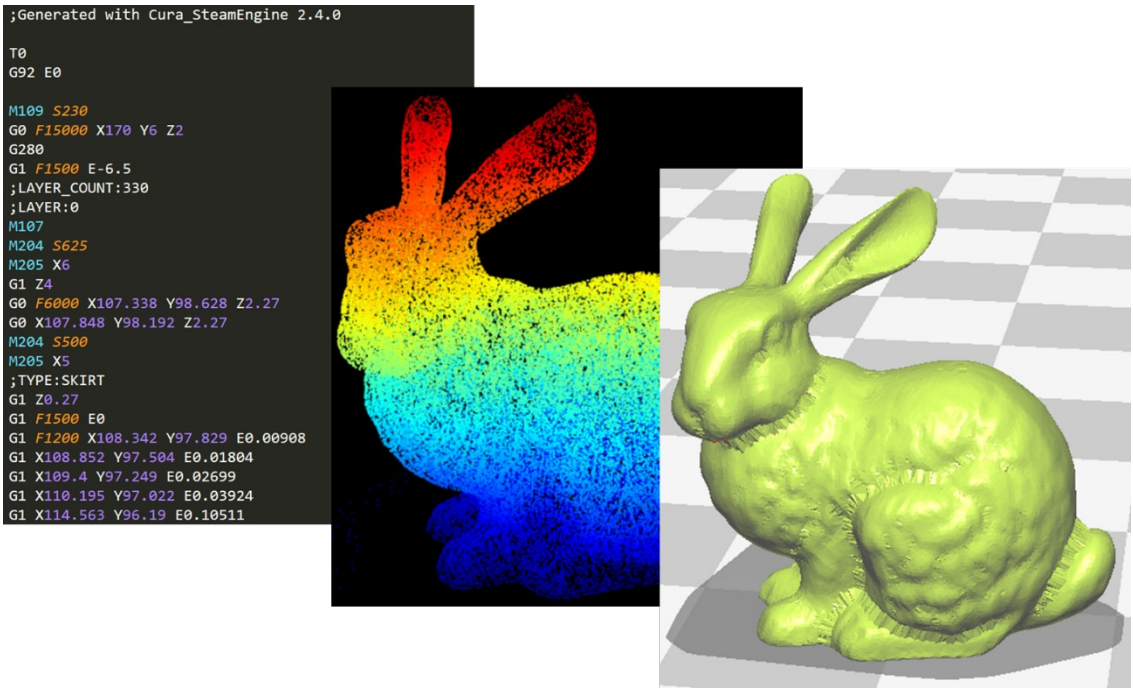
אוניברסיטת תל אביב, בית הספר למדעי המחשב
קורס הדפסה תלת ממדית, פרויקט סיום

GCode2STL

יאיר קרין



אבסטרקט

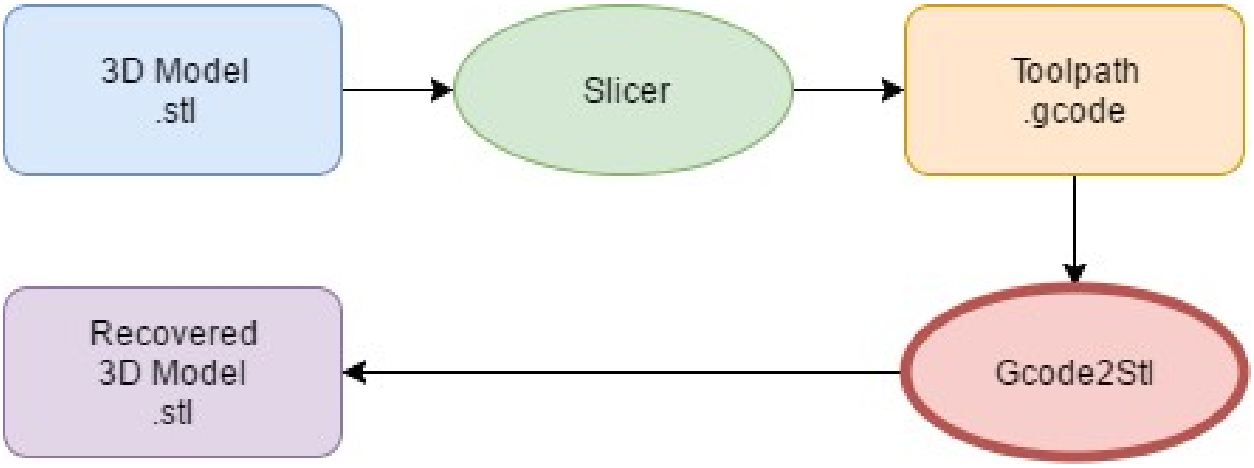
הדפסה תלת-ממדית תוארה כ"מהפכה התעשייתית הבאה". עם הצמיחה בתחום בעשורים האחרונים מתחילים לעלות שאלות ומחקרים בנושא אבטחת תהליך ההדפסה התלת-ממדית. באופן ספציפי, רבים מודאגים מבעיית "גניבת הקניין הרוחני של מודלים תלת-ממדיים". בתהליך ההדפסה התלת-ממדית עובר המודל התלת-ממדי המרה לפורמט פקודות למדפסת התלת-ממדית, דהינו פורמט GCode. כחלק מהתמודדות בבעיית זכויות היוצר מועברים מודלים בפורמט GCode מלכתחילה כצעד מונע להחזקת המודל המקורי. בפרויקט זה מוצגת הוכחה לכך שפתרון זה אינו מספיק, וניתן לשחזר את המודל המקורי מתוך ה GCode במידה מספקת. השחזור נעשה באמצעות דגימה של נקודות מתוך פקודות ה GCode ובנייה של mesh תלת-ממדי מתוך דגימת הנקודות באמצעות אלגוריתם α-shapes. בנוסף מוצגת התאוריה מאחורי α-shapes, ומוצגים ניסויים שנעשו ותוצאותיהן להערכת איכות השחזור.

מבוא

מטרת הפרויקט:

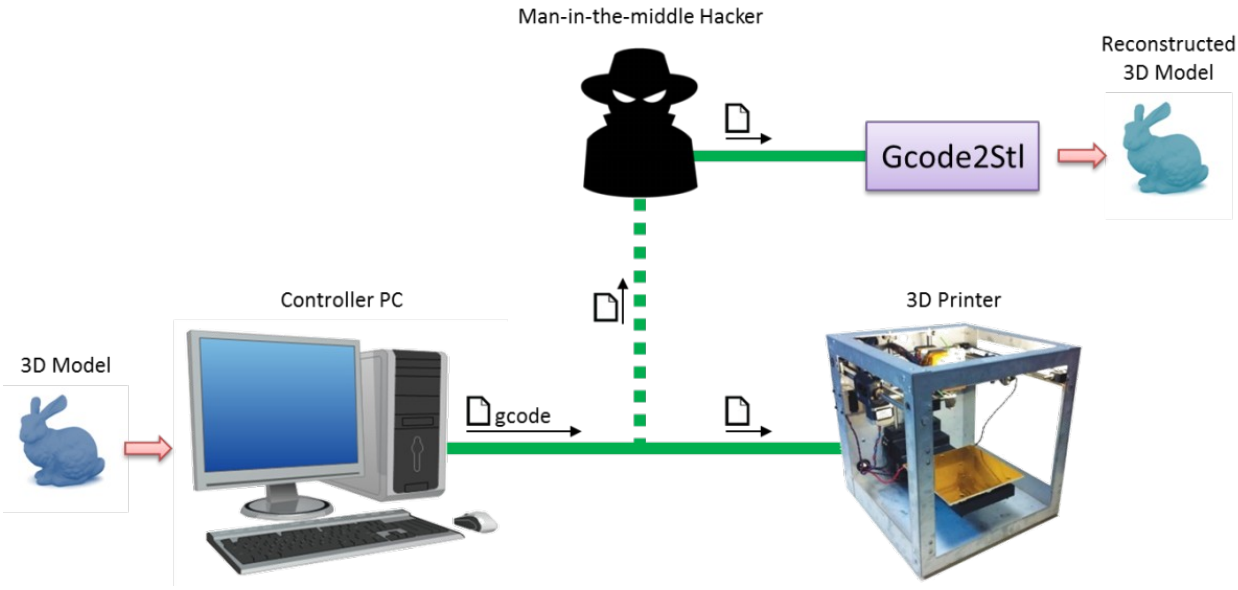
כאשר ברצוננו להדפיס מודל תלת-ממדי (stl), יש לתרגמו לפקודות של המדפסת התלת-ממדית. פקודות אלו נקראות gcode, ותהליך ההמרה נקרא slicing.

בהינתן קובץ המכיל פקודות gcode להדפסת מודל, נרצה לשחזר ממנו את המודל בפורמט stl.



תרחיש:

נתאר תרחיש אפשרי: מדפסת תלת-ממדית בחברה גדולה מחוברת בכבל רשת למחשב הדפסה. על מנת להגן על הפטנטים (מודלים תלת-ממדיים) של החברה, המודלים עוברים slicing וה gcode נשלח למדפסת מעל כבל הרשת. האקר (פצחן) זדוני השיג גישה לרשת ורוצה לגנוב את הפטנטים של החברה.



איבוד מידע בתהליך ה slicing:

מדפסות FDM עובדות בשכבות. בתהליך ה slicing עובר אובייקט תלת-ממדי **רציף** תהליך של **דגימה דיסקרטית** של שכבות. כתוצאה מכך ישנו איבוד מידע ב slicing, והפעולה ההפיכה (אותה ברצוננו להשיג) אינה אפשרית ב 100% הצלחה.

שלבי הפרויקט:

1. כתיבת פרסר ל GCode:
   - למידת הפורמט.
   - בניית ה toolpath, המסלול של ראש המדפסת במרחב.
2. יצירת point cloud, דגימה של נקודות על ה toolpath צפופה כרצוננו:
   - בחירת צפיפות.
   - מעבר על ה toolpath ודגימת נקודות בצפיפות שנבחרה.
3. בניית מודל תלת-ממדי מתוך ה point cloud:
   - יעשה באמצעות רכיב α-shapes של CGAL.
   - תאוריה, α-shapes.
4. הערכה והסקת מסקנות:
   - עריכת ניסויים למדידת איכות השחזור של מודלים שונים עם פרמטרים שונים.
   - הסקת מסקנות.
5. הדגמה מוחשית:
   - הדפסת מודל במדפסת תלת-ממדית.
   - הדפסת המודל ששוחזר מה GCode של המודל הנ"ל במדפסת תלת-ממדית.
6. הצעות לשיפור:
   - Generalized α-shapes.
   - הערות ב GCode.

