TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Dmitri Krivobokov 134850

Homomeetriliste punktikogumite moodustamine ja tuvastamine kera pinnal

Bakalaureusetöö

Juhendaja: Toomas Tamm
professor,
õppeprodekaan

Kaasjuhendaja: Marko Kääramees
dotsent

2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Dmitri Krivobokov

17.05.2020

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on koostada arvutusalgoritm homomeetriliste hulktahukate otsimiseks tingimusega, et hulktahukate tipud asuvad kera pinnal, ja programmiliselt realiseerida seda algoritmi. Loodud tarkvara visualiseerib leitud homomeetriat.

Eesmärgi saavutamiseks on loodud arvutuste vähendamise algoritm, kirjutatud programmi kood PYTON keeles ja teostatud arvutused, mis näitavad homomeetria olemasolu.

Lõputöö on jaotatud kaheks loogiliseks osaks, esimeses osas analüüsitakse ja kirjeldatakse algoritmi matemaatilised aspektid ning teises osas kirjeldatakse selle algoritmi realisatsiooni.

Töö tulemusena on töötav tarkvara, mis on kättesaadav aadressil https://drive.google.com/file/d/1ntmbb0GqBTUUzQ7t_phBw-ZlG-dsZ3og/view?usp=sharing

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 32 leheküljel, 6 peatükki, 22 joonist, 1 tabelit.

Abstract

Homometry points creation and indefication sphere surface

The aim of this bachelor's thesis is to compile a computational algorithm for finding homometric polygons with the condition that the vertices of the polygons are on the surface of the sphere, and to programmatically implement this algorithm. The created software visualizes the found homometry.

To achieve the goal, an algorithm for reducing the calculations was created, the program code was written in PYTON language, and calculations showing the existence of homometry were performed.

The dissertation is divided into two logical parts, the first part analyzes and describes the mathematical aspects of the algorithm and the second part the implementation of this algorithm.

The work results in working software that is available at https://drive.google.com/file/d/1ntmbb0GqBTUUzQ7t_phBw-ZlG-dsZ3og/view?usp=sharing

The thesis is in Estonian and contains 32 pages of text, 6 chapters, 22 figures, 1 tables.

Lühendite ja mõistete sõnastik

TalTech	Tallinna Tehnikaülikool
PYTHON	Üldotstarbeline interpreteeritav programmeerimiskeel
JSON	JavaScript Object Notation, inimsõbralik struktureeritud tekstilne andmevahetuse formaat
MM	Molekulaarne modelleerimine
Homomeetria	Kahe hulga omadus, mis on väljendatud selles, et ühe hulga elementide lahutamisel üksteisest, on tulemus võrdne teise hulga elementide üksteisest lahutamise tulemusega
Platooniline keha	Korrapärane hulktahukas
Ikosaeeder	Hulktahukas, millel on 20 võrdkülgset kolmnurkset tahku ja igaühest tipust lähtub viis serva.

Sisukord

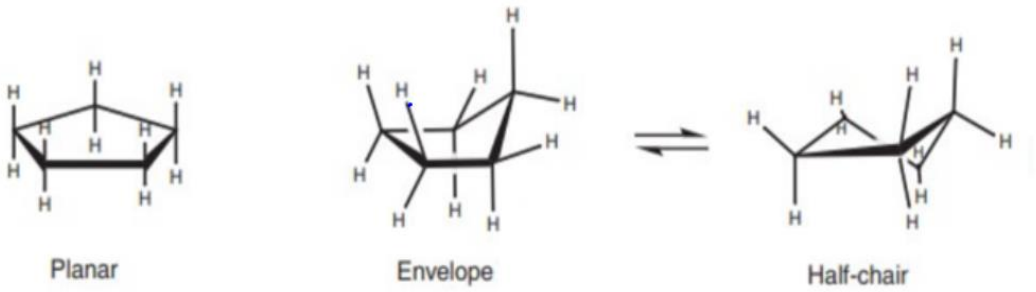
Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Bakalaureusetöö on tehtud TalTech arvutuskeemia loodusteaduskonna uurimisgrupi huvides ja seotud MM probleemi lahendamisega. See on lisamoodul molekulimudelite parametriseerimise ülesandele, kus on tarvis genereerida molekulide konformatsioone, kuid genereeritavad kujundid peavad olema unikaalsed pööramise, peegeldamise ja tippude numeratsiooni muutumise suhtes.

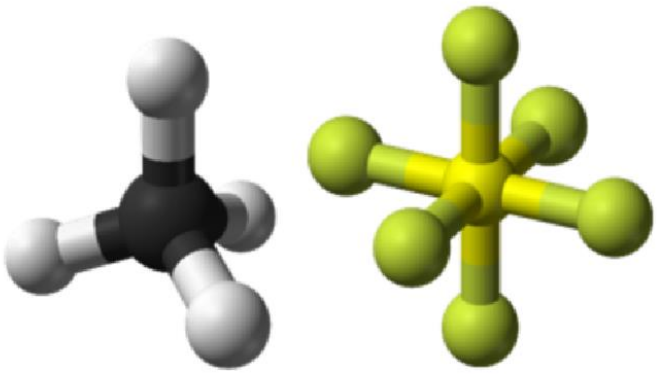
Molekuli konformatsioon on molekulis teatud konfiguratsiooniga aatomite ruumiline paigutus, mis on tingitud pöörlemisest ühe või mitme üksiku σ-sideme ümber. Erinevad konformeerid ei ole erinevad ühendid vaid ühe ja sama molekuli energeetiliselt erinevad seisundid. Joonisel 1 esitatud tsüklopentaani konformatsioonid [2].



Joonis 1 Cyclopentane konformatsioonid

TalTech arvutuskeemia loodusteaduskonna uurimisgrupp lõi algoritmi, mis genereerib süstemaatiliselt erinevatele keemiliste sidemete asukohtadele vastavad konformatsioonid ja välistab need, milles aatomivaheliste kauguste komplekt on identne mõne varem loodud konformatsiooni komplektiga. Realiseeritud algoritmi puuduseks on see, et ta jätab kõrvale ka homomeetrilised kombinatsioonid - need, milles punktide asukohad erinevad üksteisest, kuid nendevaheliste kauguste komplekt on sama. Selle puuduse ületamiseks on vaja üles otsida homomeetrilised kombinatsioonid ja selleks luua algoritm, mis väljastaks täieliku komplekti rotatsiooniliselt, reflektoorselt ja numeratsiooniliselt invariantsete asukohtade kombinatsioonidest.

MM meetodeid on iseloomustatud sellega, et nõuavad palju arvutamist ja seetõttu kasutatakse arvutamise ja visualiseerimise arvutimeetodeid. Ülalkirjeldatud eesmärgi saavutamiseks oli tarvis luua tarkvara, mis genereeriks ja visualiseeriks arvutuskeemia eesmärkidele vastavad molekuligeomeetriad ning tagaks, et homomeetrilised konfiguratsioonid tuvastatakse ja ei kõrvaldataks samaväärsete kombinatsioonide otsimisel. Programmi tingimuseks on see, et molekul on kujutatud ühe keskse aatomina ja ülejäänud 2 ... 6 aatomina, mis on võrdsel kaugusel sellest, vaata joonis 2.



Joonis 2 Molekuli võimalikud kujundid

2 Probleemi matemaatiline kirjeldus

Seoses sellega, et MM meetodid kirjeldavad molekulaarsüsteeme aatomite tasandil, võib matemaatiliselt käsitleda aatomeid punktidena. Aatomite ruumiline paigutus keskse aatomi suhtes kujutab endast matemaatiliselt punktide paigutumist kerapinnal. Aatomid võivad asetseda mistahes kohal kerapinnal, see tingib, et konformatsioonide tuvastamiseks on vaja vaadelda kogu kerapinda. Aatomitest moodustatud kujundeid võrreldakse omavahel homomeetria kaudu. Kahte erinevaid punktide struktuure S ja T nimetatakse homomeetrilisteks, kui nendel on samad punktide vahekauguste (vektorite) süsteemid [1, lk 4]

$$S - S = T - T$$

$$S = \{x_i - x_j;\ x_i, x_j \in S\}, T = \{x_i - x_j;\ x_i, x_j \in T\}$$

Kõigest sellest selgub, et matemaatiliselt ülesande võib sõnastada niiviisi, et on tarvis teada saada, kas kerapinnal võivad eksisteerida teatud punktide arvust homomeetrilised konfiguratsioonid. Ülesanne on jaotatud osadeks:

- genereerida punktikombinatsioonid kerapinnal
- eraldada genereeritud punktidest pöörete ja nihete suhtes invariantsed kombinatsioonid lähtudes punktidevahelistest kaugustest
- leida, kas järelejäänud kombinatsioonidest on homomeetrilised mõne teise genereeritud kombinatsiooniga
- kuvada leitud homomeetrilised kombinatsioonid ja nendele vastavad punktide vahekaugused

Ülalnimetatud osad on soovitatavalt eraldiseisvana kasutatavad, eelmise osa väljund on järgmise osa sisend.

3 Matemaatilised algoritmid

Seoses sellega, et tuleb kasutada meetodeid, mis töötavad suurte andmemassiividega ning teostada keerukaid arvutusi ja realiseerida seda arvutis piiratud arvutusvõimsusega, on töö elluviimiseks vaja koostada matemaatilised algoritmid, mis vähendavad arvutuste mahtu ja võimaldavad töötada ruumiliste geomeetriliste kujunditega.

Tarvis on valida või koostada algoritmid: punktikogumi mudeli moodustamiseks, invariantsete kombinatsioonide eemaldamiseks ja homomeetria leidmiseks.

3.1 Punktikogumi mudeli moodustamise meetod

Sfäär on pidev ruumiline keha, mis koosneb lõpmatust punktide hulgast, mis asuvad sfääri keskpunktist fikseeritud kaugusel. Seoses sellega, et arvutusalgoritm ei saa kasutada lõpmatut punktide hulka, on vaja piiratud arv punkte jaotada ühtlaselt kerapinnale. Punktide arvu suurenemisel nendest koosnev kujund läheneb sfäärile. On vaja leida algoritm, mis genereerib üksteisest võrdsel kaugusel asuvad punktid ja annab piisavalt väikese vea.

