a poligonokat szépen kezelik, megfelelő generalizálási eredményt nyújtanak. Az általam készített algoritmus minimális befoglaló téglalapokat alkot, ami szintén az ideális formájú épületek generalizálásánál megfelelő a reprezentálásra.

Ez a két szabály azonban általánosítás, sok épület nem felel meg ezeknek az elvárásoknak. Tartalmaznak egyenes vonalakat, amik nem derékszögben metszik egymást, és sok esetben köríves falakat is (ezeket a térinformatikában sokszöggel közelítjük), amik problémát okoznak a jelenleg létező módszereknek. Mivel a derékszögek kialakítására törekednek, a hosszabb oldalakat tartják meg, a többit pedig hozzátorzítják. A tesztelés közben ez a Halászbástya példáján látszódott legjobban. Míg a kisebb értékek eltávolították a bástyák némelyikét, az értékek növelése nem további bástyák eltűnését, hanem a főbb vonal futásának eltorzítását eredményezte. Az általam készített algoritmus sem alkalmas ennek reprezentálására.

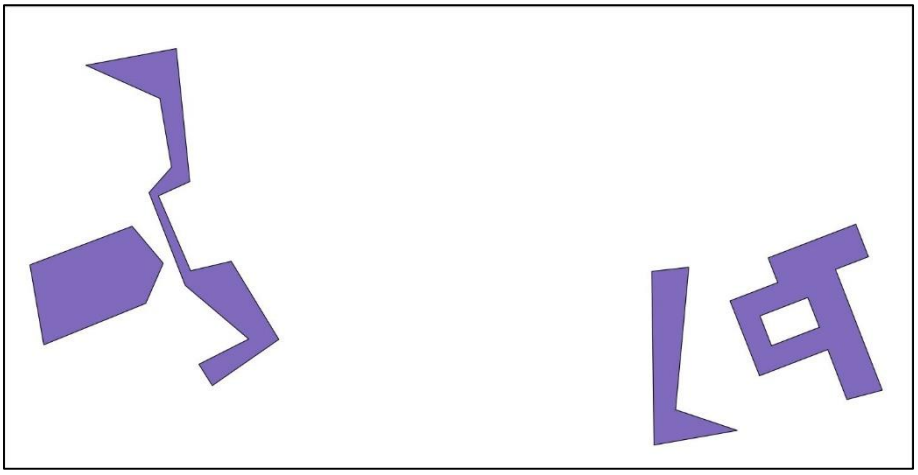
Az épületek többségére nagyon jól használhatók a jelenlegi módszerek, változatos célméretarányba történő automatikus generalizálásra, egészen addig, amíg csak téglalappal jelezzük azok térbeli helyzetét. Az automatikus generalizálás problémája azonban, hogy nem mindegyik formát kezeli megfelelően, így még továbbra is szükség van az ember szubjektivitására, ami a manuális generalizálás alapja. Amíg ezt a látásmódot nem sajátítják el az algoritmusok, az épületek automatizált generalizálása nem lesz teljes értékű.