פתרון קיים לבעיה:

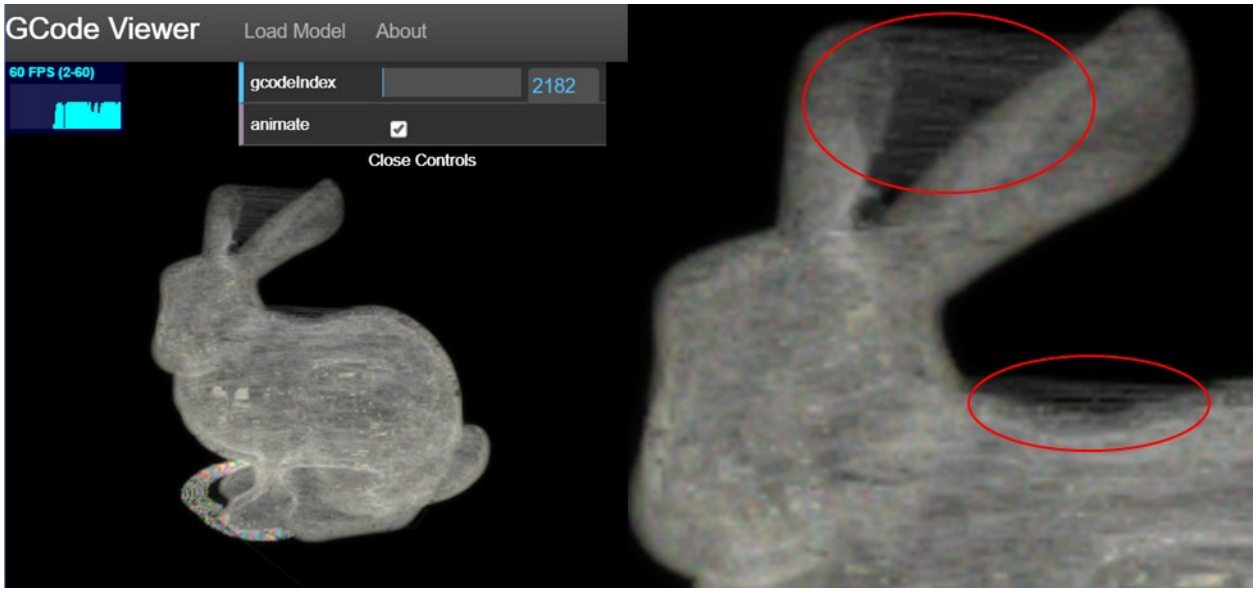
לאחר חיפוש ממצה בגוגל, לא הצלחתי למצוא תוכנה המספקת פתרון לבעיה שהצגתי מקצה לקצה. קיימים פתרונות טובים מאד לרכיבים חלקיים בתהליך הפתרון, אך לא פתרון שלם שמשלב ביניהם.

חלק 1+2 פרסר ודוגם

עבודה קודמת:

קיימות תוכנות online המסוגלות להציג את ה toolpath בהינתן ה GCode. לדוגמה,

GCode Viewer. http://jherrm.com/gcode-viewer/



הסתכלתי על קוד המקור של תוכנה זו (ועוד כמה תוכנות), ועלו מספר בעיות בשימוש בהן:

1. התוכנות מבוססות תצוגת browser, ולכן כתובות ב Javascript. JavaScript הינה שפת תכנות לא טובה לפרויקטים גדולים הדורשים דיוק גבוה ו robustness.
2. חלק מהתוכנות לא מתייחסות לכל הפרמטרים בפקודות ה GCode. למשל הן מתעלמות מהפרמטר E שמסמן כמה חומר להוציא בתנועה. כתוצאה מכך התוכנות אינן מבדילות בין תנועה של הראש לטובת הדפסת חומר ותנועה של הראש לטובת מעבר בין חלקים בלתי חופפים. ניתן לראות זאת בתנועת הראש בין אוזני הארנב בתמונה הנ"ל.

לכן החלטתי שאממש את הפרסר והדוגם בעצמי.

קצת על פורמט GCode:

1. תוכנה בנויה מבלוקים הממוקמים בשורות שונות.
2. בלוק בנוי מפקודה ומ 0 או יותר פרמטרים לפקודה.
3. פקודה הינה אות באנגלית ולאחריה מספר שלם.
4. פרמטר הינו אות באנגלית ולאחריה מספר floating point.
5. הערה מתחילה ב ";" ונגמרת בסוף השורה.
6. ישנן פקודות מיוחדות לבקרת איכות:

- * ולאחריו מספר שלם מסמל checksum של השורה, ומופיע לאחר הפרמטר אחרון.
- N ולאחריה מספר שלם מסמל את מספר השורה ומופיע לפני הפקודה.

דוגמה:

N41 G1 F3300 X100.934 Y94.082 Z20.022 E2269.34037 *37 ; Comment

G1	Rapid Linear Move, תנועה ליניארית של הראש.
F3300	האץ את תנועת הראש למהירות 3300mm/m (מ"מ/דקה).
X100.934	מיקומו הסופי של הראש בציר 100.934 x מ"מ.
Y94.082	מיקומו הסופי של הראש בציר 100.934 x מ"מ.
Z20.022	מיקומו הסופי של הראש בציר 100.934 x מ"מ.
E2269.340	הוצא 2269.340 מ"מ של חומר.

מימוש הפרסר והדוגם:

המימוש נעשה ב cpp באמצעות CGAL.

ניסיתי ליצור מודל פשוט המייצג את הקשר בין רצף פקודות ה GCode לרצף המצבים בהן נמצאת המדפסת, על מנת שיתאפשר לדגום נקודות מה toolpath בקלות.

בנוסף, שמתי דגש על קלות הוספת פקודות GCode. אני תמכתי ב2 פקודות G0, G1 אך ניתן בפשטות רבה להוסיף תמיכה בעוד.

תקציר העיצוב:

PrinterState – המצב בו נמצאת המדפסת ברגע מסוים. מכיל את מיקום הראש, מהירות המאוורר, חום וכו'.

GCodeCommand – מחלקת בסיס לכל פקודות ה GCode המייצאת את הפונקציונאליות הבאה:

- constructor: אתחול הפקודה, מקבל את הבלוק (טקסט) של הפקודה. תפקידו לפרסר את הפרמטרים של הפקודה.
- getNextPrinterState: בהינתן מצב נוכחי ופקודה, החזר את המצב של המדפסת לאחר שהפקודה התבצעה.
- samplePoints: בהינתן מצב נוכחי, פקודה וצפיפות, דגום נקודות בתנועה בצפיפות הנתונה.

פקודה לדוגמה – G0, מחלקה שיורשת מ-GCodeCommand, דורסת את 3 הפונקציות באופן הבא:

- constructor: חילוץ הפרמטרים האפשריים X, Y, Z, E, F והמרתם לטיפוס נתונים רציונלי מדויק.
- getNextPrinterState: מיקום הראש יתקדם ל X, Y, Z הנתונים.
- samplePoints: דגימה ליניארית של נקודות על הישר המוגדר ע"י מיקום הראש הקודם ומיקום הראש הבא.

GCodeToolpath – רשימה של אובייקטי PrinterState ואובייקטי GCodeCommand המקשרים ביניהם.

PrinterSettings – הגדרות גלואבליות עבור ההדפסה, לדוגמה עובי שכבה.

GCodeParser – מקבל מחרוזת GCode ומחזיר ParserProduct.

ParserProduct – מכיל GCodeToolpath ו PrinterSettings.

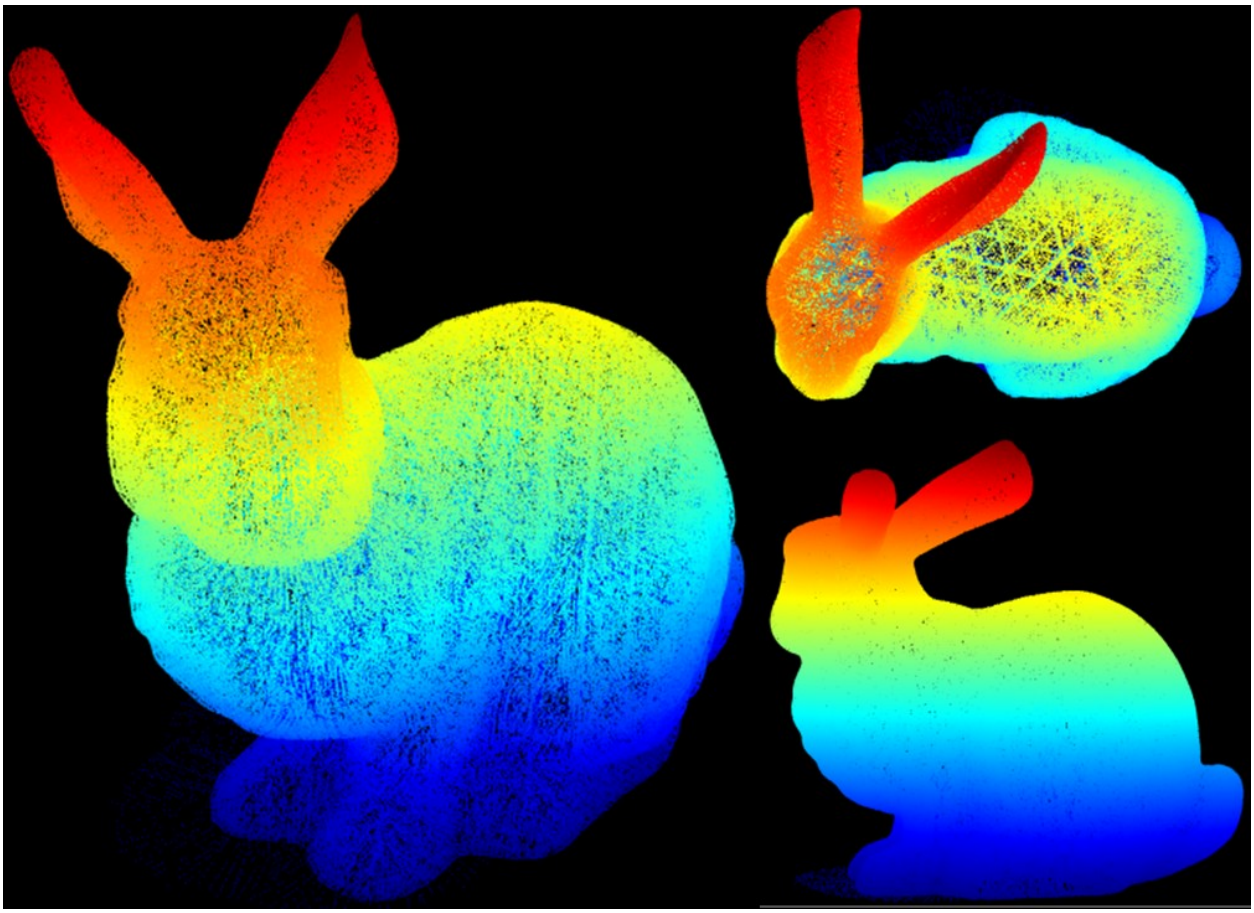
GCodeSampler – מקבל ParserProduct וצפיפות ומחזיר דגימה של נקודות מה GCodeToolpath בצפיפות הנתונה בהתחשב ב PrinterSettings.