Algoritmi valimiseks uuriti töös „Point Picking and Distributing on the Disc and Sphere“ välja pakutud kera pinnale punktide paigutamise kolme meetodit [3]:

- elektrostaatiline nihe
- geodeetiline jaotus
- spiraalne jaotus

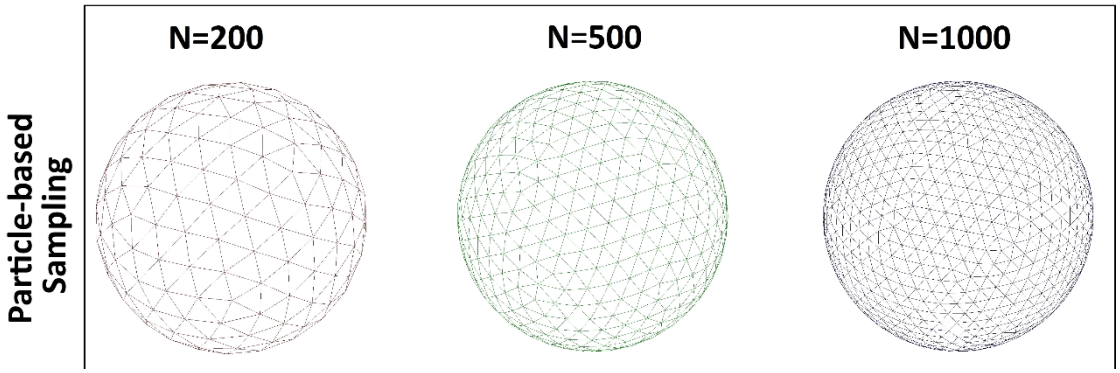
3.1.1 Elektrostaatiline meetod

Meetodi aluseks on 1904.a J. J. Thomsoni poolt välja pakutud aatomi “pudingu-mudel” (põhineb tema poolt loodud aatomimudelil). Mudelis elektronid jaotatakse "pudingust" kera pinnale nii, et Coulomb'i energia elektronide vahel oleks minimaalne. Coulomb'i

vastastikmõju on pöördvõrdeline punktilaengute vahelise kauguse ruuduga. Punktilaengute süsteem on tasakaalus, kui nendevahelised kaugused on võrdsed. Laengute väikese arvu korral süsteem on tasakaalustatud nii:

- Üksik laeng võib asuda kerapinnal ükskõik mis kohal.
- Kaks laengut peavad asuma sfääri poolustes.
- Kolm tasakaalustatud laengut paigutuvad sfääri sisse joonestatud võrdkülgse kolmnurga tippudesse.
- Neli tasakaalustatud laengut paigutuvad sfääri sisse joonestatud korrapärase tetraeedri tippudesse.
- Viis tasakaalustatud laengut paigutuvad sfääri sisse joonestatud korrapärase dipüramiidi tippudesse.
- Kuus tasakaalustatud laengut paigutuvad sfääri sisse joonestatud korrapärase oktaeedri tippudesse.
- Viis tasakaalustatud laengut paigutuvad sfääri sisse joonestatud korrapärase dipüramiidi tippudesse
- Kuus tasakaalustatud laengut paigutuvad sfääri sisse joonestatud korrapärase oktaeedri tippudesse

Seitsmendast kuni neljateistkümnenda laengu arvuni saab veel moodustada jaotust mingis sfääri sisse joonestatud püsivas geomeetrilises kujundis (isegi kui see ei ole õiges vormis). Joonisel 3 esitatud punktide jaotust elektrostaatilise nihke meetodi abil.



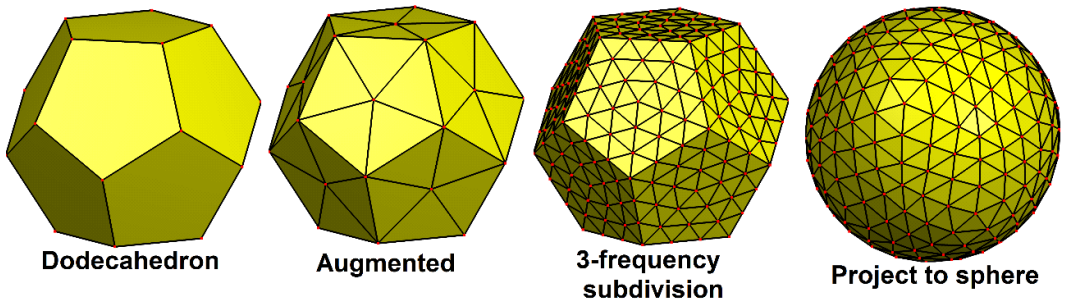
Joonis 3 Elektrostaatiline jaotus

Matemaatiline algoritm konstrueeritakse analoogselt füüsilise protsessiga, N punktid asetatakse juhuslikult kerapinnale ja seejärel muudetakse punktide asukohta, vähendades nendevahelist kauguse gradienti. Punktide arvu kasvamisega suureneb kiiresti tasakaalukonfiguratsioonide arv, mis raskendab meetodi rakendamist isegi võimsate tänapäevaste arvutite kasutamisel. Algoritmi keerukus võrdub genereeritud punktide arvu ruuduga (O (n2)).

3.1.2 Geodeetiline meetod

Meetodi põhimõte on selles, et valitakse üks sfääri sisse joonestatud korrapärane polühedron ja moodustatakse tema tahkudele mosaiik kolmnurkadest. Korrapärane polühedron on polühedron, mille tahud on võrdsed ja tema tipud võib asetada sfääri sisse, lihtsamal juhul korrapärane polühedron on platooniline keha, mis sobib hästi sfääri aproksimeerimiseks [4, lk 5].

Eksisteerivad viis platoonilist keha [4, lk 5]: korrapärane tetraeeder (4 tippu, 4 tahku), korrapärane heksaeeder (8 tippu, 6 tahku), korrapärane oktaeeder (6 tippu, 8 tahku), korrapärane dodekaeeder (20 tippu, 12 tahku), korrapärane ikosaeeder (12 tippu, 20 tahku). Joonisel 4 võib näha geodeetilise jaotuse meetodit, mis on rakendatud dodekaeedril. Platooniline keha asetatakse sfääri nii, et tipud asuvad sfääri pinnal. Punktide arvu suurendamiseks tahkudel luuakse mosaiik kolmnurkadest, mille punktid projitseeritakse kerapinnale. Algoritmi keerukus on O(N), N – punktide arv.



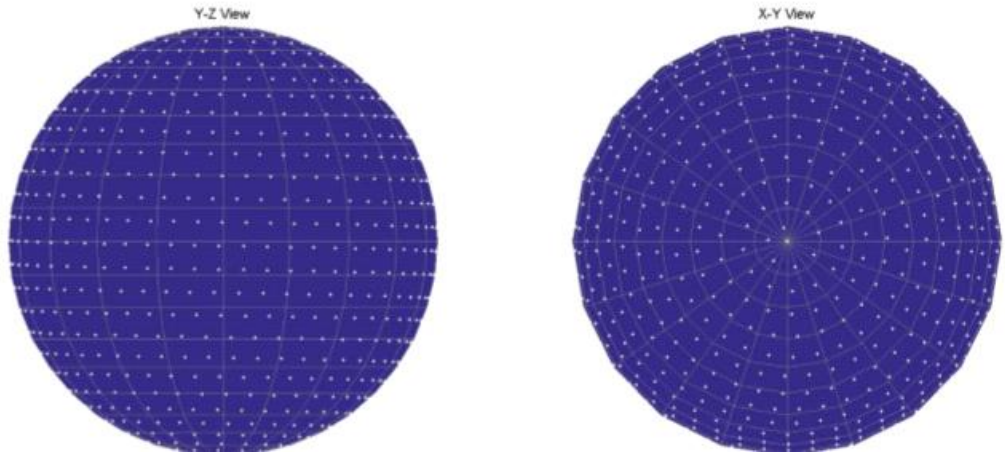
Joonis 4 Geodeetiline jaotus

3.1.3 Spiraalne meetod

Spiraalse meetodi põhimõte on selles, et kera pinnale ehitatakse spiraal. Spiraali iseloomustatakse sfääriliste koordinaatidega {θ,φ}, kus "θ ⊂ " (0,2π), φ ⊂ (0,π). Spiraali punktid ongi ühtlaselt jaotatud. Algoritmi keerukus on O(N), N – punktide arv. Töös „Point Picking and Distributing on the Disc and Sphere“ on kirjeldatud selle meetodi 4

realisatsiooni (erinevad valemid erinevate spiraalide jaoks). Joonisel 5 esitatud üks realisatsioonidest (Rakhmanov, Saff, and Zhou) [3, lk 31-38].

Meetod on piisavalt täpne 10^{-4} ühiksfääri jaoks. Täpsuse kontrollimiseks luuakse Voronoi diagramm. Iga spiraali punkti (alguspunkt) ümber luuakse lahtrid, kõik lahtris asuvad punktid on lähemal selle lahtri alguspunktile, kui teise lahtri alguspunktile. Nii moodustatakse Voronoi diagramm ja peale seda võrreldakse lahtrite pindalad ja arvutatakse standarthälve.



Joonis 5 Spiraalne jaotus

3.1.4 Valitud meetod

Varem käsitletud meetodite peamiseks probleemiks on suur punktide arv, mis nõuab palju arvutusvõimsust ja arvutuse aega. Seetõttu on vaja valida meetod, kus arvutuste arv ja andmete hulk oleks võimalikult väike. Baasmeetodiks on valitud geodeetilise jaotuse meetod, sest platoonilise keha omaduseks on sümmeetria, mille kaudu võib vähendada arvutuste mahtu. Platoonilisest kehast on valitud korrapärane ikosaeeder, sest ikosaeeder sarnaneb sfäärile rohkem kui mõni teine platooniline keha ja ikosaeedri jaoks on võimalik ilmutada sümmeetriat mitmel viisil: $E, 12C_5, 12C_3^2, 20C_3, 15C_2, i, 12S_{10}, 12S_{10}^3, 20S_6, \quad 15\sigma$ [5, lk 49]. Kasutades sümmeetriat on võimalik välja töötada algoritm, mis võimaldab genereerida punktid ikosaeedri osadest tahkudest ja ülejäänud tahkude punktid saada olemasoleva tahkude peegeldamiste ja pööramiste abil.

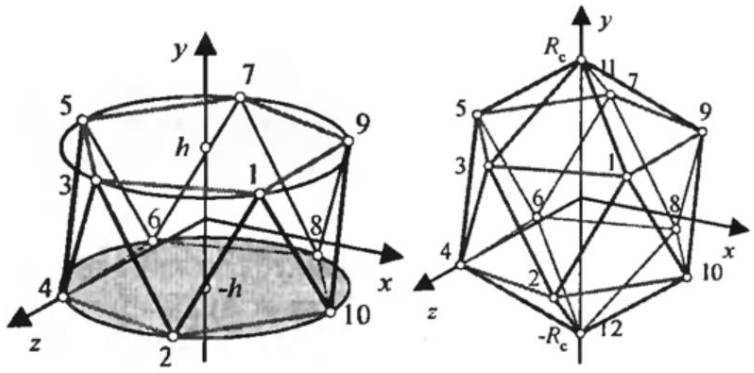
Ikosaeedri eeliseks on tahu korrapäraste kolmnurkade mosaiigiks jagamise mugavus, sest ikosaeedri tahk on ka korrapärane kolmnurk.