# 7. Mellékletek

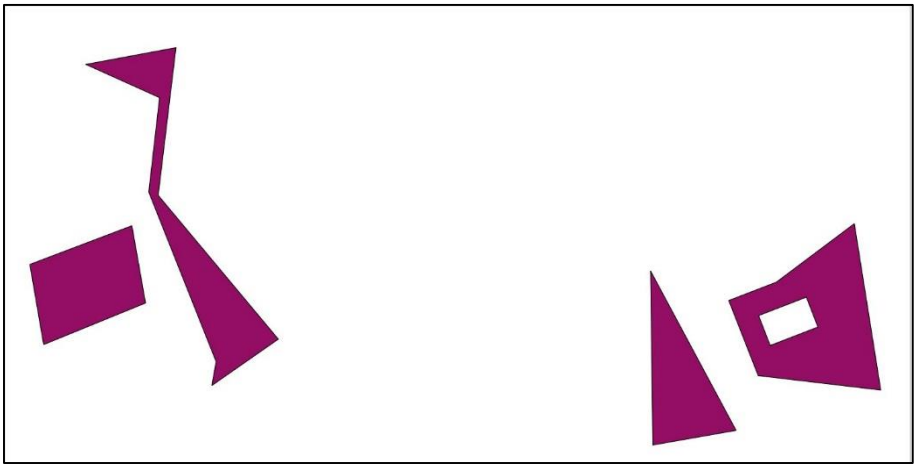
## 1. Melléklet



*1/a melléklet: A Douglas–Peucker algoritmus 3,5 méteres küszöbértékkel*

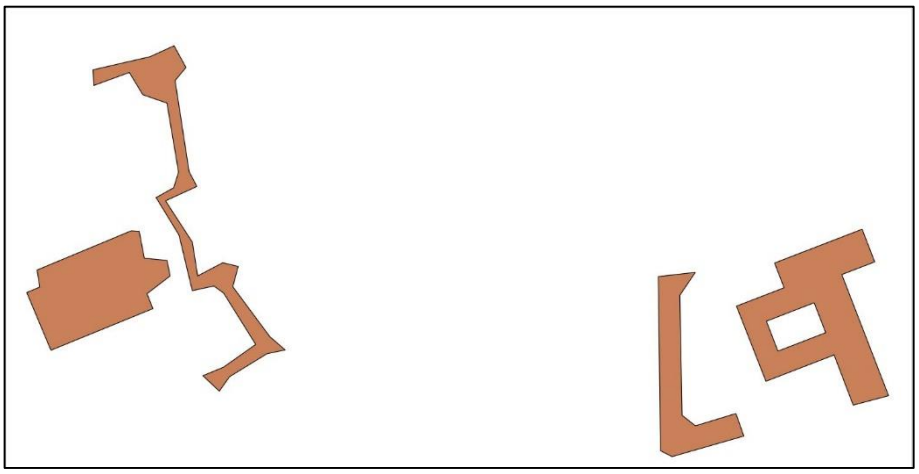


*1/b melléklet: A Douglas–Peucker algoritmus 8,5 méteres küszöbértékkel*

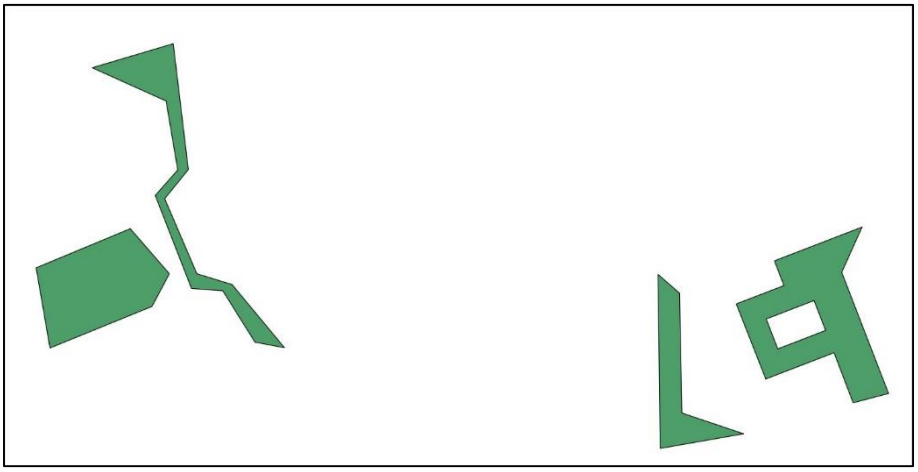


*1/c melléklet: A Douglas–Peucker algoritmus 11 méteres küszöbértékkel*