בדיקת הפרסר והדוגם:

על מנת לבדוק ויזואלית את התוצאות השתמשתי בכלי

Online LIDAR point cloud viewer. http://lidarview.com/

כלי זה מקבל כקלט קבצי xyz, ומציג את הנקודות בתלת-ממד.
להלן הוויזואליזציה של הדגימה שנעשתה באמצעות הסקריפט שכתבתי עבור ה Stanford Bunny:



הדגימה נעשתה בצפיפות של 1mm. מידות המודל 40mm x 31mm x 40mm.

חלק 3 α-shapes

α-shapes, תאוריה:

מבוסס על המאמר

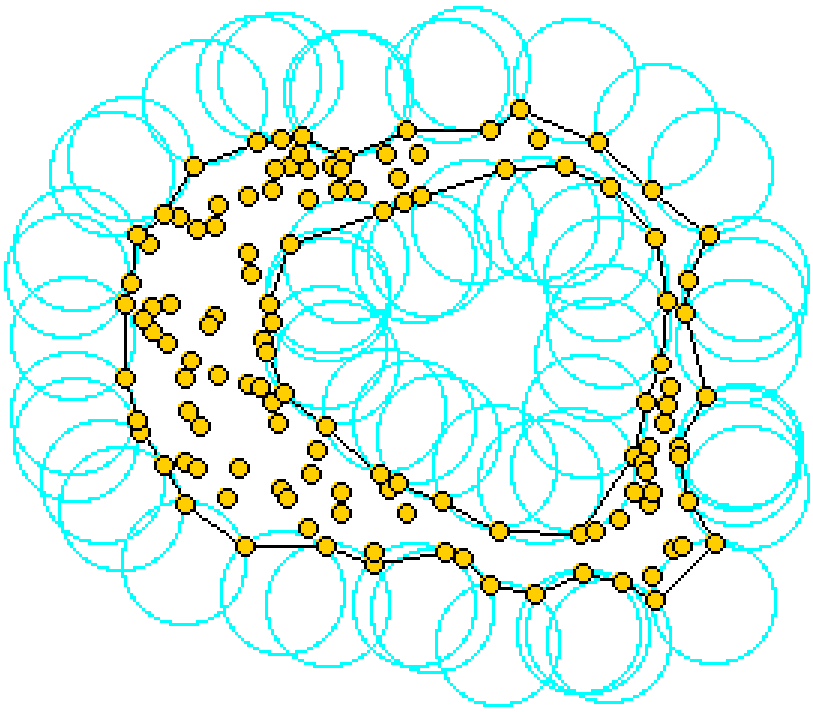
Introduction to Alpha Shapes. Kaspar Fischer.
https://graphics.stanford.edu/courses/cs268-11-spring/handouts/AlphaShapes/as_fisher.pdf

אינטואיציה ל α-shapes:

תהי $S \subset R^d$ קבוצת נקודות ב 2D או 3D ונרצה למצוא את "הצורה הנוצרת מהנקודות האלה".
זוהי הגדרה מעורפלת וניתן לפרש אותה בהרבה דרכים, כגון קמור ומעגל חוסם מינימלי.
דרך נוספת הינה α-shape.

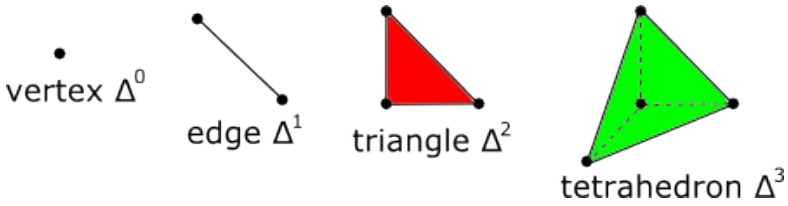
תיאור אינטואיטיבי מהמאמר המקורי על α-shapes (H. Edelsbrunner and E. P. Muc¨ ke. Three-dimensional alpha shapes):

נדמיין גוש ענקי של גלידה שמרכיבה את כל R^d המכילה בתוכה את נקודות S בתור צ'יפים קשיחים של שוקולד. באמצעות כף גלידה בצורת כיפה נחפור החוצה את כל הגלידה אליה ניתן להגיע עם הכף מבלי להתקל בצ'יפים מהשוקולד. נסיים עם אובייקט (לא בהכרח קמור) חסום ע"י כיפות, קשתות ונקודות. אם ניישר את כל השפות העגולות למשולשים וסגמנטים, יש לנו תיאור אינטואיטיבי למושג α-shape.
מהו α בתיאור זה? α הינו הרדיוס של הכף הגלידה. כאשר $\alpha \to 0$ ה α-shape מתנוון לקבוצת הנקודות S, ומצד שני $\alpha \to \infty$ ימנע מהכף לעבור דרך הצ'יפים ולאסוף גלידה מביניהם ולכן נקבל כי ה-α-shape הינו הקמור של S.



הגדרת α-shape:

הגדרה: simplex-k הינו קמור k ממדי של תת קבוצה $T \subseteq S$ בגודל $|T|=k+1$ כך ש $k \le d$.



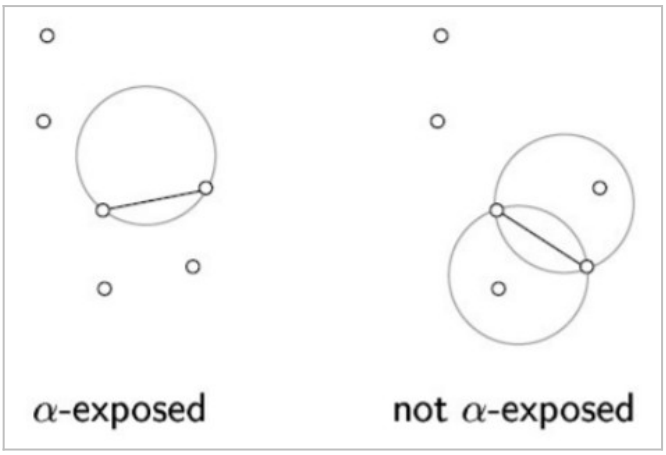
נניח כי הנקודות ב S במצב כללי:

- אין 4 נקודות על אותו המישור
- אין 5 נקודות על אותו הכדור (sphere)
- עבור α קבוע לכדור הקטן ביותר דרך על 2, 3 ו 4 נקודות יש רדיוס השונה מ α

הנחות אלה מבטיחות כי מתקיים שלכל קבוצה $T \subset S$ כך ש $|T|=k+1 \le d+1$, הפוליטופ $\Delta_T = CH(T)$ בממד k ולכן הינו simplex-k. כלומר 2 נקודות מגדירות קו, 3 נקודות מגדירות משולש, 4 נקודות מגדירות טטרהדרון.

מספר הגדרות:

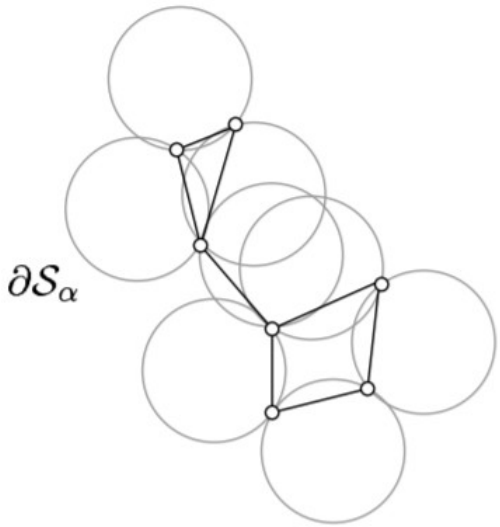
- עבור $0<\lambda<\infty$, λ-**ball** מוגדר בתור הכדור (ספרה) הפתוח עם בעל רדיוס λ.
- נסמן ∂b המעטפת של b.
- simplex-k Δ_T הינו α-**exposed** אם קיים α-ball b **ריק** כך ש $T = \partial b \cap S$.



נסמן S_α בתור ה-shape α של S. אז ∂S_α המעטפת של S_α מכיל את כל ה-simplices-k של S שהם α-exposed:

$$\partial S_a = \{\Delta_T | T \subset S, |T| \le d, \Delta_T \, is \, \alpha - exposed\}$$

נשים לב כי shape-α תלת-ממדי מורכב ממשולשים, סגמנטים ונקודות.



שילוש דלוני ו α-shapes:

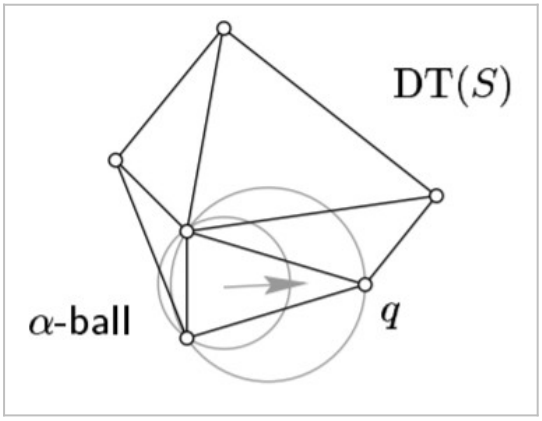
שילוש Delaunay של $S \subset R^d$ הוא שילוש של S (כלומר, חלוקה של הקמור של קבוצת הנקודות ל d-simplices זרים כך שכל הנקודות בקבוצה הן קודקודים שלהם) $DT(S)$ המורכב מ:

- כל ה-simplices-d שהמעגל החוסם שלהם לא מכיל אף נקודה אחרת.
- כל ה-simplices-k שהם פאות של simplices אחרים ב $DT(S)$.

טענה: אם Δ_T הוא α-*exposed* אז $\Delta_T \in DT(S)$.

הוכחה:

אם Δ_T הוא *simplex*-d אז ה *ball*-α מתאחד עם המעגל החוסם ולכן $\Delta_T \in DT(S)$.
נניח כי Δ_T הוא *simplex*-k ($k<d$) ונניח בשלילה כי $\Delta_T \notin DT(S)$.
נזיז את המרכז של ה *ball*-α b הריק ברציפות תוך כדי התאמת הרדיוס כך שנקודות T ישארו על המעטפת ∂b, עד שבסופו של דבר יפגע בנקודה $q \in S \setminus T$.