3.2 Arvutuste vähendamise metoodika

Arvutuste vähendamiseks on pakutud järgmine meetod. Võetakse ikosaeedri kaks paralleelset tahku, üht neist nimetame edaspidi alustahuks. Joonestatakse sirge joon läbi tahkude nii, et see läbiks ühe kolmnurkse tahu mediaanide (bisektriss) ristumispunkti ja teise (paralleelse) kolmnukrkse tahu mediaanide ristumispunkti. See joon on ikosaeedri pöördetelg. Kui me pöörame ikosaeedrit pöördetelje ümber, siis on näha, et osa ikosaeedrist luuakse pöörates selle üksikuid tahkusid teatud nurga all. Puuduvaid tahke luuakse kasutades peegelsümmeetriat enne moodustatud tahkude servade suhtes. Tahke, millega saab kogu ikosaeedri uuesti luua, nimetame N-tahkudeks. Ikosaeedri N-tahkude leidmiseks luuakse vahekauguste massiivid. Alustahu ülaosadest joonestatakse sirged jooned kõigi muude tahkude tippudeni. Sellega on saadud iga ikosaeedri tahkude kohta massiivid, mis koosnevad alustahu tippude kaugusest ülejäänud üheksateistkümne tahu tippudeni. Seejärel massiivid võrreldakse omavahel ja jäävad ainult need massiivid, kus kauguste kogum on unikaalne. Lõpptulemuseks jääb viis ainulaadset massiivi (tahku), ülejäänud neliteist võib saada pööramisega pöördetelje ümber ja peegelsümmeetria kaudu. Valitud tahkudele rakendatakse geodeetilist meetodit ja luuakse kolmnurkne koordinaatvõre. Saadud punktide massiiv on suhteliselt väike. Selline meetod tagab vajaliku täpsuse ja vähendab tunduvalt arvutuste mahtu.

3.3 Ikosaeedri tippude genereerimine

Matemaatiline mudel on koostatud töö „Построение графических примитивов Математические модели поверхностей и объектов“ järgi [6, lk 9–12]. Ikosaeedri tippusid ehitatakse kasutades muutujaid h - antiprisma poolkõrgus ja Rc - sfääri raadius, sfäär on joonestatud ikosaeedri ümber. Meetodit selgitab joonis 6.



Joonis 6 Ikosaeeder

Selleks, et leida h, ikosaeeder lõigatakse kahe paralleelse tasapinna abil nii, et tasapinnad läbiksid kahe püramiidi viisnurkseid aluseid. Tulemuseks on kaks püramiidi ja antiprisma, mille aluspinnad on korrapärased viisnurgad, mis on pööratud üksteise suhtes 36° nurga all.

Ikosaeeder on asetatud Descartes'i koordinaatsüsteemi nii, et Y-telg läbib ikosaedri vastasseisvad tipud, X-telg jagab antiprisma aluse vahekauguse võrdselt ja läbib antiprisma kolmnurkse tahu (joonisel 6 kolmnurk tippudega 1-9-10) mediaanide ristumiskohta, Z-telg on ristsirge X-teljega ja läbib X ja Y-telgede ristumiskohta. Antiprisma viisnurkse aluspinna ümberringjoone raadiuse r kaudu leitakse h ja Rc.

Esiteks arvutatakse viisnurga serva pikkus

$$d = \sqrt{(x_9 - x_1) + (y_9 - y_1) + (z_9 - z_1)}$$

$$x_9 = x_1, \quad y_9 = y_1, \quad z_9 = z_1 + 2r * \sin 36° \rightarrow d = 2r * \sin 36°$$

$$d = \sqrt{(x_{10} - x_1) + (y_{10} - y_1) + (z_{10} - z_1)}$$

$$d = \sqrt{r^2(\cos 360° - cos36°)^2 + 4h^2 + r^2 (\sin 360° - sin36°)^2}$$

Järgmisena arvutatakse antiprisma poolkõrgus h

$$h = \frac{1}{2}\sqrt{d^2 - (\cos 360° - cos36°)^2 - (\sin 360° - sin36°)^2}$$

$$h = \frac{r}{2}\sqrt{4\, sin^2 36° - (1 - cos36°)^2 - (0 - sin36°)^2}$$

$$h = \frac{r}{2}\sqrt{4\, sin^2 36° - 1 + 2 * cos36° - cos^2 36° - sin^2 36°}$$

$$h = \frac{r}{2}\sqrt{4\, sin^2 36° - 2 + 2 * cos36°}$$

Seejärel leitakse raadius Rc

$$Rc = \sqrt{x_1^2 + y_1^2 + z_1^2}$$

$$Rc = \sqrt{r^2 * \cos^2 36 + h^2 + r^2 * \sin^2 36}$$

$$Rc = \sqrt{r^2 + h^2}$$

Seejärel ikosaeedri tippudele omistatakse koordinaadid. Kõigepealt määratakse Y-teljel ikosaeedri vastasseisvad tipud ehk püramiidide ülaosad, punkt 11 koordinaat (0, Rc, 0) ja punkt 12 koordinaat (0, -Rc, 0). Järgmisena tekitatakse loodud tippude naabertipud (punkt 11, 12). Antud juhul on need viisnurkade tipud, mis arvutatakse kahe tsüklilise N/5N rühma abil vastavalt järgmistele valemitele:

- Ülemise püramiidi aluse tippude koordinaadid. Need punktid moodustavad esimese tsüklilise rühma, joonisel 6 punktid 1, 3, 5, 7, 9.

$$p1(k) = \begin{cases} x = r * (\cos 36° + k * \cos 72°), & k \in 0 \dots 4 \\ y = h & \\ z = r * (\sin 36° + k * \sin 72°), & k \in 0 \dots 4 \end{cases}$$

- Alumise püramiidi aluse tippude koordinaadid. Need punktid moodustavad teise tsüklilise rühma, joonisel 6 punktid 2, 4, 6, 8, 10.

$$p2(k) = \begin{cases} x = r * k * \cos 72°, & k \in 0 \dots 4 \\ y = -h & \\ z = r * k * \sin 72°, & k \in 0 \dots 4 \end{cases}$$

Kõik moodustatud punktid ühendatakse üksteisega lihtgraafi kaudu. Selleks naaberpunktid ehk sama serva lõpupunktid seostatakse omavahel nii:

- esimese tsüklilise rühma iga punkt p1(k) on seotud punktidega: Rc, $p1(k+1)$, $p1(k-1)$, $\mathrm{p2(k)}$, $\mathrm{p2(k-1)}$.
- teise tsüklilise rühma iga punkt p2 (k) on seotud punktidega: $-\mathrm{Rc}$, $\mathrm{p2(k-1)}$, $\mathrm{p2(k+1)}$, $\mathrm{p1(k)}$, $\mathrm{p1(k-1)}$.

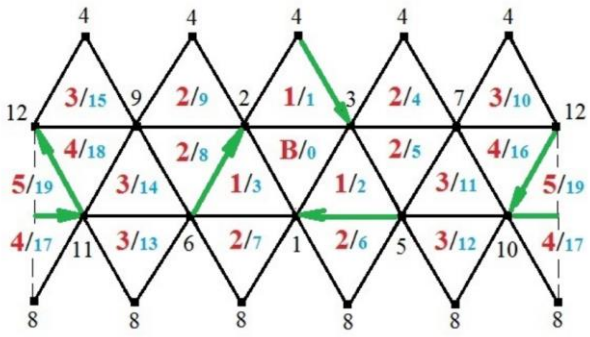
Viimasena lihtgraafi punktid ühendatakse ikosaeedri tahkudeks. Selleks kasutatakse Breadth-first search (BFS) algoritmi ehk laiuti otsingut. Otsingus osalevad alguspunkt ja sellest esimese ja teise taseme punktid. Kui teise taseme naabriks on jälle alguspunkt, siis kõik ühe tahu punktid on leitud. Iga järgmine ikosaeedri tipp määratakse alguspunktiks. Otsing teostatakse järjestatult iga alguspunkti jaoks.

3.4 Sümmeetriarühmade otsing

Sümmeetriarühmade otsing on vajalik selleks, et eemaldada arvutustest need tahud, mis on sümmeetrilised üksteise suhtes. On vaja leida kõik sümmeetria variandid, mida on mitu tegelikult korduvat, ühesugust varianti. Selleks valitakse üks suvaline tahk (edaspidi alustahk) ja leitakse selle suhtes kõik sümmeetrilised tahud. Ikosaeeder omab 20 sümmeetria operatsiooni S_6, need sümmeetria teljed läbivad kahe ikosaeedri tahu keskpunktid [5, lk 49]. Neid sisaldavad: C_3 - pöördesümmeetria (tahu pöörlemine ümber ikosaeedri sümmeetriatelje) [5, lk 23-26], σ - peegelsümmeetria (tahkude sümmeetria ühise tasandi suhtes) [5, lk 18-21] ja nende kombineeritud variant (S_6 - pöörd-peegeldussümmeetria [5, lk 27-29]). Sellest järeldub, et iga tahu võib leida läbi alustahu ja ühe tahu igast sümmeetriarühmast pööramise, pöörlemise või peegelduse kaudu.

Ikosaeedri tahkusid ja nende tippusid suunatakse ehk sorteeritakse vastavalt sümmeetriarühmadele. Selleks ikosaeedri tahud kirjeldatakse tippude koordinaatide loendina niiviisi: alustahu (i = 0) koordinaatide komplekt - $p_{10} = (X_{10}, Y_{10}, Z_{10})$, $p_{20} = (X_{20}, Y_{20}, Z_{20})$, $p_{30} = (X_{30}, Y_{30}, Z_{30})$; esimese tahu (i = 1) komplekt $p_{11} = (X_{11}, Y_{11}, Z_{11})$, $p_{21} = (X_{21}, Y_{21}, Z_{21})$, $p_{31} = (X_{31}, Y_{31}, Z_{31})$; jne, kokku 20 tahku. Seejärel leitakse sümmeetriarühma numbri ja iga tahu vastavus alustahu suhtes ning tahud sorteeritakse sümmeetriarühma numbri järgi.

Sümmeetriarühm on tahkude kogum, mille kaugus alustahust on sama. Sümmeetriarühmadest arusaamiseks vaadeldakse ikosaeedri pinnalaotust, mis on toodud joonisel 7. Sümmeetriarühmad on märgitud punase värviga, sorteeritud tahud - sinise värviga, punktid - musta värviga. Alustahk on paigutatud laotuse keskele ja sellele on omistatud number 0.

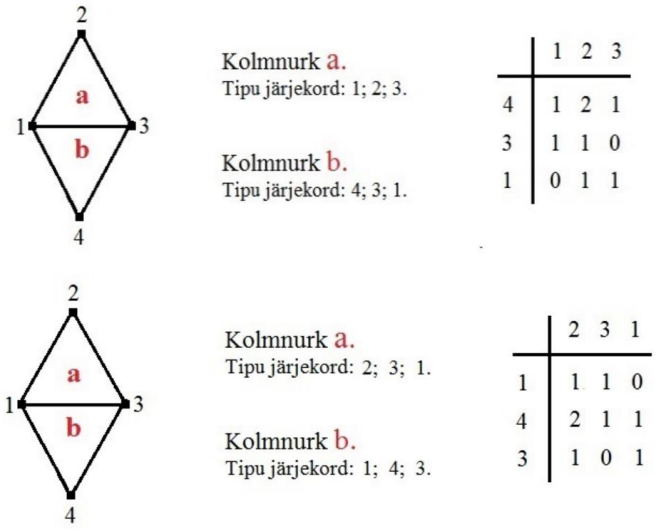


Joonis 7 Ikosaeedri laotus

On näha, et esimene tahk on alustahu peegelsümmeetria läbi ühise serva, teine tahk on esimese tahu peegelsümmeetria läbi ühise serva ehk alustahu topelt peegelsümmeetria. Alustahu peegeldamisega saab ehitada ülejäänud tahkusid ja peegelduste maksimaalne arv on viis. Siit selgub, et ikosaeeder on iseloomustatud viie sümmeetriarühmaga. Iga ikosaedri tahk kuulub mingile sümmeetriarühmale. Igast sümmeetriarühmast ühe suvalise tahu nimetame N-tahuks, kus N=1..5. Komplekt N-tahkudest ja alustahk taastab terve ikosaedri.