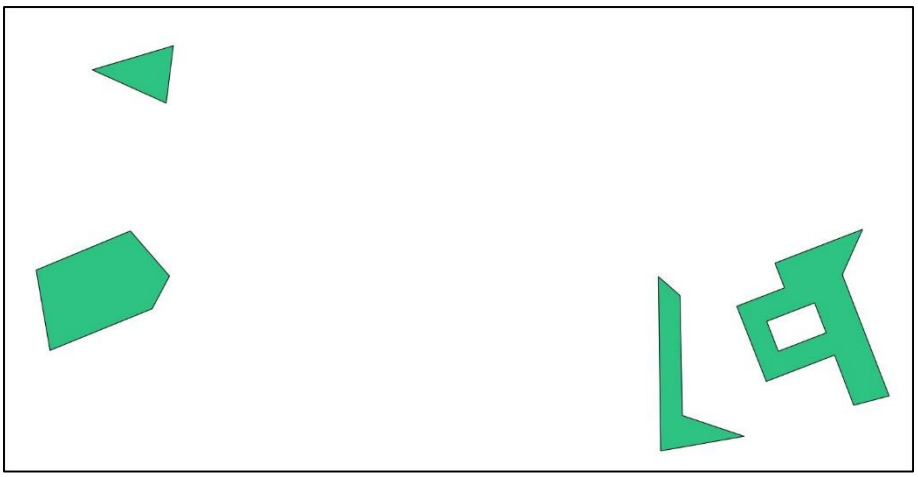
# 2. Melléklet



*2/a melléklet: A Visvalingam algoritmus eredménye 35 $m^2$ toleranciaértékkel*

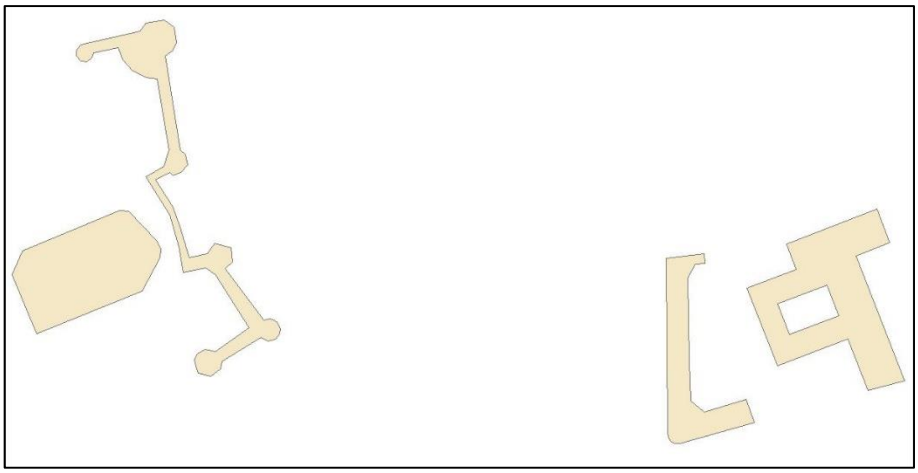


*2/b melléklet: A Visvalingam algoritmus eredménye 250 $m^2$ toleranciaértékkel*

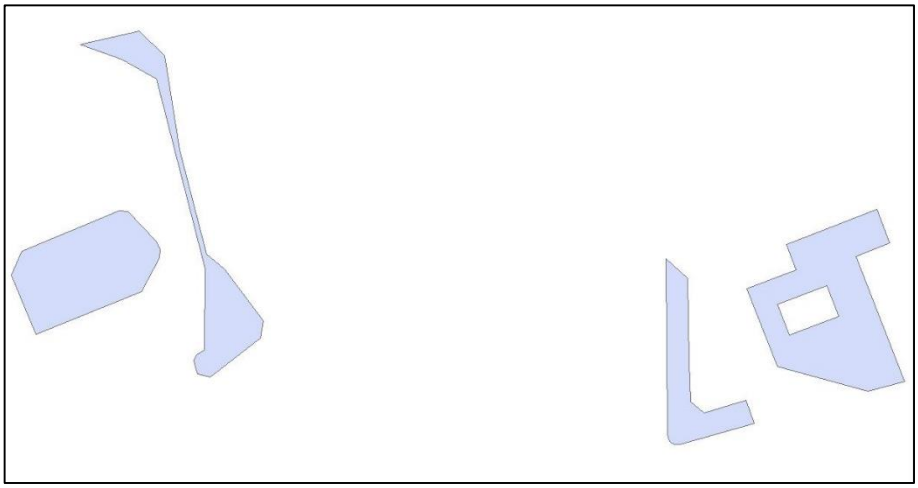


*2/c melléklet: A Visvalingam algoritmus eredménye 350 $m^2$ toleranciaértékkel*