בכך הגדלנו את k ל $k+1$. אם $k+1=d+1$ מצאנו $\Delta_{T \cup \{q\}} \in DT(S)$ ולכן גם $\Delta_T \in DT(S)$ (בתור שפה שלו). אחרת, נמשיך בתהליך אינדוקטיבית עד ש b יגע ב $d+1$ נקודות. ∎

מטענה זו נובע כי $\partial S_\alpha \subset DT(S)$ לכל $\alpha \geq 0$, כלומר ה shape-α הינו תת גרף של שילוש דלוני.

אלגוריתם:

- לכל **simplex**-$d-1$ ב $DT(S)$ Δ_T
- אם אחד מהכדורים ברדיוס α החוסמים שלו ריק
- אז Δ_T הוא exposed-α

מה אם simplices-k עם $k<d-1$? ישנם אינסוף balls-α שנוגעים בהם. ישנו חלק נוסף לאלגוריתם של Edelsbrunner שמטפל בהם, אך מאחר ואנו מעוניינים במשולשים בלבד נסתפק בגרסה זו של האלגוריתם.

סיבוכיות: שילוש דלוני בתלת-ממד: $O(n^2)$. עבור כל simplex-$d-1$ בודקים אם יש נקודה בכדורים החוסמים ברדיוס α, לכן סה"כ $O(n^3)$.

ניתן ליעל את האלגוריתם הנ"ל.

אלגוריתם משופר:

- לכל **simplex**-d ב $DT(S)$ C_T
- אם רדיוס הכדור החוסם שלו קטן מ α
- אז פאות C_T הן exposed-α, החזר את פאות C_T החיצוניות

מתי פאה $f \in C_{T_1}$ חיצונית? אם f חלה גם ב $C_{T_2} \in DT(S)$ בעל רדיוס קטן שווה α, היא פנימית, אחרת היא חיצונית.

סיבוכיות: עבור כל simplex-d בודקים אם הרדיוס שלו קטן מ α ($O(1)$), ולכן סה"כ $O(n^2)$.

נותר להוכיח שהאלגוריתם המשופר שקול.

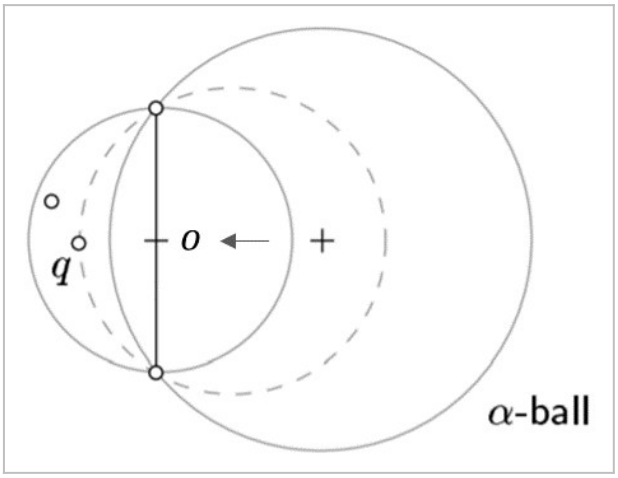
טענה: אם $\Delta_T \in \partial S_\alpha$ simplex-$(d-1)$ אז קיים $C_{T'} \in DT(S)$ simplex-d כך ש Δ_T פאה של $C_{T'}$ וגם הרדיוס של המעגל החוסם של $C_{T'}$, α', מקיים $\alpha' \leq \alpha$.

(ב 3D עבור משולש Δ_T ב ∂S_α קיים טטרהדרון C'_T ב $DT(S)$ כך ש Δ_T פאה של $C_{T'}$.

ב 2D עבור סגמנט Δ_T ב ∂S_α קיים משולש $C_{T'}$ ב $DT(S)$ כך ש Δ_T צלע של $C_{T'}$.)

הוכחה:

נסמן o בתור המרכז של המעגל החוסם של Δ_T. נזיז את המרכז של ה α-ball של Δ_T ברציפות לכיוון o תוך כדי התאמת הרדיוס שלו כך שנקודות T ישארו על המעטפת שלו, עד שבסופו של דבר יפגע בנקודה $q \in S \setminus T$.



ברגע זה מצאנו simplex-d $C_{T'}$ עבור $T' = T \cup \{q\}$ שהמעגל החוסם שלו ריק ולכן $C_{T'} \in DT(S)$. אבל מכיוון שבהכרח הקטנו את הרדיוס של הכדור אזי מתקיים כי $\alpha' \leq \alpha$. ∎

נותר להוכיח עוד שתי טענות (ברוח ההוכחה הנ"ל):

- אם $\Delta_T \in \partial S_\alpha$ simplex-$(d-1)$ אז קיים $C_{T'} \in DT(S)$ simplex-d כך ש Δ_T פאה של $\partial C_{T'}$ וגם הרדיוס של המעגל החוסם של $C_{T'}$, α', מקיים $\alpha' \leq \alpha$. כלומר נחזק את הטענה הנ"ל ש Δ_T יהיה על החלק **החיצוני** של $C_{T'}$.
- הטענה ההפוכה: אם $C_{T'} \in DT(S)$ simplex-d ו Δ_T simplex-$(d-1)$ פאה חיצונית שלו אז $\Delta_T \in \partial S_\alpha$.

חסכתי את ההוכחות הנ"ל כי הן מצריכות הצגת קונספט נוסף בשם complex-α שהוא פחות רלוונטי להבנה הכללית של הוריאנט של האלגוריתם שבחרתי להציג.

α-shapes ב CGAL:

מבוסס על התיעוד של CGAL

CGAL 4.10 - 3D Alpha Shapes. Tran Kai Frank Da, Sébastien Loriot, and Mariette Yvinec.
https://doc.cgal.org/latest/Alpha_shapes_3/index.html

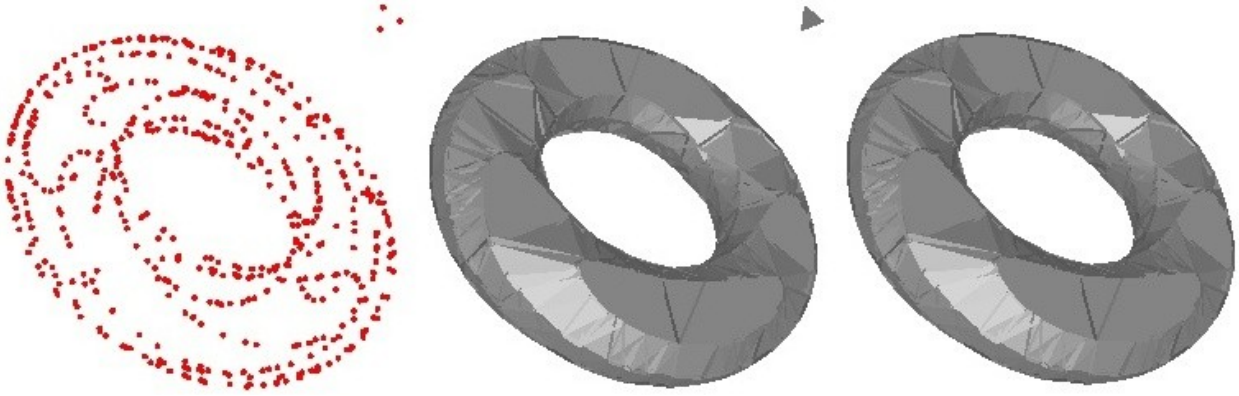
באלגוריתם שהוצג עוברים על כל הטטרהדרונים ב $DT(S)$ (שילוש הדלוני של קבוצת הנקודות S) ומחזירים את הפאות החיצוניות של הטטרהדרונים שנכנסים בכדור ברדיוס α.

באופן כללי α-shape יכול להכיל גם פאות שאינן פאות של טטרהדרון כנ"ל, פאות כאלו מכונות singular.

CGAL מציעה 2 שיטות להתמודד עם פאות אלה:

- General: פאות שהן singular נכללות ב α-shape.
- Regularized: פאות שהן singular אינו נכללת ב α-shape.

להלן השוואה ביניהן.
משמאל: נקודות שנדגמו מפני טורוס, ו 3 נקודות רחוקות באופן יחסי מהפנים שלו.
באמצע: α-shape של הנקודות בשיטת general, המכילה את 3 הנקודות.
מימין: α-shape של הנקודות בשיטת regularized, שלא מכילה את 3 הנקודות.



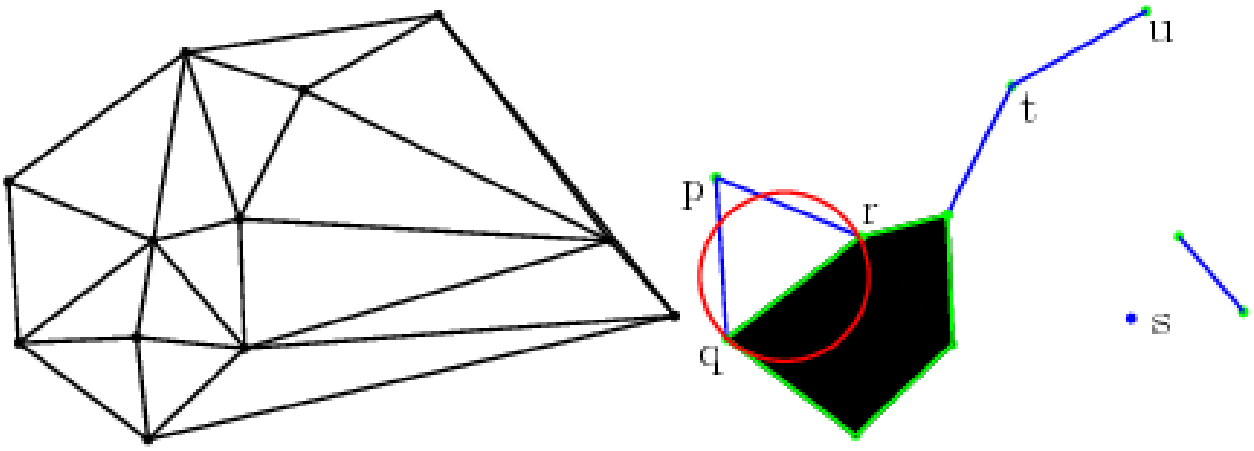
שיטות לחישוב ה α-shape ב CGAL:

- Fixed α: זהו האלגוריתם שהצגתי, המסוגל לחשב α-shape בהינתן ערך α. זהו אלגוריתם יעיל יותר ועל כן מהיר יותר מהשיטה השנייה שמציעה CGAL. בנוסף, הוא מצריך רק predicates (בניגוד לשיטה שנייה המצריכה גם constructions) ועל כן עם אותו ה kernel נצפה לביצועים משמעותית יותר טובים.
- Family of α-shapes: בהינתן קבוצת נקודות S, ניתן לחשב ערך a_{min} עבור כל $\Delta_T \in DT(S)$ שעבור Δ_T $\alpha > a_{min}$ יהיה חלק מהמעטפת של ה α-shape ו α_{max} שעבור Δ_T $\alpha > a_{max}$ יהיה בחלק הפנימי של ה α-shape. חישוב זה מאפשר אופרציות חזקות כגון מציאת ערך ה α האופטימלי כך שיהיו x רכיבי קשירות ב α-shape. מכיוון שמצאתי דרך לחשב ערך α טוב על סמך עובי שכבה וצפיפות הדגימה (יתואר עוד בהמשך) אין צורך באופרציה כזו. בנוסף, איננו יודעים בהכרח כמה רכיבי קשירות היו במודל המקורי.