Tahud sorteeritakse sümmeetriarühmadeks enne koostatud lihtgraafi punktidest (vaata lõige 3.3). Alustahk on juba valitud lihtfraafi koostamisel. Alustahu ja selle ühise servaga esimese sümmeetriarühma kuuluva tahu jaoks kehtib järgmine väide: kahe sama serva tippude jaoks saab leida kaks ühist naabrit, mis ei ole samas servas. Kolm punkti on alustahu tipud, kuid neljas punkt on teise tahu tipp. See punkt ja serva lõpupunktid moodustavad esimese sümmeetriarühma tahu. Selle reegli järgi valitakse esimese sümmeetriarühma tahud. Teise sümmeetriarühma tahud valitakse esimese sümmeetriarühma tahkude tippude järgi jne, kuni neljanda sümmeetriarühmani.

Tahkude orienteerimine üksteise suhtes teostatakse tahkude tippudevaheliste kauguste võrdluse kaudu. Selleks koostatakse kauguste maatriksid, kus ridadeks on ühe tahu tipud, veergudeks on teise tahu tipud ja lahtriteks on nendevaheline minimaalne servade arv ehk tippudevaheline kaugus. Joonisel 8 on toodud kaks varianti tippudevahelistest kombinatsioonidest, kuid kombinatsioonide arv kahe tahu jaoks on 36.



Joonis 8 Esimese sümmeetriarühma kaugusmaatriksid

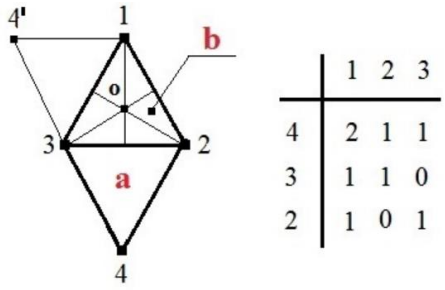
Maatriksid on vaja maksimaalselt orienteerida, see tähendab, et need peavad vastama tingimustele:

- kauguste summa ridades on järjestatud kahanevas järjekorras vasakult paremale
- kauguste summa veergudes on järjestatud kahanevas järjekorras ülevalt alla

Maatriksid on vaja koostada nii, et üks tahkudest oleks alustahk. Alustahu punktid siin ja edaspidi asetatakse maatriksi veergudesse.

Kui maksimaalne orientatsioon puudub, siis kasutatakse osalist orientatsiooni, siis järjestatakse ainult ainult teiste tahkude punktid, mitte alustahu punktid, antud juhul maatriksi read. Kui maatriksite ridades on võrdsed summad, siis need sorteeritakse tippude numbri järgi vasakult paremale.

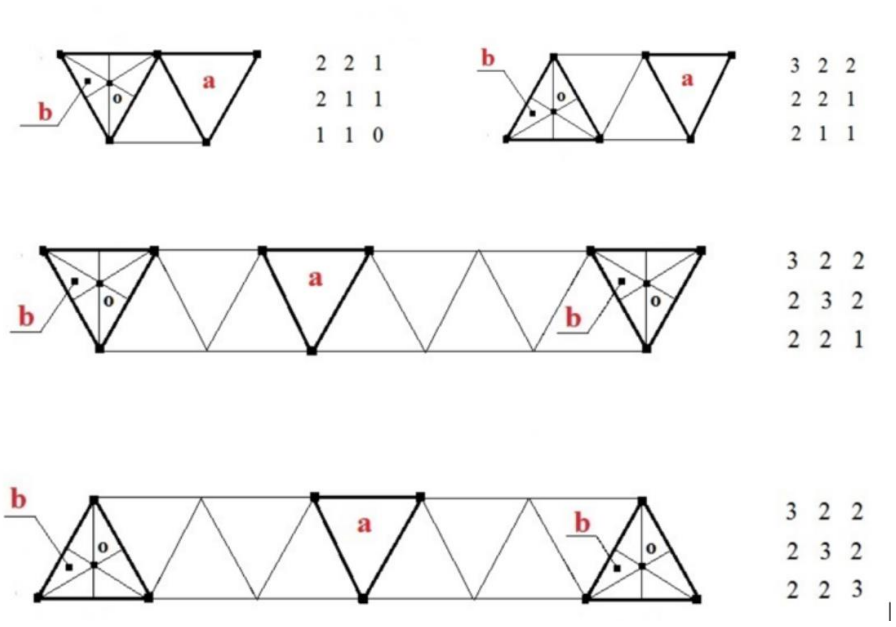
Joonisel 9 on toodud osalise orienteerimise näide. On näha et alustahu (1-2-3) peegeldamise läbi serva 1-3 tulemuseks on kolmnurk 1-3-4'. Tahule 1-3-4' ei saa koostada maksimaalselt orienteeritud maatriksit. Enne on tarvis pöörata seda tahku 120° ümber ristsirge telje, mis läbib alustahu mediaanide ristumiskohta. Nii võib tahku 1-3-4' orienteerida vastavalt N-tahule 2-3-4.



	1	2	3
4	2	1	1
3	1	1	0
2	1	0	1

Joonis 9 Maksimaalselt orienteeritud kaugusmaatriks

Sama meetodit on rakendatud ülejäänud sümmeetriarühmades kaugusmaatriksite orienteerimiseks. Joonisel 10 on toodud maksimaalselt orienteeritud erinevate sümmeetriarühmade maatriksid



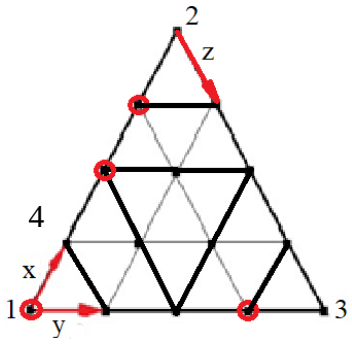
Joonis 10 2-5 sümmeetriarühma kaugusmaatriksid

Kui kahe tahu vahel eksisteerib maksimaalne või üks osaline orientatsioon, siis tahuc suunatakse selle järgi. Kui kahe tahu jaoks võib leida mitu orientatsiooni, siis ei saa osaliselt orienteerida tahkusid ja neid on vaja "pöörata" kasutades suundvektorit nii, et suunavektorid koos läbiksid täisringi (joonis 7, rohelised nooled).

Tahkude sümmeetria rühmiti jaotuse tulemuseks on see, et terve ikosaeedri asemel osalevad arvutustes ainult kuus tahku: alustahk ja viis N-tahku.

3.5 Kolmnurkse mosaiigi ehitamine

Ikosaeedri tahk on võrdkülgne kolmnurk ja mosaiigi ehitamiseks kolmnurksed tahud jagatakse mitmeks subkolmnurgaks vastavalt etteantud jaotuste arvule. Jaotamine toimub nii: kolmnurga küljed jagatakse pooleks ja uusi punkte kasutades ehitatakse algkolmnurga sees neli identset subkolmnurka. Neid võib vaadelda kui erinevalt suunatud sama kolmnurga variante. Subkolmnurkade servad jagatakse jälle pooleks ja neid punkte kasutades, iga subkolmnurga sees, ehitatakse veel neli kolmnurka jne. Mosaiigi kõik punktid ühendatakse omavahel niiviisi, et ükssama punkt ei tohi osaleda mitmes subkolmnurgas korraga. Selle tõttu tulemuseks võib olla, mitte ainult subkolmnurkade hulk, vaid ka üksikpunktide hulk, sest keskmine kolmnurk võib sisaldada endas rohkem või vähem punkte kui ülejäänud kolmnurgad. Joonis 11 kuvab mosaiigi genereerimist ja ühikvektorite paigutust mosaiigis.



Joonis 11 Ühikvektorid kolmnurga mosaiigis

Mosaiigi punktid väljendatakse oma indeksiga. Indeksid arvutatakse ühikvektorite kaudu (ühik on minimaalne kaugus kahe mosaiigi punkti vahel) ja nad näitavad mitu ühikvektorit on vaja läbida tahu alguspunktist vaadeldava punktini. Indeksi arvutuseks on tarvis teada algkolmnurga alguspunkti, mis on tahu üks kindel tipp ning tahu serva jaotuste arv. Ühikvektorid on suunatud algkolmnurga servade järgi. Algkolmnurga servade pikkus võrdub ühikvektori pikkusega korrutatud jaotuste arvuga.

- kui sub- ja algkolmnurgad on samasuunalised, siis esimese kolme subkolmnurga alguspunktid võib väljendata järgmises vormis:

$$\begin{cases} p1 = p0 \\ p2 = p0 + n//2 \ * \boldsymbol{e_{12}} \\ p3 = p0 + n//2 * \boldsymbol{e_{13}} \end{cases}$$

  p1, p2, p3 - kolme võrdselt orienteeritud subkolmnurkade alguspunktid
  n //2n -subkolmnurga serva jaotuste arv, ümardatud allapoole
  p0 – algkolmnurga alguspunkt

- kui sub- ja algkolmnurgad on vastassuunas, siis subkolmnurga alguspunkt

$$\mathrm{p4} = \mathrm{p0} + \mathrm{n} // 2 * \boldsymbol{e_{12}} - ((\mathrm{n} + 1) \bmod 2) * (\boldsymbol{e_{12}} - \boldsymbol{e_{13}})$$

Iga punkti võib väljendada indeksiga [S_{e12}, S_{e13}, S_{e23}], kus S – ühikvektorite kogus. Juhul, kui kolmnurgad on samasuunalised, siis indeksis suurendatakse muutujaid S_{e12}, S_{e13}, vaid S_{e23} jääb samaks. Kui kolmnurgad on vastassuunalised, siis indeksis suurendatakse muutujaid S_{e12}, S_{e13} ning S_{e12}, S_{e23} vahetavad oma kohad indeksis, sest subkolmnurgad on vastassuunalised.

Tahu jaotus ja indeksite arvutus toimub üheaegselt, arvutus on rekursiivne. Rekursiivsus lõpeb kui subkolmnurga jaotuste arv n on üks või kaks. Rekursiivsuse viimased punktid:

- kui n = 1, siis punkt on võrdne tahu alguspuntiga.
- kui n = 2, siis punktid on p0, p0 + $\boldsymbol{e_{13}}$, p0 + ($\boldsymbol{e_{12}}$ või $\boldsymbol{e_{23}}$), kus $\boldsymbol{e_{23}}$ vastassuunaliste kolmnurkade ja $\boldsymbol{e_{12}}$ samasuunaliste kolmnurkade puhul.

Indeksid teisendatakse [S_{e12}, S_{e13}, S_{e23}] väärtusest [S_{e12}, S_{e13}] väärtusele nii: [$S_{e12} - S_{e23}$, $S_{e13} + S_{e23}$].