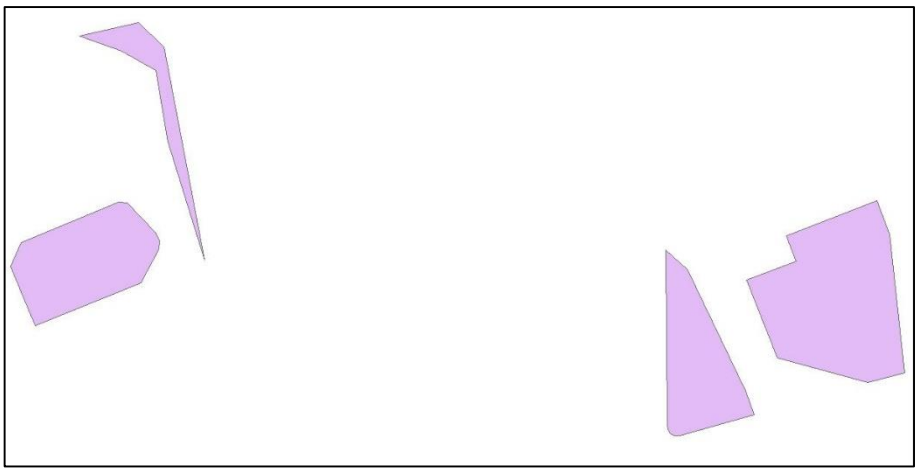
# 3. Melléklet



*3/a melléklet: A Bend Simplify algoritmus eredménye 10 méteres küszöbértékkel*

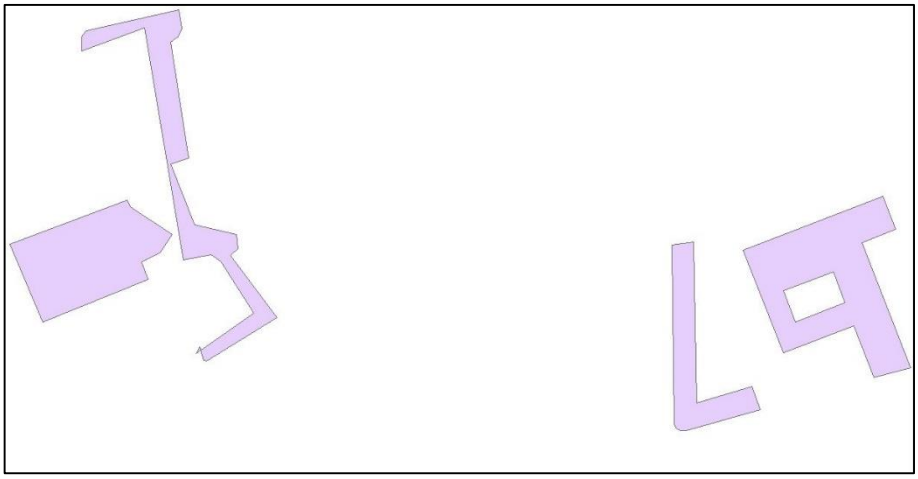


*3/b melléklet: A Bend Simplify algoritmus eredménye 35 méteres küszöbértékkel*

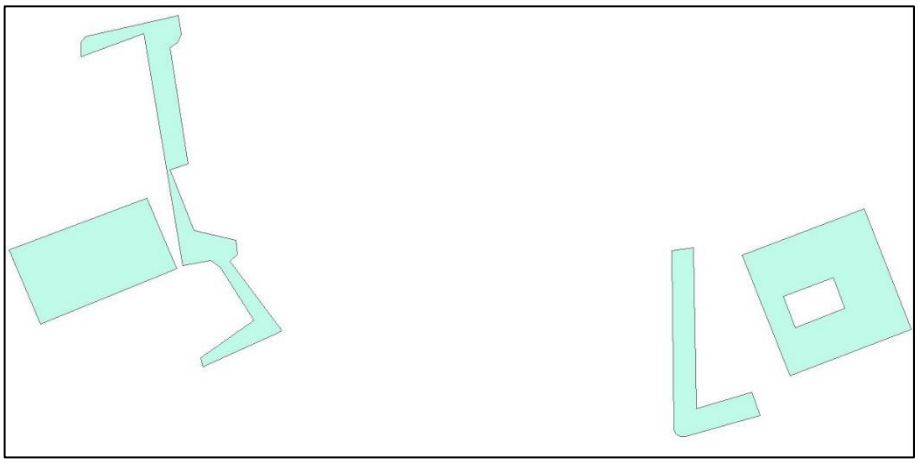


*3/c melléklet: A Bend Simplify algoritmus eredménye 50 méteres küszöbértékkel*