סיווג רכיבים ב α-shape ב CGAL:

החישוב מסווג כל k-simplex ב Δ_T ב $DT(S)$ לאחד מהבאים:

- REGULAR — Δ_T שפה של d-simplex שרדיוסו קטן שווה α. (בירוק)
- SINGULAR — Δ_T אינו שפה של $k+1$-simplex אחר ב α-shape. (בכחול)
- INTERIOR — אם Δ_T אינו על המעטפת של ה α-shape. (בשחור)
- EXTERIOR — אם Δ_T על המעטפת של ה a-shape.



חישוב ערך α:

הדגימה מתקבלת מ GCode toolpath, הבנוי משכבות. בנוסף, הדגימה נעשית בצפיפות נתונה.

דרישות מערך α:

דרישה	ביטוי מתמטי
נרצה שכדור ברדיוס α לא יחדור בין שכבות המודל. נסמן עובי שכבה l.	$\alpha \geq 0.5 \cdot l 2 \cdot \alpha \geq l$
נרצה שכדור ברדיוס α לא יחדור בין שתי נקודות שנדגמו. נסמן צפיפות דגימה e.	$\alpha \geq 0.5 \cdot e 2 \cdot \alpha \geq e$
מסקנה	$\alpha \geq max(0.5 \cdot l, 0.5 \cdot e)$

סיבוכיות האלגוריתם:

מספר הנקודות תלוי בצפיפות הדגימה של ה GCode toolpath. ככל שהיא קטנה יותר ישנן יותר נקודות.

נסמן M בתור האורך של התנועה הארוכה ביותר ב toolpath שיוצא בה חומר מראש המדפסת. מתנועה זו ידגמו $\lceil M/e \rceil$ נקודות כאשר e הינה צפיפות הדגימה.

נסמן n בתור מספר התנועות ב toolpath. על כן, ישנן לכל היותר $O(\lceil M/e \rceil \cdot n)$ נקודות בדגימה.

נסמן בתור S את קבוצת הנקודות שנגדמו, $|S|=O(\lceil M/e \rceil \cdot n)$. כידוע ב $DT(S)$ בתלת-ממד יש $O(|S|^2)$ טטרהדרונים, וסיבוכיות הבנייה היא $O(|S|^2)$ בזמן ובמקום.

סיבוכיות חישוב ה α-shape ליניארית ב $|DT(S)|$ בזמן ובמקום.

סיבוכיות הדגימה ליניארית ב n.

לכן סה"כ, סיבוכיות האלגוריתם הינה $O((\lceil M/e \rceil \cdot n)^2)$ בזמן ובמקום.

חלק 4 ניסויים ומסקנות

ניסוי 1:

תיאור הניסוי:

בניסוי זה צפיפות הדגימה קבועה ואילו ערך ה α משתנה.
יצרתי קובץ GCode למודל באמצעות Cura עם ההגדרות הבאות:

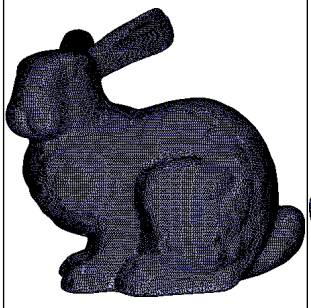
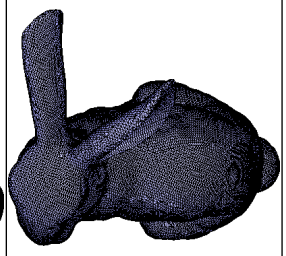
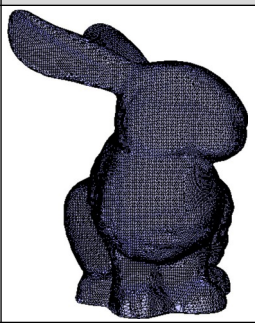
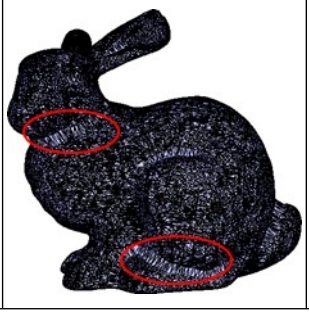
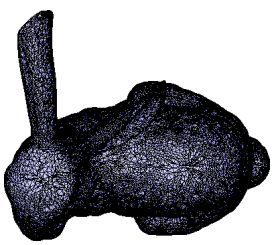
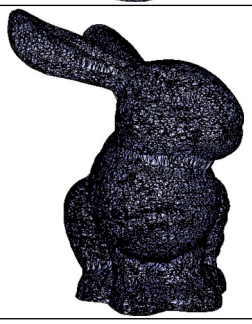
- חומר: ABS
- פרופיל: High Quality
- מילוי: Solid
- תמיכה: בלי

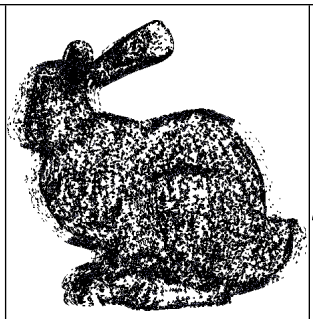
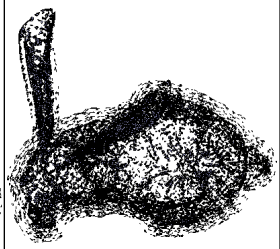
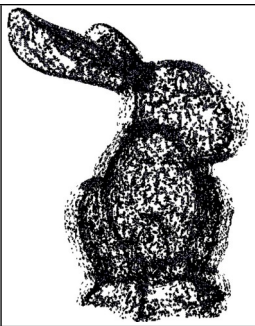
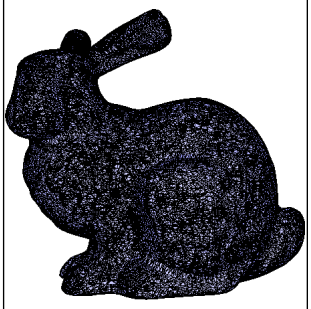
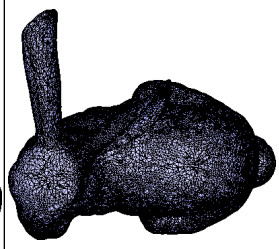
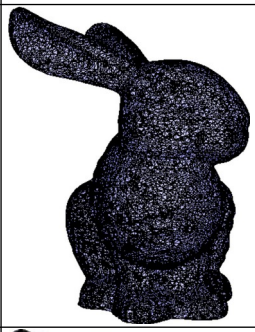
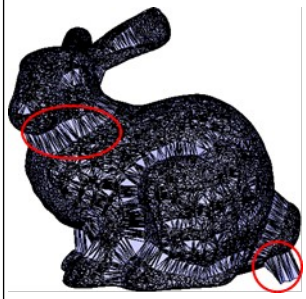
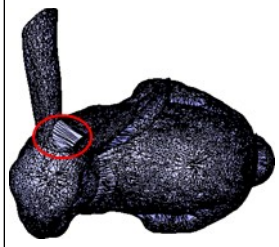
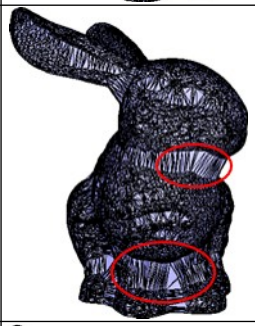
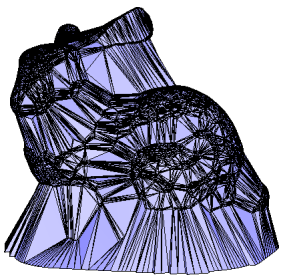
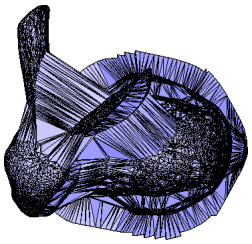
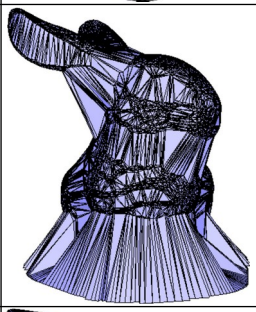
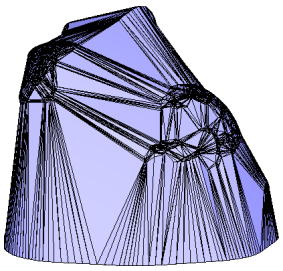
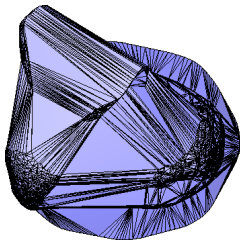
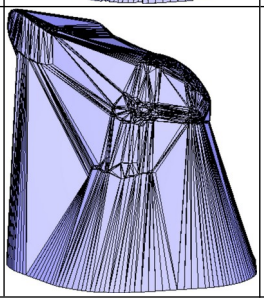
לאחר מכן שחזרתי את המודל באמצעות GCode2STL עם צפיפות דגימה קבועה וערכי α משתנים, כאשר אחד מהם הוא ערך ה α שהתוכנית מחשבת כאשר לא מסופק ערך נתון.