3.6 Unikaalsete tahkude komplektide otsing

Homomeetriat otsitakse kujundite vahel tingimusega, et kujunditel on sama tippude arv ja tipud asetatakse erinevatele tahkudele. Tippude arv antakse parameetrina ette ja see arv määrab tahkude kogust, mis kuulub unikaalsete tahkude komplekti.

Unikaalne tahkude komplekt sisaldab alati alustahku, ülejäänud tahud peavad olema sorteeritud alustahust kauguste järgi. Unikaalsesse tahkude komplekti kuuluvad need tahud, millest saab koostada kaugusmaatriksi, mida ei või saada teiste maatriksite veergude või ridade ümbertõstmisega. Kaugusmaatriksid koostatakse kõigi tahkude jaoks.

Unikaalsete tahkude otsing teostatakse selleks, et eraldada pöörete ja nihete suhtes invariantsed kombinatsioonid lähtudes kauguste kriteeriumist. Kauguste kriteeriumiks on tahkude läbimise arv, mis on tarvis selleks, et jõuda alustahust vajaliku tahuni tingimusega, et naabertahkudes on ühine serv. Kauguste kriteeriumit iseloomustatakse tahu sümmeetriarühma numbrina, kui üks tahkudest on alustahk.

Unikaalsete tahkude komplekti moodustamiseks on vaja leida tahud, millevaheliste kaugusmaatriksite komplekt on unikaalne. Selleks koostatakse kauguste maatriksid kasutades meetodit, mis on kirjeldatud lõigus 3.4. Igale maatriksile omistatakse identifikatsiooni number ehk *hashcode*, mis koostatakse valemi järgi:

$$hashcode = \sum_{j=0}^{n} \left(\sum_{i=0}^{n} a_{ij} * n^i \right) * n^j, \qquad [a_{ij}] = sort([b_{ij}]),$$

$$b_{ij} - kaugusmaatriks$$

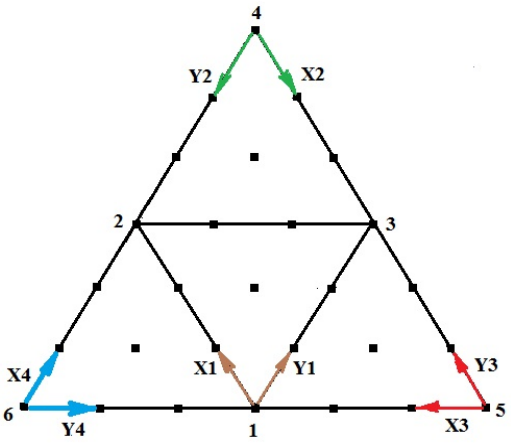
Juhul, kui erinevates maatriksites on sama *hash code,* siis maatrikseid võrreldakse jälle niiviisi, et ühe maatriksi elemendid jäävad samaks, kuid teises maatriksis read vahetavad oma järjekorda, ka veerud vahetavad oma järjekorda. Kui mingi teise tahu maatriksi permutatsioon on võrdne esimese tahu maatriksiga, siis jääb ainult üks tahkudest, sest teine on pööratud või nihutatud esimene tahk.Niiviisi eemaldatakse dublikaadid ja unikaalne tahkude komplekt on leitud

3.7 Homomeetria otsing

Homomeetria leidmiseks võrreldakse kahe punkti vahekauguseid, punktid asuvad erinevates tahkudes, tahud kuuluvad unikaalsete tahkude komplekti. Unikaalses tahkude komplektis on alati olemas alustahk. Ülejäänud tahud on suvalised ikosaeedri tahud. N-tahud võetakse sümmeetriarühmadeks sorteeritud tahkudest. N-tahu ja alustahu punktidevahelised kaugused on salvestatud maatriksitele ja väljendatakse indeksitega.

Uuritavate kujundite minimaalne punktidevaheline kaugus antakse ette programmi käivitamisega ja see kaugus on suurem kui tahu serva pikkus. Seetõttu võib eksisteerida niisugune punktide paar, mille vahekaugus on väiksem minimaalsest kaugusest, ja seda paari ei võeta arvesse.

Kui unikaalsete tahkude komplekti ei kuulu valitud N-tahud, siis unikaalseid tahkusid on tarvis orienteerida vastavalt N-tahkudele. Uurime seda protsessi alustahu ja esimese sümmeetriarühma tahkude näitel. Joonisel 12 on toodud alustahk 1-2-3, N-tahk 4-3-2 ja suvaline tahk 5-1-3, mis ei kuulu N-tahkude hulka. On vaja leida vastavust 5-1-3 tahu suvalise punkti ja alustahu suvalise punkti vahel.



Joonis 12 12Alustahu ja esimese sümmeetriarühma tahude ühikvektorid erinevates koordinaatsüsteemides

Selleks koostatakse nendevaheline kaugusmaatriks, mis on esitatud joonisel 13.

	1	2	3
5	1	2	1
3	1	1	0
1	0	1	1

	3	1	2
4	1	2	1
3	0	1	1
2	1	1	0

Joonis 13 Kaugusmaatriksid punkt 2 jaoks erinevates koordinaatsüsteemides

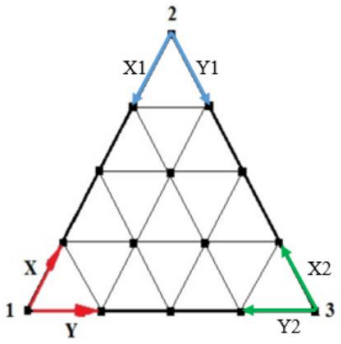
See maaktriks on ekvivalentne osaliselt orienteeritud maatriksiga N-tahu ja alustahu vahel. Siit järjeldub, et 5-1-3 tahu ja alustahu punktide paari võib asendada N-tahu ja alustahu punktide paariga. Kõik maksimaalselt orienteeritud maatriksid alustahu ja N-tahkude vahel on enne koostatud ja ainult on tarvis nendest valida sobiv variant.

Punktid väljendatakse indeksina, see tähendab, et on vaja transformeerida mitte N-tahkude indeksid vastavuses N-tahkude indeksitega. Selleks on vaja teisendada tipupunktide indeksid paari kuuluvatest tahkudest. Tahu tippude indeksid on ekvivalentsed koordinaatidega, mis on määratud muudetud Cartesiuse koordinaatsüsteemis, kus X-telg läbib punkte 1, 2; Y-telg läbib punkte 1, 3 ning X-Y vaheline nurk on 60º, nii nagu on kuvatud joonisel 14.

Punktide indeksid on määratud ühikvektorite arvuna (x, y), mis on tarvis lisada üksteisele alates alguspunktist ja seega saadakse igal tahul asuvad punktid. Vastavalt sellele, mis tippudest on valitud koordinaatsüsteemi alguspunkt, võib sama punkti väljendada erinevate indeksite variantide kaudu. Indeksite kohandamine N-tahkude

indeksiteks toimub koordinaattelgede asukoha ning ühikvektorite suuna muudatustega. Indeksite muutmise protsess toimub kahte moodi, mis on selgitatud joonise 12 abil:

- koordinaatsüsteemi telgede suundade muutmine
- koordinaatsüsteemi alguspunkti muutmine



Joonis 144 Ühikvektorite paigutus tahu servadel

Kui võtta alguspunktina, punkt 1 ja ühikvektorid (x, y), siis punkti 2 koordinaadid väljendatakse alguspunkti koordinaatidest pluss n ühikvektorit X-telje järgi, mis meie töös vastab indeksile (4, 0); punkti 3 koordinaadid on alguspunkti koordinaadid pluss n ühikvektorit Y-telje järgi, mis meie töös vastab indeksile (0, 4). Kui koordinaatteljed on vahetatud (X muutub Y-ks ja Y muutub X-ks), siis vahetatakse omavahel ka liikmeid indeksites ja indeks on muutunud (0, 4)-ks punkti 2 jaoks ja (4, 0)-ks punkti 3 jaoks.

Kui koordinaatsüsteemi alguspunktina on punkt 2, siis punkte väljendatakse ühikvektorites (x1, y1), mis on koordinaatsüsteemis (x,y) väljendatud (-1, 0), (-1, 1) kaudu. Kui alguspunktina on punkt 3, siis samad punktid väljendatakse ühikvektorite (x2, y2) kaudu, mis on koordinaatsüsteemis (x, y) väljendatud (1,-1), (0, -1) kaudu.

Selleks, et kohandada tahkude, mis ei kuulu N-tahude hulka, indeksid N-tahkude indeksiteks, on esiteks vaja leida esialgse koordinaatsüsteemi alguspuntk x_0, y_0; teiseks väljendada alguspunkti uues koordinaatsüsteemis; seejärel väljendada vana koordinaatsüsteemi ühikvektorid x, y vastavalt uue koordinaatsüsteemi ühikvektoritele e_{xnew}, e_{ynew} ja lõpuks arvutada punktide indeksid uues koodrinaatsüsteemis x_{new}, y_{new} kasutades valemit

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} \overrightarrow{e_{xnew}} \\ \overrightarrow{e_{ynew}} \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\text{Näiteks } \begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix} + \begin{bmatrix} 0 & -1 \\ 1 & -1 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix}$$

Uute indeksite järgi arvutatakse punktidevahelised kaugused, kõik kujundi tippudevahelised kaugused koondatakse homomeetria tunnuseks ja olukord, kui kujundid on erinevad, kuid tunnus on sama, näitab homomeetria olemasolu.

4 Tehniline lahendus

Selles lõigus on kirjeldatud programmi nõuded, valitud programmeerimise keskkond ja loodud tarkvara loogika ja struktuur.

4.1 Funktsionaalsed ja mittefunktsionaalsed nõuded programmile

Funktsionaalsed nõuded vastavad küsimusele, mida peab tegema programm ja määravad, mis funktsionaalsust peab omama programm oodatud eesmärgi saavutamiseks. Mittefunktsionaalsed nõuded näitavad programmi töötamise välised tingimused, kitsendused, kvaliteedi ootused, arenguvõimalused, täitmisomadused jne.