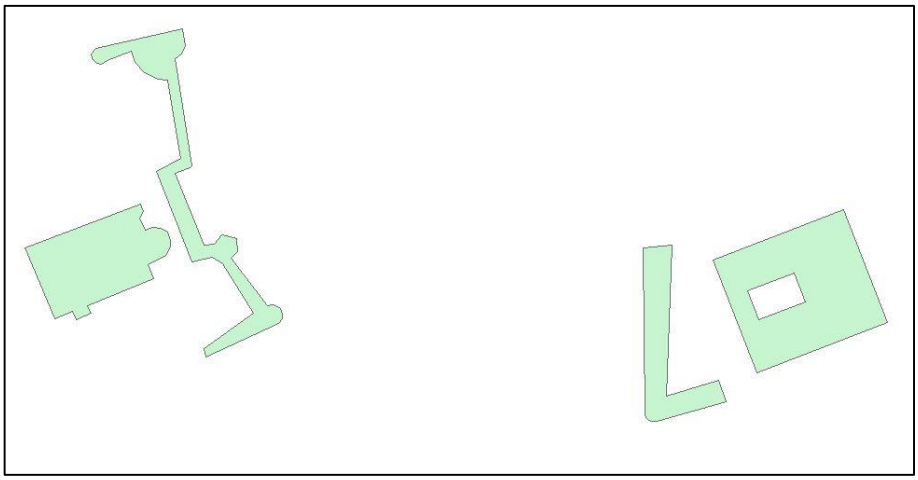
# 4. Melléklet



*4/a melléklet: A Simplify Building algoritmus eredménye 15 méteres határértékkel*

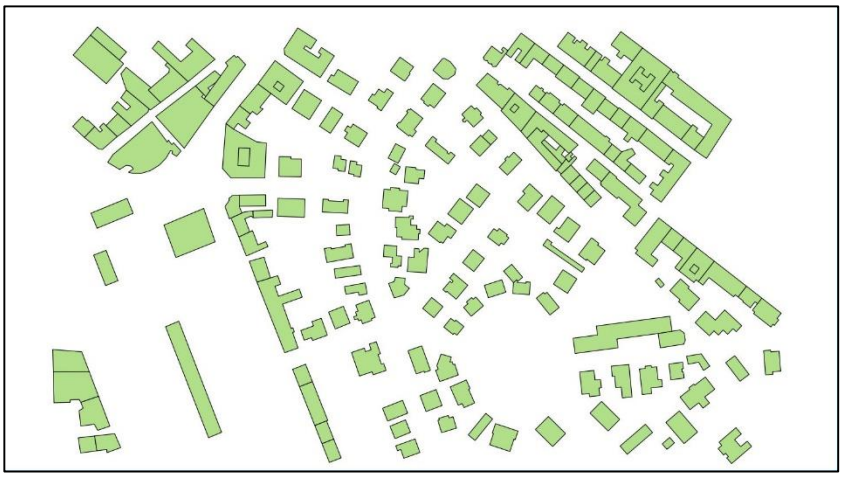


*4/b melléklet: A Simplify Building algoritmus eredménye 25 méteres határértékkel*

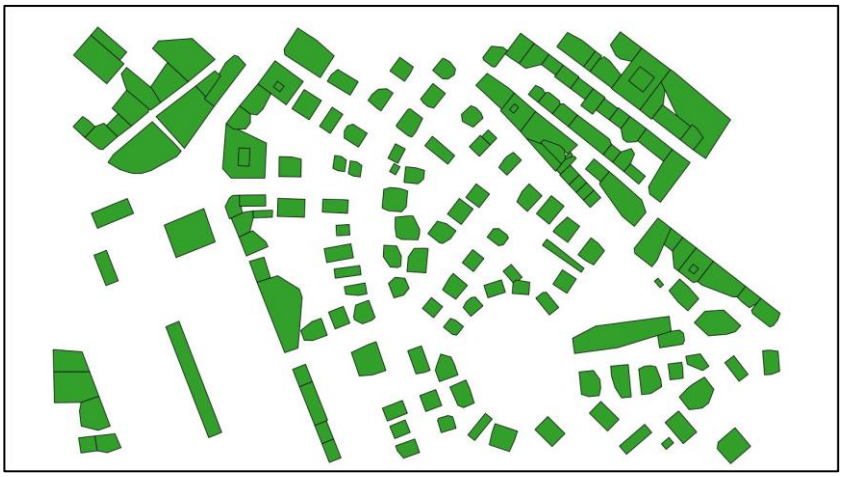


*4/c melléklet: A Simplify Building algoritmus eredménye 55 méteres határértékkel*