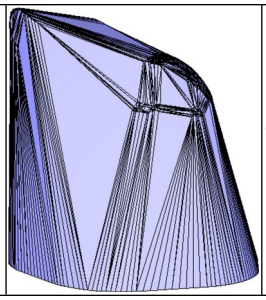
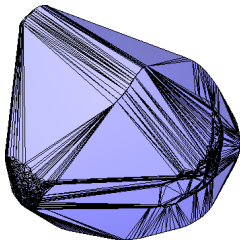
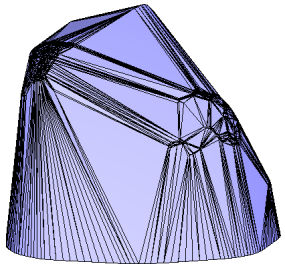
בעייתי למדוד מרחק Hausdorff בין המודל המשוחזר למודל המקורי מאחר וה GCode מבוסס על מערכת צירים אחרת, ואין הבטחה שהמודל אינו יזוז או יסובב בעקבות תהליך ה slicing. לכן, איכות השחזור נמדדה באופן ויזואלי.

המודל:
Stanford Bunny
מידות: 20 mm x 16 mm x 20 mm
צפיפות דגימה: 0.5mm

תוצאות הניסוי:

(α (mm	Side	Top	Front	הערות
מודל מקורי				
ערך שחושב ע"י התוכנה 0.25				השחזור טוב. ניתן לראות שהמשולשים אינם מסודרים כמו במודל המקורי. מצד שני, המודל אחיד. ישנם "קרומים" שנוצרו בחריצים (מסומן באדום) המצביעים על ערך α מעט גבוה מדי.

0.01				השחזור מפורר לרכיבי קשירות רבים. זה מצביע על ערך α נמוך מדי, המאפשר חדירה לתוך פנים המודל מבעד לקיר החיצוני.
0.1				השחזור טוב מאד. ערך ה α טוב מאד, לא נוצרו רכיבי קשירות נוספים המצביעים על ערך נמוך מדי ומצד שני לא נוצרו קרומים בחריצים המצביעים על ערך גבוה מדי.
1				ערך α גבוה מדי. ניתן לראות את המרווחים בהם כדור ה α לא הצליח להיכנס למרות שהיה צריך ע"פ המודל המקורי.
10				
100				

ערך α גדול מאד שלא מאפשר כניסה של הכדור לאף מרווח של המודל, ולכן מתקבל הקמור שלו.				1000

ניסוי 2:

תיאור הניסוי:

בניסוי זה צפיפות הדגימה משתנה ואילו ערך ה α קבוע.
יצרתי קובץ GCode למודל באמצעות Cura עם ההגדרות הבאות:

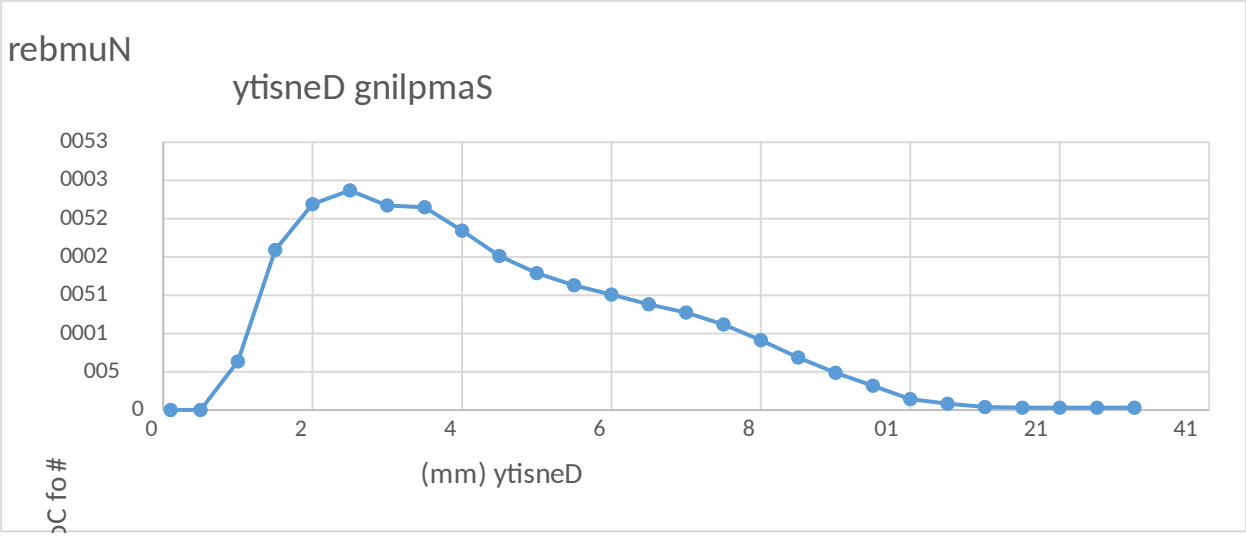
- חומר: ABS
- פרופיל: High Quality
- מילוי: Solid
- תמיכה: בלי

לאחר מכן שחזרתי את המודל באמצעות GCode2STL עם צפיפות דגימה משתנה וערך α קבוע.

בניסוי זה נמדד מספר רכיבי הקשירות במודל המשוחזר. אנו יודעים כי מספר זה אמור לצאת 1.

המודל:
Stanford Bunny
מידות: 20mm x 15mm x 20mm
α: 0.1mm

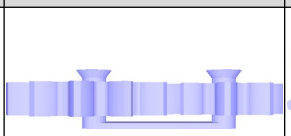
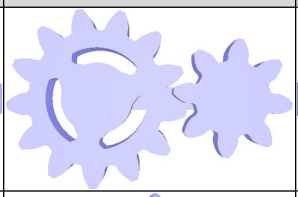
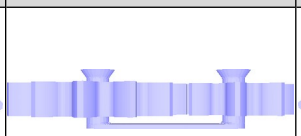
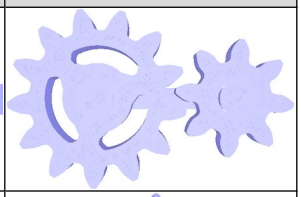
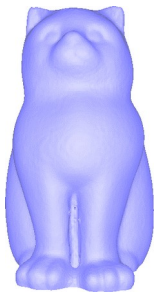
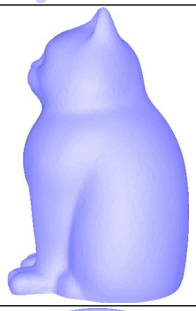
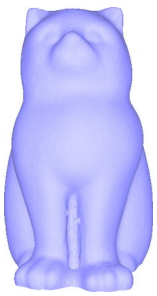
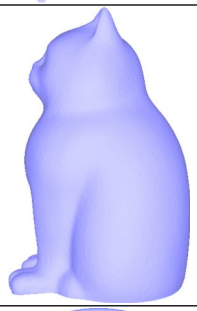
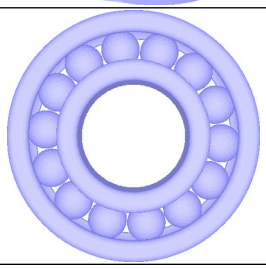
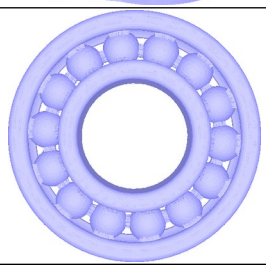
תוצאות הניסוי:



עבור ערכי α נמוכים (0.1-0.5) התקבל רכיב קשירות אחד. לאחר מכן הגדלת הצפיפות מגדילה דרסטית את מספר רכיבי הקשירות. הסיבה לכך היא שכדור ה α מסוגל לחדור למודל ולכן נוצרים מספר "גושים" מספיק צפופים של נקודות שכדור ה α אינו מצליח לחדור. אבל כשממשיכים להגדיל את הצפיפות המספר קטן עד לערך של 31. הסיבה לכך היא שה"גושים" המדוברים לא מספיק צפופים כדי להפוך לרכיבי קשירות.

ניסוי 3:

בניסוי זה שוחזרו מודלים שונים. ערכי α נבחרו באופן אוטומטי ע"י התוכנית.

Dim (mm)	Original Model		Reconstructed Model	
	1	2	1	2
x 47 30 x 10				
x 30 33 x 50				
x 50 50 x 10				

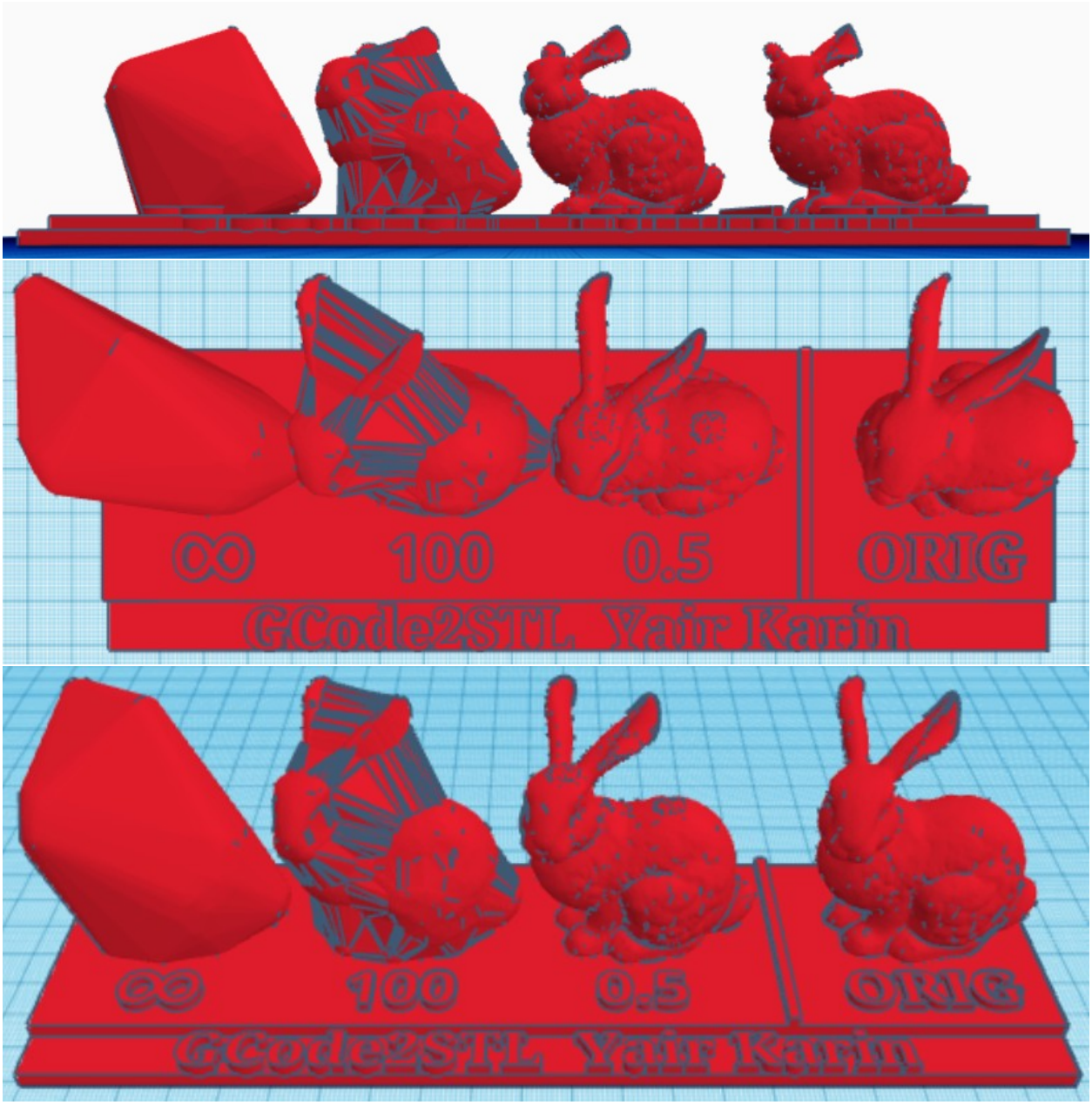
חלק 5 הדגמה מוחשית

המודל שנבחר הינו ה Stanford Bunny. יצרתי עבורו GCode באמצעות Cura.

