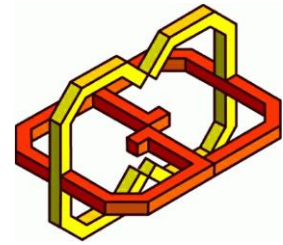




Leap Motion Matrix



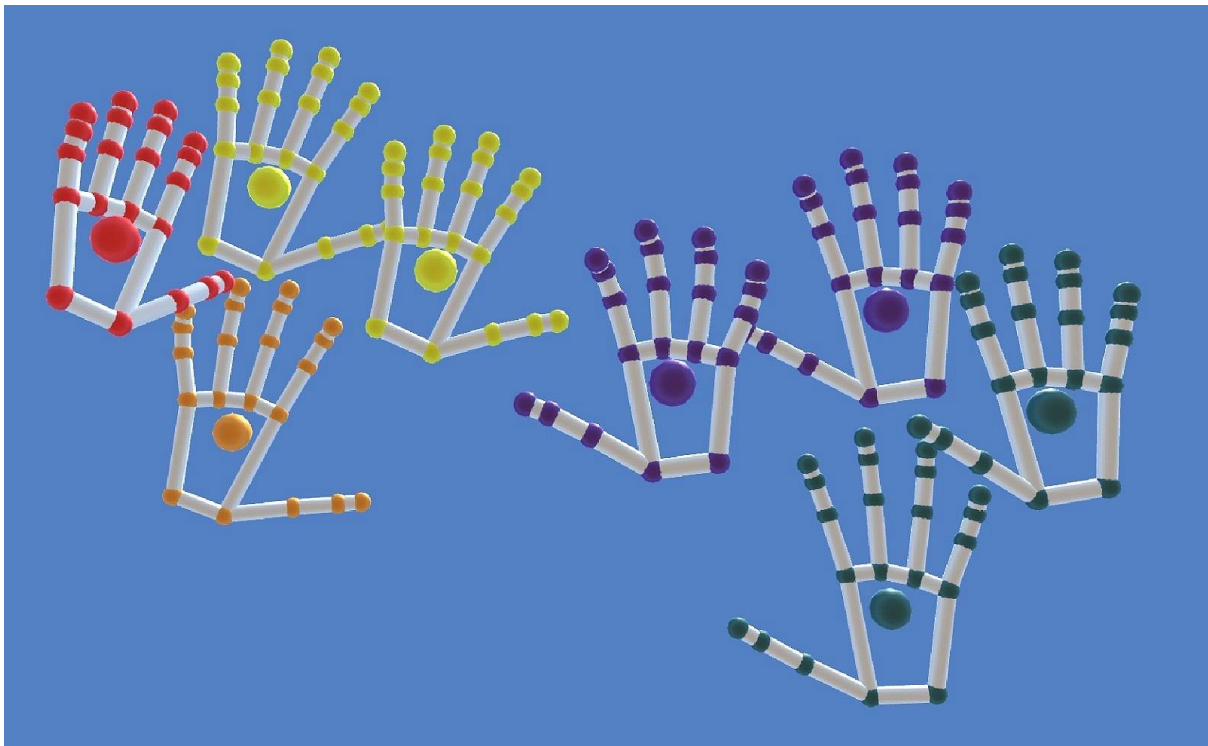
## Leap Motion – Multiple Device

# Increasing the field of work for desktop apps, using multiple devices synchronization

**מגישים:** דניאל מינס, שי דדון

**מנחים:** ירון חונן, בועז שטרנפלד

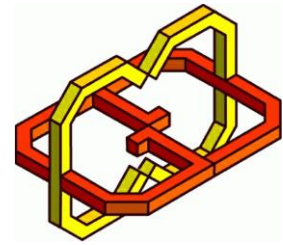
**יועץ:** אלון זבירין



אוגוסט 2019



Leap Motion Matrix



## תוכן עניינים

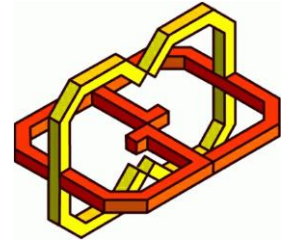
---

- 2 ----- תוכן עניינים •
- 3 ----- מבוא ותקציר •
- 4 ----- תיאור המערכת •
- 6,5 ----- תיאור הביצוע •
- 7 ----- תוצאות •
- 8 ----- המלצות •