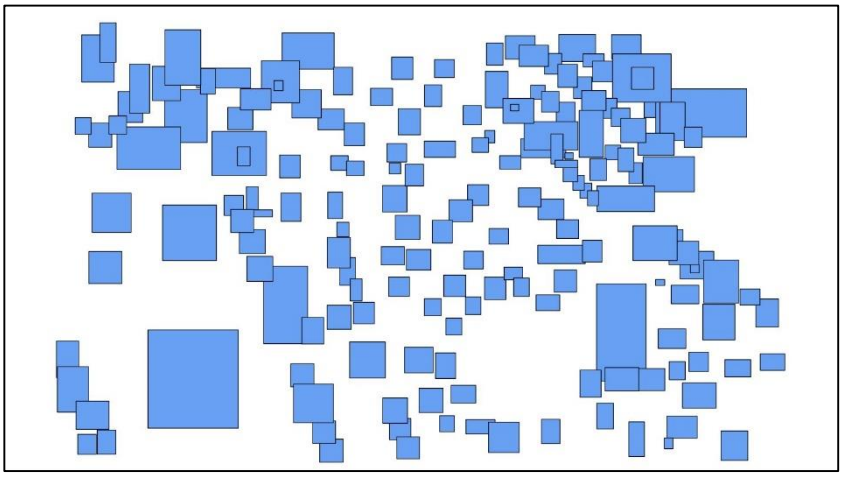
# 5. Melléklet



*5/a melléklet: A programhoz használt kiindulási anyag*



*5/b melléklet: A konvex burkok*



*5/c melléklet: A minimum befoglaló téglalapok visszaforgatás nélkül*

# 8. Köszönetnyilvánítás

Szeretnék köszönetet mondani témavezetőmnek, Ungvári Zsuzsannának, a téma kiválasztásában és körvonalazásában nyújtott segítséget és türelmet, valamint a dolgozat és a program készítése közben nyújtott folyamatos útmutatást.

# 9. Hivatkozott irodalom

### Felhasznált nyomtatott, illetve interneten elérhető kiadványok, publikációk


O. Aichholzer, D. Alberts, F. Aurenhammer, B. and Gärtner, (1995): Straight skeletons of simple polygons. Proceedings of 4th International Symposium of LIESMARS, pp 114–124.

M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, (2000): Computational Geometry: Algorithms and Applications, Springer, pp. 2–8.

D. Burghardt & A. Cecconi (2007): Mesh simplification for building typification. International Journal of Geographical Information Science Vol. 21 , Iss. 3.

L. Paul Chew (1989) Constrained Delaunay triangulations. Algorithmica 4, pp 97–108.

J. Damen, M. van Kreveld, and B. Spaan (2008): High quality building generalization by extending the morphological operators. 11th ICA Workshop on Generalization and Multiple Representation, Montpellier, France

David Douglas & Thomas Peucker (1973): Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The Canadian Cartographer 10(2), pp 112–122.

Faragó Imre (2014): Sokrétű térképészet, Tankönyv.

http://mercator.elte.hu/~farago/sokretu_terkepeszet/ST_I_terkepeszet_alapjai_10_foldmeresi_topografiai_terkepek_2014.pdf Utolsó elérés: 2017. június 08.

H. Freeman and R. Shapira, (1975): Determining the minimum-area enclosing rectangle for arbitrary closed curve, Communication of ACM, 18, pp 175–180.

Graham, R. L. (1972): An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. Information Processing Letters 1, pp 132–133.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W. (1993): Mesh Optimization. Computer Graphics, 27 (Annual Conference Series), pp. 19-26.

Iványi Péter–Radó János (2015): Előfeldolgozás párhuzamos számításokhoz, Digitális jegyzet.

http://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0063_07_elofeldolgozas_parhuzamos/ar01s04.html Utolsó elérés: 2017. június 01.

R. A. Jarvis (1973): "On the identification of the convex hull of a finite set of points in the plane". Information Processing Letters. 2: pp 18–21.

Liu Y, Guo Q, Sun Y, Ma X (2014): A Combined Approach to Cartographic Displacement for Buildings Based on Skeleton and Improved Elastic Beam Algorithm. PLoS ONE 9(12): e113953.

https://www.researchgate.net/publication/269189206 Utolsó elérés: 2017. február 20.