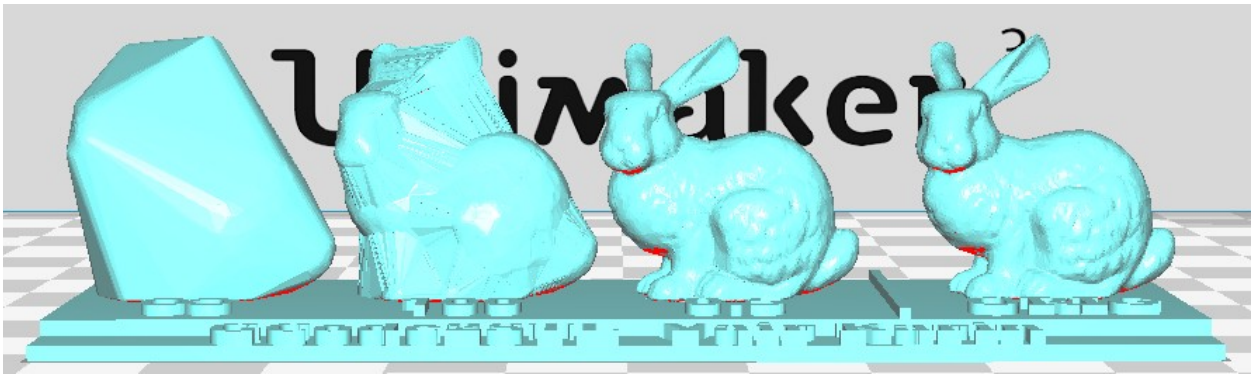
דגמתי את ה GCode בצפיפות 0.5 ושיחזרתי עם מספר ערכי α שונים: 100000 (אינסוף), 100, 0.5.

יצירת מודל להדפסה באמצעות Tinkercad:

מימין לקו ההפרדה, המודל המקורי תחת כותרת "ORIG". משמאל לקו ההפרדה, שחזורים עם ערכי α שונים.


∞
100
0.5
ORIG
GCode2STL Yair Karin

טעינת המודל ב Cura:

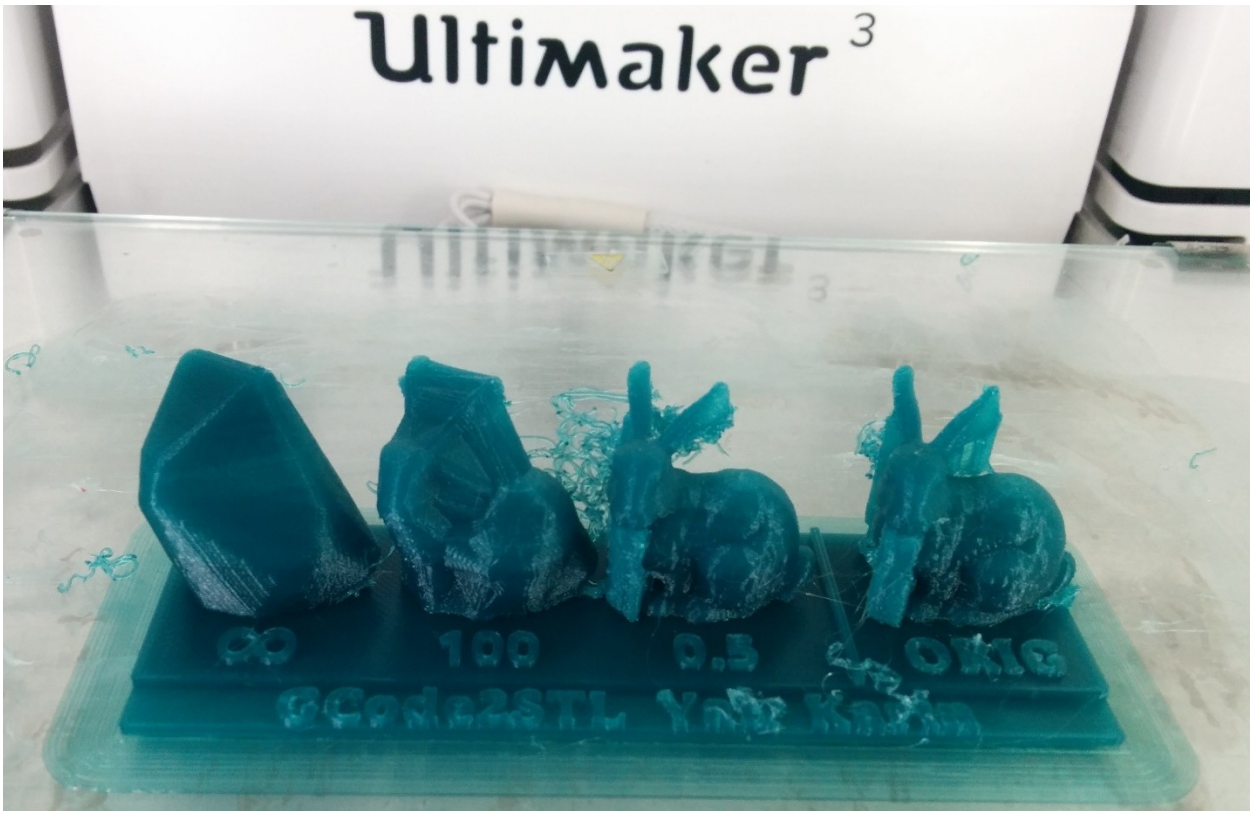


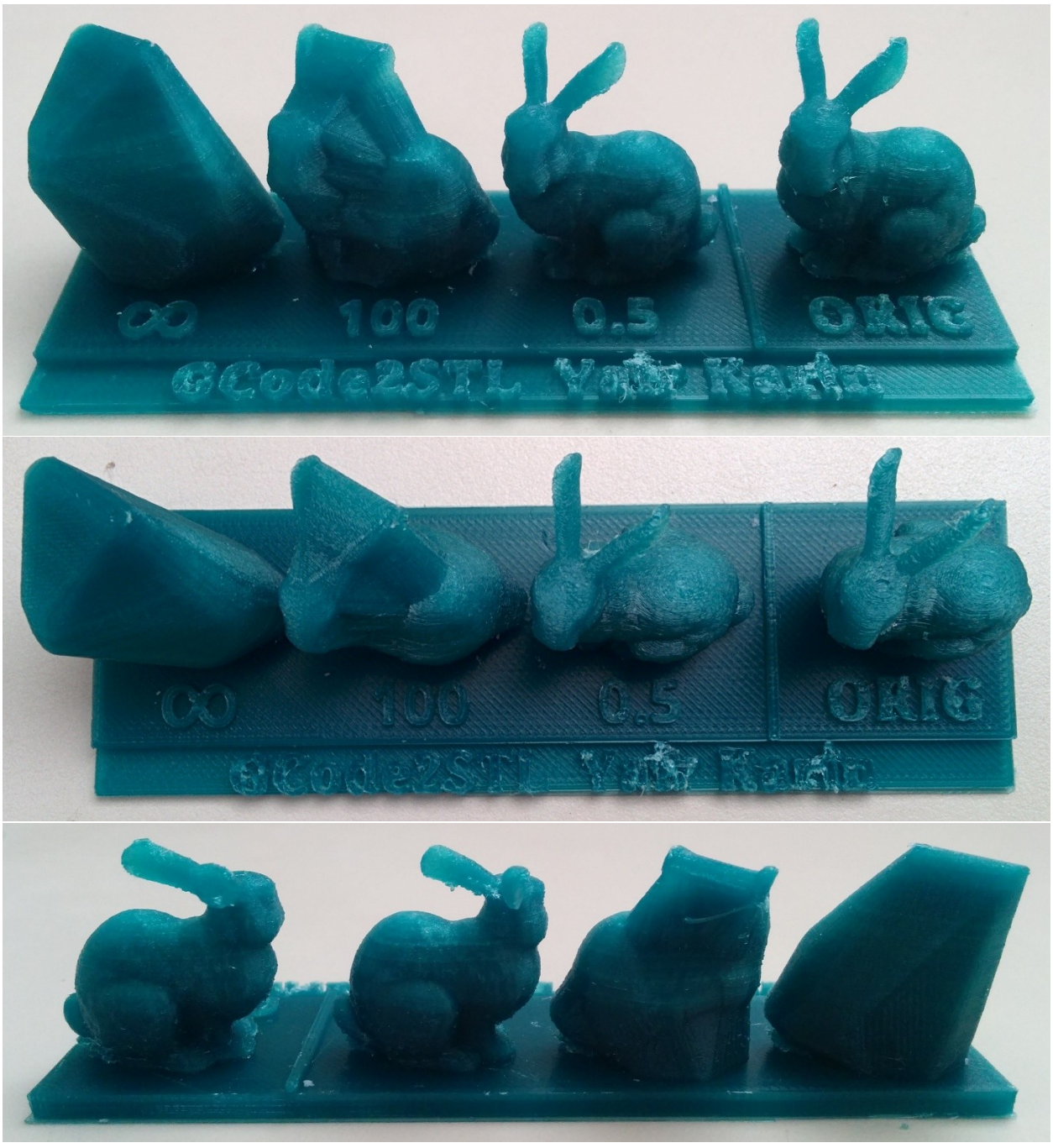
הגדרות הדפסה:

- איכות: High
- מילוי: Solid
- תמיכה: כן
- קוטר: 1.75mm

ממדי המודל: 100mm x 32mm x 26mm.