4.1.1 Funktsionaalsed nõuded

Lähtudes ülesande püstitusest luuakse tarkvara, mille põhiülesandeks on teostada arvutusi, mis võimaldavad:

- genereerida punktikombinatsioonid kera pinnal (mudel)
- eemaldada punktidest invariantsed kombinatsioonid lähtudes punktidevahelistest kaugustest
- tuvastada homomeetrilised konfiguratsioonid
- visualiseerida loodud mudel ja tuvastatud homomeetria
- luua väljund leitud homomeetriast ja homomeetriliste kujundite punktide vahekaugustest

4.1.2 Mittefunktsionaalsed nõuded

Loodav tarkvara ei vaja graafilist kasutajaliidest ega integreerimist teise süsteemi, sest peaülesandeks on teostada arvutusi. Sellest tuleneb suhteliselt kitsas mittefunktsionaalsete nõuete loetelu:

- omab laienduse võimalust (moodulite lisamine)
- saab töötada erinevalt platvormilt (Windows, Linux)
- võimaldab monitoorida protsesse ja teostada testimist
- lihtne tarkvara käivitamine
- tagada stabiilsed tulemused
- programmid, milles on kirjutatud tarkvara, peavad olema tasuta või kättesaadavad TalTech-i serveritest
- tarkvara peab olema mugav installimisel ja ei võta palju arvuti mälu

4.2 Programmeerimise keele valik

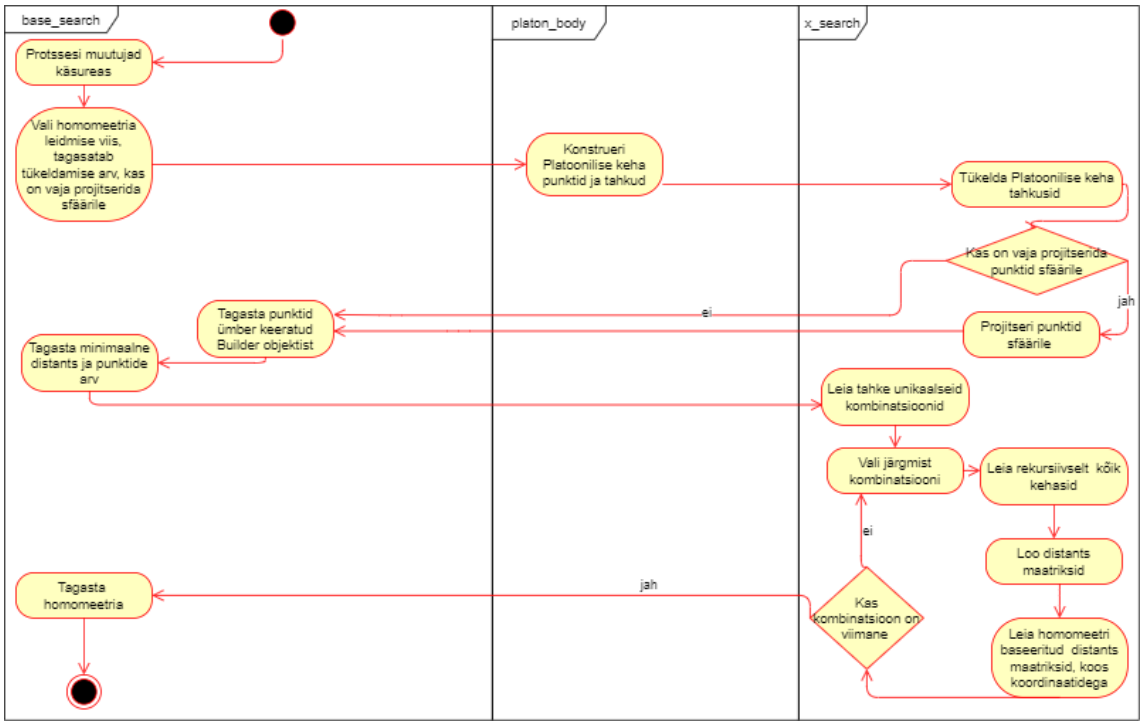
Programmeerimiskeele valik on tingitud nõuetest, et see peab olema vabalt kättesaadav, tagama modulaarsust ja seda võib käivitada erinevates platvormides. PYTON-il on kõik need omadused. PYTON levitatakse tasuta Python Software Foundation litsentsi alusel, mis võimaldab kasutada seda tasuta ja piiranguteta. PYTON on hästi adapteeritav ja töötab peaaegu kõigil tuntud platvormidel. PYTON võimaldab lisada teistes keeltes kirjutatud mooduleid. Selle omaduse tõttu PYTON on üks populaarsemaid programmeerimiskeeli teadlaste arvutustes, sest PYTON-isse on ehitatud teegid, mida on lihtne käivitada PYTON keskkonnas, kuid sisuliselt on teekides keerukamad ning võimsamate keelte programmid [8].

PYTON-i keel on objektorienteeritud programmeerimiskeel, mille omaduseks on modulaarsus ja mastaapsus. Seetõttu on kirjutatud programmide lähtekoodid arusaadavad ja loetavad. PYTON-i sünktaksis võrreldes teiste objektorienteeritud keeltega on lihtne ja mugav õppimiseks. PYTON on aktiivselt arenev keel, versioonid antakse välja umbes iga kahe ja poole aasta tagant. PYTON-i miinuseks on suhteliselt madal kiirus. See võiks olla suureks probleemiks, sest lõputöö tarkvara põhitegevuseks on suur andmete massiivide töötlus. Selle probleemi vältimiseks kasutatud NumPy teek, mis on loodud programmeerimiskeeles C (C keeles on suur kiirus, kuid raske süntaksis) ja ettenähtud mitmemõõtmeliste massiivide töötlemiseks, mis võimaldab saavutada arvutuste jõudlust, mis on võrreldav spetsiaalsete pakettidega [10].

PYTON-il on ka olemas PlotLy teek, mis on ettenähtud graafikute, diagrammide ja 3D mudelite loomisel [9]. Selle võimaluse olemasolu põhjendab ka seda, miks PYTON oli valitud baaskeeleks tarkvara arendamisel.

4.3 Tarkvara jaotus mooduliteks

Tarkvara funktsionaalsus on jagatud kolmeks rühmaks: Base_search (peamoodul kust kutsutakse välja ülejäänud moodulid ja koostatakse väljund), platon_body (platoonilise keha tippude genereerimine), x_search (kõik ülejäänud arvutused). Joonis 15 näitab nendevahelised seosed.



Joonis 15 Tegevuste diagramm

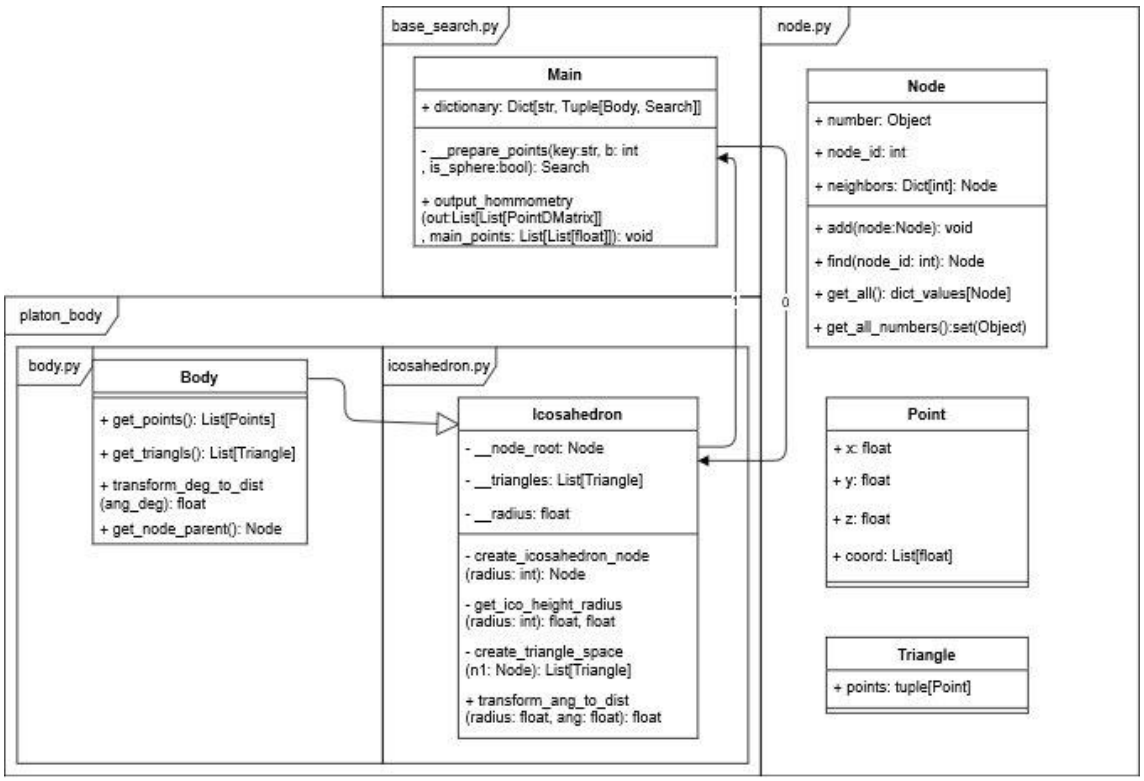
4.3.1 BASE_SEARCH moodul

See moodul vastutab teiste moodulite käivitamise järjekorra eest. Sisendiks on kasutaja poolt või vaikimisi määratud parameetrid, mis on kirjeldatud lõigus 4.4. Moodulid käivitatakse automaatselt üksteise järgi, iga eelmise mooduli väljund on järgmise mooduli sisend. Lõppväljundiks JSON fail, kus kujundid on koondatud homomeetriliseks komplektiks. Visualiseeritud 3D mudel, mis kuvab homomeetrilised kujundid, realiseeritud moodulis visualisate_hom.py.

4.3.2 PLATON_BODY moodul

Kasutatakse platoonilise keha tippude genereerimiseks. Joonisel 16 on toodud PLATON_BODY mooduli kontseptuaalne mudel. Realiseeritud kahes moodulis icosahedron.py ja node.py.

Node.py on universaalne moodul, mis sisaldab andmed, mida on tarvis objekti kirjeldamiseks. Node on globaalne objektne struktuur. Selles moodulis Node objekt on lihtgraafi punkt. Icosahedron.py on moodul ikosaeedri genereerimiseks. Valemid ja loogiline jaotus on kirjeldatud lõigus 3.3 ja realiseeritud meetodis get_ico_height_radius(radius), mis arvutab abimuutujad h ning Rc, create_icosahedron_node(radius), mis genereerib tippude koordinaadid, kus vaikimisi radius=1, create_icosahedron_node(radius) ühendab tipud omavahel lihtgraafiks ja moodustab Node objektid, create_triangle_space (n1) jaotab lihtgraafi punktid tahkudeks.