Leap Motion Matrix



## מבוא ותקציר

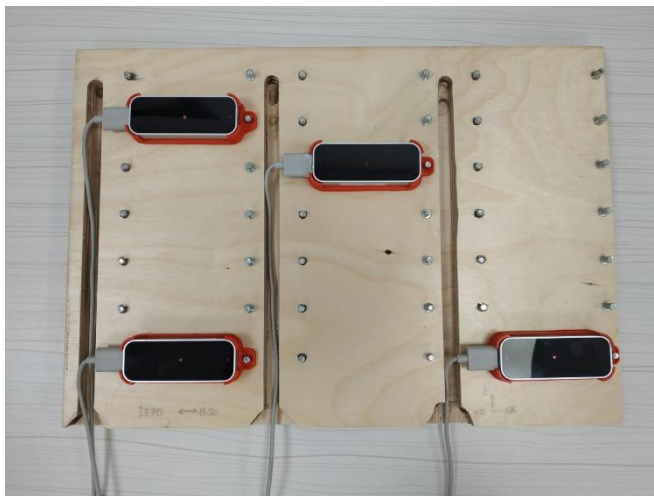
פרוייקט זה עוסק בסנכרון מספר מצלמות [Leap Motion](#), על מנת להגדיל את תוֹך העבודה האפשרי שמצלמה אחת מספקת תוך כדי ניסיון לשמר את איכות ביצועי המצלמה. פרוייקט זה נולד כתוצאה מהוצאת גרסת בטא חדשה של Leap Motion המאפשרת שימוש במספר מצלמות במקביל ובמחשב יחיד.

בפרוייקט זה השתמשנו בעד 4 מצלמות Leap Motion. על מנת ליצור סנכרון מדוייק ככל האפשר של המצלמות היינו צריכים להכין מטריצה שעליה ניתן להלביש את המצלמות בדיוק גבוהה של מיקומם.

בתחילת הפרוייקט התמקדנו בניסיון להגדיל את תוֹך הראיה של המצלמות. תחילה השתמשנו באלגוריתם Kabsch על מנת לבצע כיוול דינמי של המצלמות באמצעות פריימים שבהם כל המצלמות רואות את היד. כיוול זה הניב תוצאות טובות אך לא מדויקות מספיק לצרכים שלנו לכן החלטנו לבנות את המטריצה שבה נוכל להכין טרנספורמציה קבוע ומדויקת יותר. באמצעות מטריצה זאת, השימוש בטרנספורמציות ושימוש ב-4 מצלמות הצלחנו להגדיל את טווח הקליטה לעומת מצלמה אחת בכפי 4 (ליניארית בכמות המצלמות). בשלב זה עדיין נראו 4 זוגות ידיים שהיו ממוקמים בקירוב אחת על השנייה.

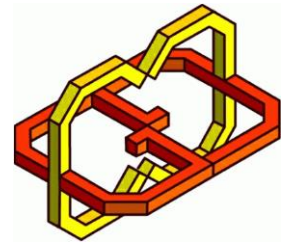
בשלב הבא התמקדנו בהצגת יד יחידה על המסך, בשלב זה החלטנו לפעול בגישה האומרת שהמצלמה הקרובה ביותר ליד היא זאת שרואה הכי טוב את היד ותיתן דיוק הכי גבוהה. בסוף שלב זה הגענו למצב שהמערכת מציגה רק זוג ידיים אחד בכל רגע אך עדיין קיימת בעיה והיא בעיית "קפיצה" שבה במעבר בין אזור שבו מצלמה א' נבחרת לאזור שבו מצלמה ב' נבחרת מיקום היד קופץ קרוב יותר לכיוון המצלמה הנבחרת במקום תנועה חלקה.

בשלב האחרון ניסינו גישות שונות על מנת לפתור את בעיית הקפיצה: גישה אחת כללה שמירה של פריים אחורה ושימוש בממוצע עם הפריים החדש על מנת לקבל "תמונת מעבר", דבר שלא היה משמעותי מבחינה וויזואלית. ניסינו גישות נוספות הכללו שמירה של מספר ממוצאי פריימים אחורה על מנת להחליק את ה"קפיצה" אך לבסוף השיטה המוצלחת ביותר הייתה בנייה עצמאית של פריימים המתאימים למעבר והצגתם על מנת למתן את הקפיצה.