הדפסת המודל:




∞
100
0.5
OKIC
GCode2STL
∞
100
0.5
OKIC

חלק 6 הצעות לשיפור

Generalized α-shapes:

מבוסס על המאמר

On the shape of a set of points and lines in the plane.
Marc van Kreveld Thijs van Lankveld Remco Veltkamp.
https://pdfs.semanticscholar.org/15a4/66a3646625dfc58d6e4b902a04e6328a5783.pdf

עד כה התעסקנו ב α-shape של קבוצה של נקודות. ניתן להרחיב את ההגדרה לקבוצות של יצורים אחרים.

נדבר על **ממד 2 בלבד**, לא הצלחתי למצוא מספיק אינפורמציה לגבי ממד 3 ומעלה.

הגדרת Generalized α-shapes:

בהינתן קבוצת נקודות S וקבוצת ישרים L (סגמנטים \ קרניים \ ישרים), ה generalized α-shape של $S \cup L$ הינו ה α-shape של קבוצת הנקודות $S \cup \{x | x \in l \in L\}$. כלומר ה α-shape של $S \cup L$ הינו ה shape של כל הנקודות ב S וב L.

בניית Generalized α-shapes:

הבנייה משתמשת בסגמנטים בלבד (אך ניתן להרחיב את הבנייה לישרים באופן יחסית פשוט).

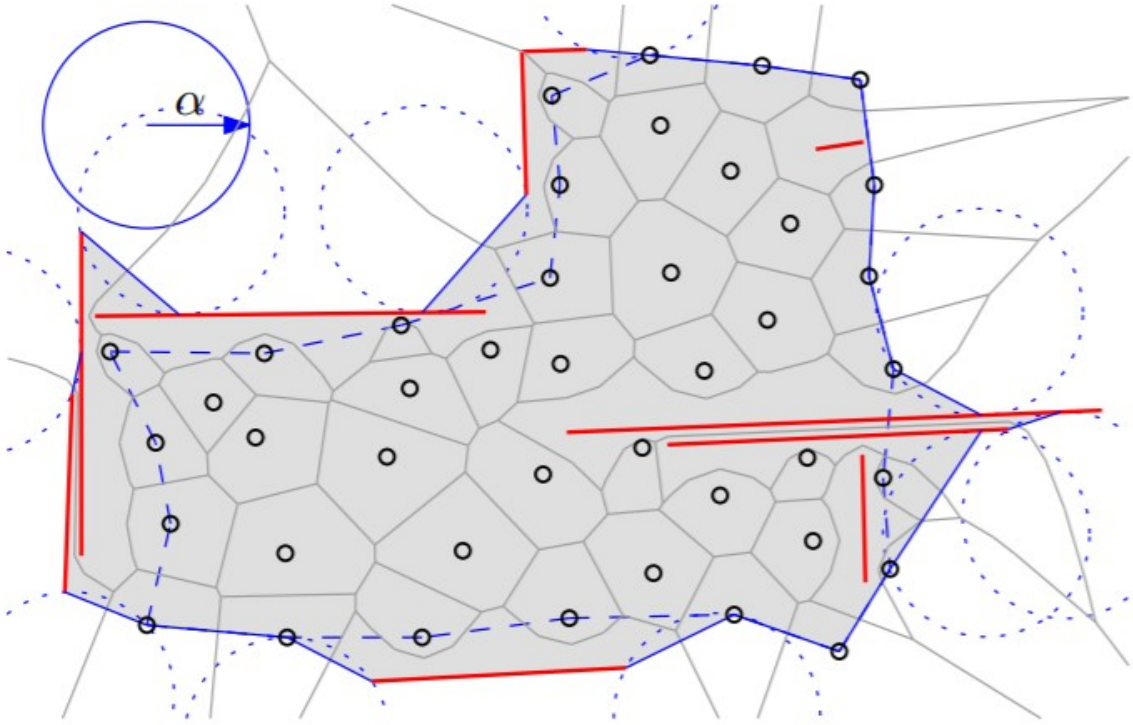
קונספטואלית, generalized α-shapes מתייחס לסגמנטים בתור אינסוף נקודות שיש להוסיף לקבוצת הנקודות. בפרקטיקה, המעטפת של ה α-shape בנויה ממספר צלעות לינארי במספר הנקודות n ומספר הסגמנטים m וניתנת לבניה בזמן $O((n+m)\log(n+m))$ ובמקום $O(n+m)$.

אבחנה חשובה להבנת הסיבוכיות היא שיש מספר נקודות מוגבל על הסגמנטים שמשפיעות על המעטפת. כל הדיסקים הריקים (α-balls) עם נקודה מסגמנט על ההיקף שלהם חייבים להשיק לסגמנטים או שהדיסק נוגע בנקודת קצה של הסגמנט.

תהליך הבנייה:

ל generalized α-shape יש קשת בין כל זוג נקודות על דיסק ריק ברדיוס α. ניתן לחלק את הקשתות האלה ל 3 קטגוריות: נקודה-נקודה, סגמנט-נקודה, סגמנט-סגמנט.
המקרה הראשון זהה לחישוב α-shape רגיל, עם התוספת שלדיסק אסור לחתוך סגמנט בבפנים.
במקרה של קשת המחברת בין נקודה לאינסוף נקודות המיוצגות ע"י סגמנט s, הקשת תחול בנקודה $p \in S$ כך ש p נקודת קצה של הסגמנט או נקודת השקה של הדיסק עם s.
במקרה של קשת המחברת בין שני סגמנטים, שתי הנקודות יכולות להיות נקודות קצה של סגמנט או נקודות הקשה של סגמנט עם דיסק.
בכל 3 המקרים מרכז הדיסק במרחק α מ 2 קודקודי הקשת, אינפורמציה זו מוכלת ב medial-axis בדיאגרמת ורונוי של $S \cup L$.
בנייה של דיאגרמת ורונוי של n נקודות ו m סגמנטים הינה בזמן $O((n+m)\log(n+m))$ ובמקום $O(n+m)$.

בתמונה למטה, העיגולים השחורים הינם נקודות S והסגמנטים האדומים הינם בסגמנטים ב L. הקו המקווקו הכחול הינו ה α-shape המקורי והקו הכחול הרציף הינו ה α-shape בהתחשב בסגמנטים ב L.



מהות השיפור:

ה GCode toolpath מורכב מרצף של סגמנטים. בפתרון שלי נדגמו נקודות מכל סגמנט בצפיפות נתונה. Generalized α-shapes חוסך את דגימת הנקודות, במקום לדגום סגמנטים בצפיפות נתונה ניתן לחשב את ה α-shape ישירות על הסגמנטים. זה יועיל בדיוק ויחסוך את התלות בצפיפות הדגימה.

הערות ב GCode:

GCode עשוי להכיל הערות שעשויות לעזור בשחזור המודל.

```
;TYPE:SKIRT
;TYPE:WALL-INNER
;TYPE:WALL-OUTER
;TYPE:SKIN
```

לא מצאתי תיעוד בנושא אך ניתן לנחש כי אפשר לחלץ באמצעותן את הקירות החיצוניים של המודל (TYPE:WALL-OUTER) ובכך לפשט את השחזור ולמזער משמעותית את כמות הנקודות בדגימה (לא לדגום בבפנים של המודל).

נספחים

GCode2STL, הוראות שימוש:

תיקיות:

מצורפות 3 תיקיות:

- gcode2stl קוד מקור
- meshconv כלי להמרה בין פורמטים של מודלים תלת ממדיים (stl->off וכו')
- input מודלים וקבצי gcode שלהם

קומפילציה:

התיקיה gcode2stl מכילה קובץ CMakeLists.txt המשמש כקלט לתוכנת CMake. יש להשתמש בתוכנה על מנת ליצור פרויקט שאותו ניתן יהיה לקמפל.

GCode2STL:

כאשר התקבל קובץ הרצה מקופל, ניתן להריצו משורת הפקודות עם הדגל -help או -h על מנת לקבל את תיעוד הפרמטרים כדלקמן:

Options:

-h [--help]	Help screen
-i [--input] arg	Input .gcode file
-o [--output] arg	Output .off file
-e [--density] arg (=0.5)	Density for GCode toolpath point sampling, the distance between two sampled points
-a [--alpha] arg	The alpha value for the Alpha-Shape algorithm, if not supplied will be heuristically calculated
--output-xyz arg	Output .xyz file of sampled points
--output-alpha-xyz arg	Output .xyz file of points on alpha shape boundary

עבור מודלים גדולים, צפיפות דגימה גבוהה משמעותה מספר גבוה של נקודות וזמן ריצה ארוך. **ככל שהצפיפות קטנה יותר, תוצאות השחזור טובות יותר**. את הבדיקות ביצעתי על מודלים בסדר גודל של סנטימטרים בודדים בכל ממד ובצפיפות של 0.1-0.5. כאשר ערך alpha אינו נתון התוכנית מנסה להעריך אותו באותו האופן שמתואר בסעיף "חישוב ערך α".

שורות הרצה לדוגמה:

שחזור מודל מתוך input_file.gcode לתוך output_file.off עם ערכי alpha ו density דיפולטים:

./gcode2stl --input=input_file.gcode --output=output_file.off

שחזור מודל מתוך input_file.gcode לתוך output_file.off עם density=0.5 וערך alpha המחושב אוטומטית:

./gcode2stl --input=input_file.gcode --output=output_file.off --density=0.5

שחזור מודל מתוך input_file.gcode לתוך output_file.off עם density=0.5 ו alpha=0.1:

./gcode2stl --input=input_file.gcode --output=output_file.off --density=0.5 --alpha=0.1

Meshconv:

כלי להמרה של קבצים בין פורמטים שונים של מודלים תלת-ממדיים.
אני השתמשתי בו כדי להמיר קבצי off ל stl מכיוון ש CGAL לא יודעת לכתוב פוליהדרון ישירות ל stl.

שורת הרצה לדוגמה, המרת קובץ off לקובץ stl:

./meshconv input_file.off -c stl -o output_file

באגים:

את התוצאות הטובות ביותר קיבלתי כאשר השתמשתי באיכות הדפסה high ובמילוי solid. היה מודל מסוים בו ניסיתי איכות הדפסה normal ומילוי dense, וכתוצאה מכך Cura יצר חור יחסית גדול בתחתית המודל. זה השפיע לרעה על התנהגות ה shape-α מכיוון ש α קטן מרדיוס החור משמעותו כניסת כדור ה α לתוך המודל.

באג נוסף שנתקלתי בו היה כאשר שכחתי לכבות את השימוש בתמיכה (support) ב Cura. כתוצאה מכך ה GCode toolpath הכיל תמיכה שמנעה שחזור מוצלח של המודל המקורי, אשר מן הסתם לא הכיל תמיכה.