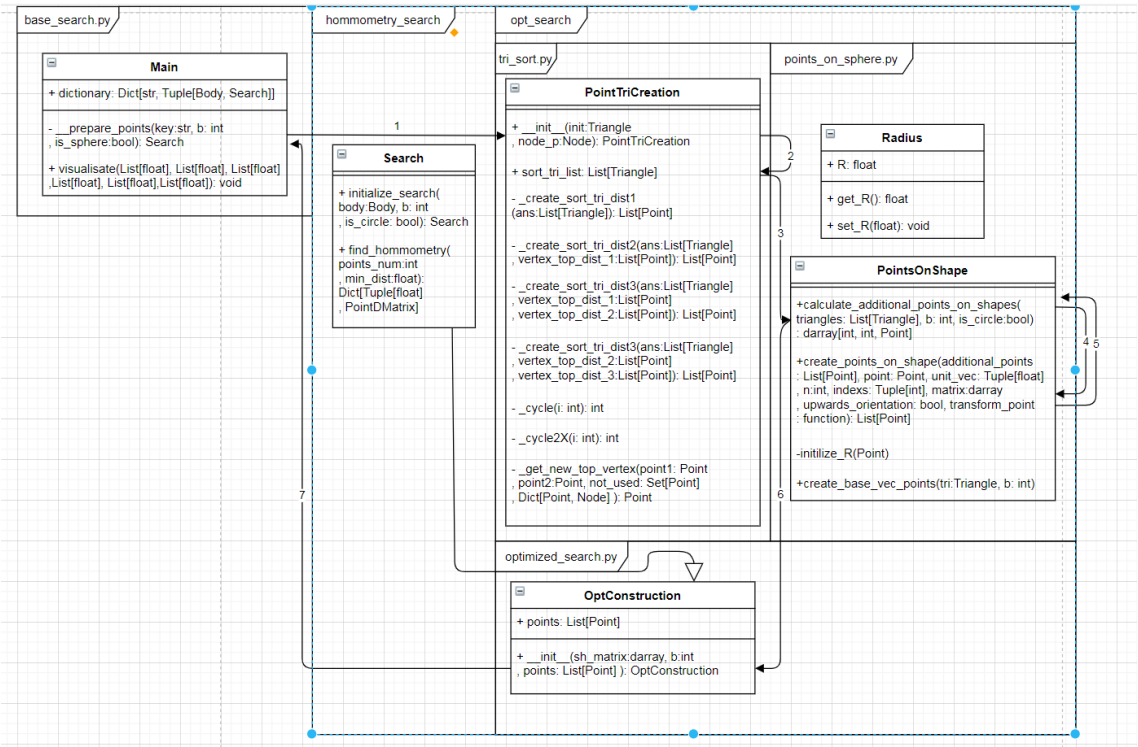


Joonis 16 Platooniline keha konstrueerimise kontseptuaalmudel

4.3.3 *X*_SEARCH moodulid

Kõik keerulised arvutused milleks kulutatakse palju aega on koondatud sellistes moodulites eesmärgiga kõrvaldada arvutuste dubleerimist ja selle kaudu suurendada programmi töökiirust. Arvutuste jaoks kasutatakse ka *hash-set* struktuuri, mis võimaldab mugavalt töötada suurte andmemahtudega ja kiiresti leida dublikaate andmekogumites [7].

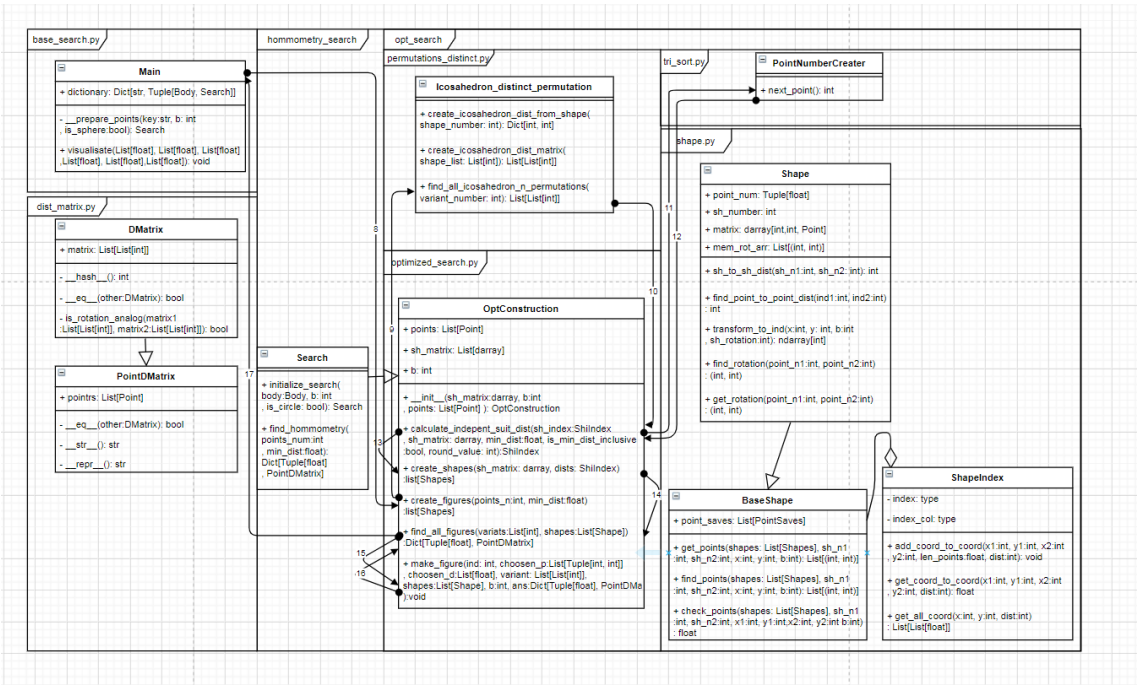
Joonisel 17 on esitatud kontseptuaalne mudel tegevusest, mis on seotud kolmnurkse mosaiigi ehitamisega. Enne kolmnurkse mosaiigi koostamist sorteeritakse tahud objekti PointTriCreation vastavalt sümmeetriarühmadele, mis on kirjeldatud lõigus 3.4. Seejärel moodul Points_on_sphere.py ehitab mosaiigi objektis PointsOnShape, vastavalt meetodile, mis on kirjeldatud lõigus 3.5. Meetod calculate_additional_points_on_shape läbib järjest tahkusid ja käivitab meetodi create_points_on_shape, mis rekursiivselt genereerib mosaiigi punktid.



Joonis 17 Punkti genereerimise protsessi kontseptuaalmudel

Mosaiigi punktidest maatriks edastatakse parameetrina objektile OptConstruktion moodulis Optimized_search.py, mis käivitab meetodi find_hommometry. Joonisel 18 on toodud kontseptuaalne mudel tegevusest, mis on seotud homomeetria otsinguga.

Objektiga icosahedron_distinct_permutation moodulis Permutation_distinst.py otsitakse unikaalsete tahkude komplektid, otsing on kirjeldatud lõigus 3.6. Iga komplekti kuuluva tahu jaoks meetodi create_icosahedron_dist_from_shape otsib kaugused ühest tahust kuni teiseni ja salvestab neid *hash-map*. Meetod create_icosahedron_dist_matrix loob kauguste maatriks, mis on eraldi objekt DMatrix, mille omaduseks on DMatrix objektide võrdlus omavahel. Meetod find_all_icosahedron_n_permutations otsib kõik võimalikud tahkude kombinatsioonid ja meetodiga create_icosahedron_dist_matrix loob iga kombinatsiooni jaoks oma kaugusmaatriksi, mis on ka objekt DMatrix ja oma omaduse tõttu automaatselt võrreldakse eelmise DMatrix-iga. Tulemuseks on unikaalsete tahkude komplekt. Seejärel meetod calculate_indepent_suit_dist genereerib alus- ja N-tahkude vahekauguste massiivi. Tulemuseks on objekt ShapeIndex. Järgmisena luuakse objekt Shape, mis oskab orienteeruda enda teise Shape suhtes vastavalt lõigus 3.7 kirjeldatud meetodile. Üks Shape-dest on objekt BaseShape, mis pärib Shape omadused ja lisaks hoiab endas objekt ShapeIndex. Kasutades enne loodud objekte meetoditega find_all_figures ja make_figures abil luuakse kõik võimalikud kujundid, mis koondatakse homomeetrilisteks rühmadeks, kui rühmas on olemas mitu kujundit, siis dublikaadid ei salvestu. Homomeetria on leitud juhul kui homomeetria rühmas osalevad vähemalt kaks kujundit, objektid PointDMatrix. Homomeetriliste kujundite andmed edastatakse mooduli Base_search.py.



Joonis 18 Homomeetria otsingu kontseptuaalmudel

4.4 Kasutamise juhend

Tarkvara töötab PYTON keskkonnas, see tähendab, et kasutaja arvutis peab olema installeeritud PYTON 3.8, mida saab installeerida PYTON ametlikust saidist https://www.python.org/

Nõutud funktsioonide tagamiseks on vaja lisada numpy (töö massiividega) ja plotly (visualiseerimine) raamatukogud.

Programm hakkab töötama peamooduli base_search.py, meetod main() käivitamisest. Selleks käsureast (aken command prompt) tuleb sisestada käsk *koht_kus_installeritud_python/* python.exe python_modul/base_search.py. Programm kasutab parameetreid, mis on toodud tabelis 1.

Tabel 1 Programmi parameetrid

Parameetri nimetus	**Võimalikud väärtused**	**Väärtused vaikimisi**	**Kirjeldus**
b	Natural arv 2-8	4	mitmeks punktiks jaguneb tahu serv
point	Natural arv4-6	4	tippude arv kujundis, mis kasutatakse homomeetria leidmiseks
dist	Reaalne arv 0-1	0.6	minimaalne kaugus kujundi tippude vahel. Kujundid -dist väiksema tippude vahekaugusega ei võeta arvesse
sphere	True/False	True	määrab kas luuakse sfäär (vaikimisi) või platooniline keha
method_key	Vaata moodulis base_sertch	ico_opt	platoonilise keha ja homomeetria leidmise meetodi lühendatud nimetus

Programmi töökäiku saab jälgida käsurea aknas. Visualiseeritud mudel avatakse brauseri aknas. Väljundandmed leitud homomeetriast kantakse faili JSON formaadis, mis automaatselt salvestatakse output.json.

4.5 Programmi arendamise võimalused

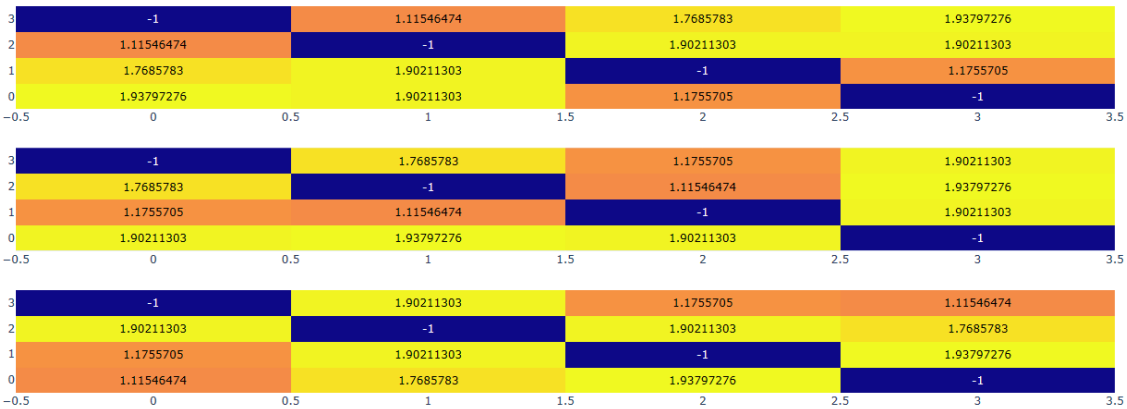
Üks mittefunktsionaalsest nõuetest on süsteemi laiendamise võimalus. Tarkvara on ehitatud niiviisi, et võimaldab juurde lisada moodulid, mis realiseerivad teised arvutusalgoritmid. Praegu realiseeritud aproksimatsioon ainult ikosaeedri baasil, kuid

olemas võimalus lisada moodulid teiste platooniliste kehade alusel. Selleks uus klass päritakse klassist Body, mis asub moodulis platon_body.body.py. Järgmiseks on vaja ka valida homomeetria leidmise meetod. Homomeetria leidmise klass peab pärinema klassist Search moodulis hommometry_search.search.py. Kõik uued moodulid on vaja lisada sõnastikku dict base_search.