B. M. Meijers, (2016): Building Simplification using Offset Curves obtained from the Straight Skeleton. Proceedings of AGILE workshop on 'Automated generalisation for on-demand mapping' and 19th ICA Workshop on Generalisation and Multiple Representation (conference paper) pp 1–8.

Monika Sester, (2000): Generalization Based On Least Squares Adjustment. International Archives of Photogrammetry and Remote Sensing, Vol. XXXIII, Part B4

Godfried Toussaint (1983): Solving geometric problems with the rotating calipers. Proceedings of IEEE MELECON ’83, Athens, Greece, May 1983

Drazen Tutić–Miljenko Lapaine (2011): Area Preserving Cartographic Line Generalization. Cartography and Geoinformation, Vol. 8, No. 11, pp 84–100.

Ungvári Zsuzsanna (2017): A térképi generalizálás vizsgálata különféle méretarány-tartományokban domborzatmodelleken. Doktori értekezés. Eötvös Loránd Tudományegyetem, pp 36–37.

M. Visvalingam and J. D. Whyatt (1993): Line Generalisation by Repeated Elimination of Points. Cartographic Journal, 30(1) pp. 46-51.

Zeshen Wang – Jean-Claude Müller (1998): Line generalization Based on Analysis of Shape Characteristics. Cartography and Geographic Information Systems, 1998/1, pp. 3-15.

## Felhasznált internetes oldalak:

ArcMap Help: http://desktop.arcgis.com/en/arcmap/10.4/tools/coverage-toolbox/an-overview-of-the-coverage-generalization-toolset.htm Utolsó elérés: 2017. május 9.

Curve Simplification Turk Project: http://web.cs.sunyit.edu/~poissad/projects/Curve/about_algorithms/reumann.php Utolsó elérés: 2017. május 20.

GRASS GIS manual: https://grass.osgeo.org/grass64/manuals/v.generalize.html Utolsó elérés: 2017. április 30.

JOSM Wiki: http://wiki.openstreetmap.org/wiki/JOSM

OCAD Inc: https://www.ocad.com/en/ Utolsó elérés: 2017. május 9.

Openstreetmap Wiki: https://wiki.openstreetmap.org/wiki/Main_Page Utolsó elérés: 2017. április 15.

Psimpl: http://psimpl.sourceforge.net/lang.html Utolsó elérés: 2017. május 20.

QGIS Python Plugins Repository: http://plugins.qgis.org/plugins/ Utolsó elérés: 2017. május 9.

## Az algoritmus létrehozásához felhasznált irodalom, és internetes tartalom

Python GDAL/OGR Cookbook

https://pcjericks.github.io/py-gdalogr-cookbook/index.html Utolsó elérés: 2017. június 07.

GDAL/OGR Python API

http://gdal.org/python/ Utolsó elérés: 2017. június 09.

Ungvári Zsuzsanna: GDAL és OGR modul telepítése Python alá Windowson, rövid bevezetés a modulok használatába

http://mercator.elte.hu/~ungvarizs/oktatas/qgis/qgis_jegyzet_19.pdf Utolsó elérés: 2017. június 7.

Zhilin Li (2006): Algorithms for Transformations of Individual Area Features**.**

In: Zhilin Li: Algorithmic foundation of Multi-Scale Spatial Representation, CRC Press, pp. 183–211.

## A felhasznált adatok forrása

OpenstreetMap**:** http://www.openstreetmap.org

Letöltve http://overpass-turbo.eu/ weboldalon keresztül.

# 10. Ábrajegyzék

Az ábrajegyzékben forrás nélkül látható képek és a mellékletek képei saját munkák.

NYILATKOZAT

Alulírott **Kocsis Viktória** (NEPTUN azonosító: **GOIPXZ**) az „Épületek generalizálása a gyakorlatban” című diplomamunka szerzője fegyelmi felelősségem tudatában kijelentem, hogy dolgozatom önálló munkám eredménye, saját szellemi termékem, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

A témavezető által elfogadott és elbírált diplomamunka elektronikus közzétételéhez (PDF formátumban a tanszéki honlapon) **hozzájárulok**.

Budapest, 2017. június 12. ……

*a hallgató aláírása*

---

Hozzájárulok a szakdolgozat benyújtásához:

Budapest, 2017. június 12. ……

*a témavezető aláírása*