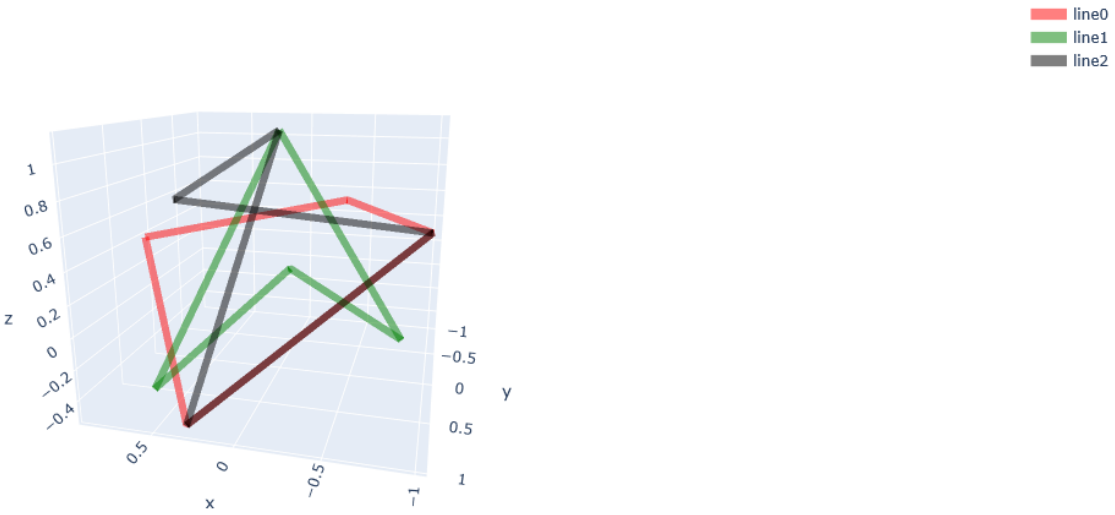
5 Tulemuste analüüs

Programmi visualiseeritud väljund koostatakse eraldi iga homomeetrilise paari jaoks paari järjekorra numbri järgi. Koosneb kahest osast. Üks nendest on 3D mudel, mis kuvab omavahel homomeetrilisi kujundeid. Mugavaks kasutamiseks on need kujundid asetatud koordinaatsüsteemi, mille alguspunkt on kerapinna keskpunkt. Vajadusel võib kuvada ka kerapinnal paigutatud punktid, mis osalevad homomeetria otsingus. 3D mudelit võib pöörata ümber keskpunkti ja vaadelda mudelit erinevast vaadepunktist. Kujundi tipule vajutades võib näha selle koordinaate.

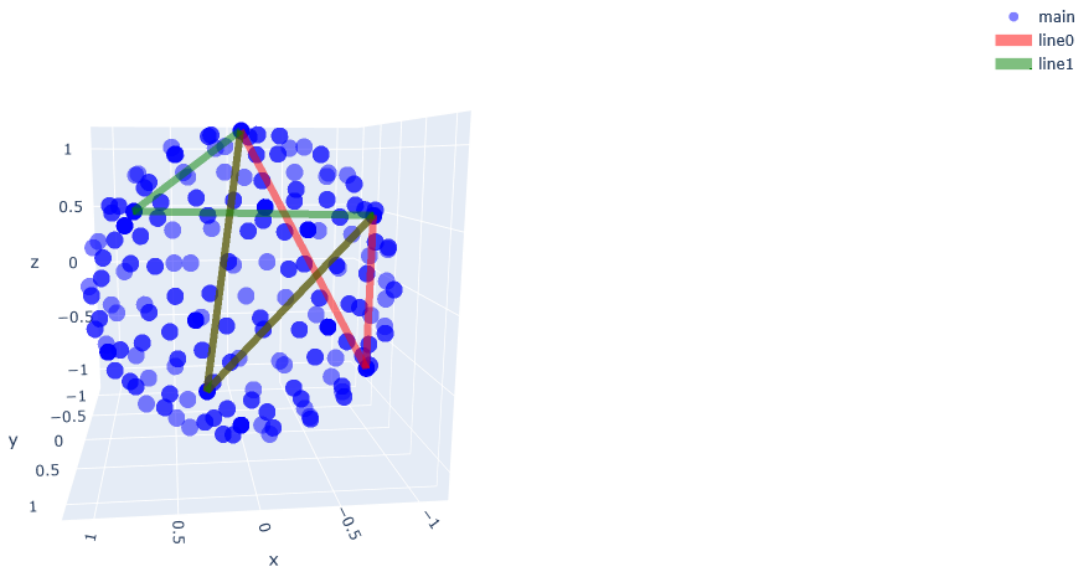
Teine väljund näitab tippudevaheliste kaugusmaatriksite väärtusi, mis on realiseeritud *heat matrix*-ina . Iga kujundi jaoks on tekitatud oma maatriks, veerud ja read näitavad kujundi tippude järjekorranumbrit. Mugavaks kasutamiseks on lahtrite väärtused kuvatud nii arvuna kui ka värvina.



Joonis 19 Homomeetria heat maatriks



Joonis 20 Homomeetria paarid ilma sfääri punktideta



Joonis 21 Homomeetria paarid koos sfääri punktidega

Väljund mis on salvestatud JSON failina näitab järjest homomeerililste kujundite paarid ja nendevaheliste kaugusmaatriksid järgmise struktuurina:

{[{matriks:[[d11, d12, d13],[d21, d22, d23], [d31, d32, d33]], punktid:[{x: p1x, y: p1y ,z: p1z}, {x: p2x, y: p2y ,z: p2z}, {x: p3x, y: p3y ,z: p3z}]}]}


```
{"hommometry": [{"pair-hommometry": [{"dist-matrix": [[-1,
1.90211303, 1.1755705, 1.90211303], [1.90211303, -1, 1.90211303,
1.1755705], [1.1755705, 1.90211303, -1, 1.90211303],
[1.90211303, 1.1755705, 1.90211303, -1]], "points": [[0.0, 0.0,
1.118033988749895], [0.30901699437494745, 0.9510565162951535, -
0.5], [-1.0, 1.2246467991473532e-16, 0.5], [-0.8090169943749473,
0.5877852522924732, -0.5]]}, {"dist-matrix": [[-1, 1.90211303,
1.1755705, 1.1755705], [1.90211303, -1, 1.90211303, 1.90211303],
[1.1755705, 1.90211303, -1, 1.90211303], [1.1755705, 1.90211303,
```

```
1.90211303, -1]], "points": [[0.0, 0.0, 1.118033988749895],
[0.30901699437494745, 0.9510565162951535, -0.5], [-1.0,
1.2246467991473532e-16, 0.5], [0.8090169943749473, -
0.5877852522924734, 0.5]]}]}, …]}
```


Käsureas väljastatakse JSON faili loomise informatsioon. Esimeses reas kuvatud kõik võimalikke mitte unikaalsete kombinatsioonide arv. Järgmisel real on aeg, mis läks unikaalsete kombinatsioonide leidmisele, aeg on sekundites. Seejärel on info, mis iseloomustab unikaalsete tahkude kombinatsiooni leidmise protsessi – viimase komplekti töötlemise aeg, töödeldava komplekti järjekorranumber ja komplektide kogumaht, alustahu punkti number ning aeg, mis on läinud eelmise punktiga seotud kujundite töötlemiseks. Järgmisel real on aeg, mis on läinud kõigi kujundite moodustamiseks. Viimasel real on aeg, mis kulutatud kogu protsessile.

```
969 permutation number
0.27132153511047363 permutation minimaize time
last segment time: 10.771827459335327 fig 59/60 (point [(4, 0)] 0.9932861328125
478.942910194397
529.4006519317627 time_needed
```

Joonis 22 Käsurea väljund

6 Kokkuvõte

Käesoleva töö eesmärgiks oli teha tarkvara molekuligeomeetriate mudeli genereerimiseks, mis vastab loodusteaduskonna arvutuskeemia uurimisgrupi nõuetele selleks, et tuvastada, kas eksisteerivaid seal homomeetrilised konfiguratsioonid või mitte. Eesmärgi täitmiseks olid uuritud olemasolevad punktikombinatsioonide algoritmid kera pinnal genereerimiseks ja matemaatilised meetmed invariantsete kombinatsioonide eemaldamiseks ja homomeetria leidmiseks. Tulemuseks oli loodud mitu matemaatilist mudelit, mis on realiseeritud PYTHON programmeerimiskeeles.

Töös on püstitatud eesmärk täidetud. Tarkvara töö tulemusest selgub, et molekulis, milles on üks keskse aatom ja sellest võrdsetel kaugustel asuvad ülejäänud kahest kuni kuueni aatomid, eksisteerivad homomeetrilised kombinatsioonid. Toodud tarkvara võib tulevikus laiendada ja täiendada vastavalt TalTech loodusteaduskonna arvutuskeemia uurimisgrupi vajadusele.

Kasutatud kirjandus

1. Marjorie Senechal, 2008, “A point set puzzle revisited” [Võrgumaterjal]. Avalailable: https://www.sciencedirect.com/science/article/pii/S0195669808000322 [Kasutad 17 05 2020]
2. Conformational isomers [Võrgumaterjal]. Available: https://www.brainkart.com/article/Conformational-isomers_29832/ [Kasutatud 17 05 2020]
3. Point Picking and Distributing on the Disc and Sphere, 2015 [Võrgumaterjal]. Avalailable: https://www.semanticscholar.org/paper/Point-Picking-and-Distributing-on-the-Disc-and-Arthur/10a2b95eaebf87f458db31f00aaf8c699c99caff [Kasutatud 17 05 2020]
4. H.S.M.Coxeter, 1973, “Regular polytopes” [Võrgumaterjal]. Avalailable: https://books.google.ee/books/about/Regular_Polytopes.html?id=iWvXsVInpgMC&redir_esc=y [Kasutad 17 05 2020]
5. F. Albert Cotton, 1990, „Chemical Applications of group theory” [Võrgumaterjal]. Avalailable: https://www.worldcat.org/title/chemical-applications-of-group-theory/oclc/19975337 [Kasutad 17 05 2020]
6. „Построение графических примитивов Математические модели поверхностей и объектов“ [Võrgumaterjal]. Available: https://works.doklad.ru/view/1SxrxyeGZoA.html [Kasutatud 17 05 2020]
7. Python hashing tutorial [Võrgumaterjal]. Avalailable: http://zetcode.com/python/hashing/ [Kasutad 17 05 2020]
8. The Python Tutorial https://docs.python.org/3/tutorial/ [Kasutad 17 05 2020]
9. Plotly Python Open Source Graphing Library Fundamentals [Võrgumaterjal]. Avalailable: https://plotly.com/python/plotly-fundamentals/ [Kasutad 17 05 2020]
10. NumPy Documentation¶ [Võrgumaterjal]. Avalailable https://numpy.org/devdocs/user/quickstart.html [Kasutad 17 05 2020]