Leap Motion Matrix



## תיאור המערכת



### ציוד:

- 4 מצלמות Leap Motion.
- 4 תבניות שהודפסו במצלמת תלת מימד על מנת לשמש כ- case למצלמות.



- מטריצה קשיחה שאפשר להציב בה את המצלמות מבלי שיזוזו במיקומים שונים.

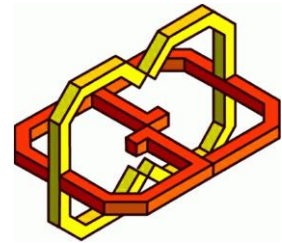
### תוכנה:

- Unity 2018.1.0f2
- [LeapDeveloperKit 4.0.0+52236](#) – קישור ל-GitHub, מכיל build התומך ב-MultipleDevice וגם Assests המתאימים ל-Unity.





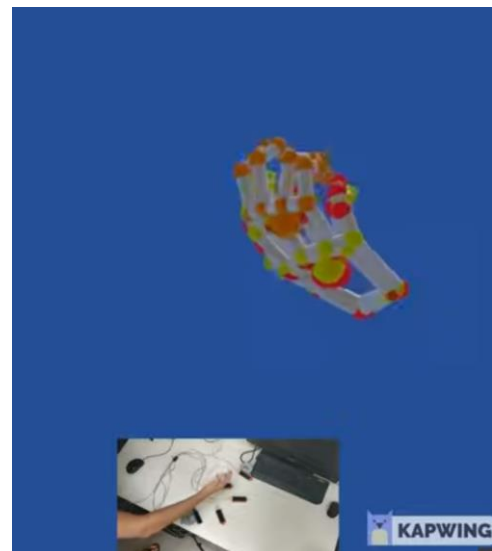
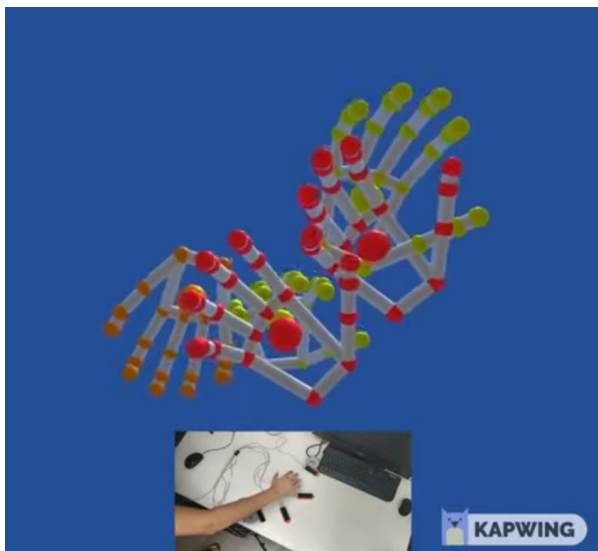
Leap Motion Matrix



## תיאור הביצוע

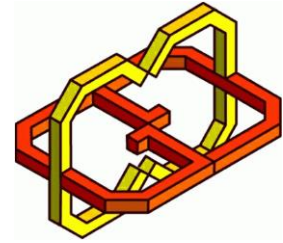
**שלב א' - הגדלת תווך המצלמות באמצעות סנכרון:**

- שימוש באלגוריתם Kabsch על מנת לסנכרן את המצלמות, זאת באמצעות לקיחת תמונות שכל המצלמות רואות את הידיים והפעלת טרנספורמציות עליהן על מנת לעבור למימד אחיד לכולם. Kabsch: אלגוריתם זה מוצא את מטריצת הרוטציה שממזערת את סטיית השורש בריבוע (root mean squared deviation) בין שני סטים של נקודות. כאשר נעשה גם חישוב של העתקה (translation) האלגוריתם נקרא Procrustes superimposition, אך לרוב משתמשים בשם Kabsch בתחום מדעי המחשב.
- כחלק מ- assets ששחררו היה גם תמיכה באלגוריתם Kabsch על מנת לסנכרן בין המצלמות. כעבור מספר ניסיונות הבנו שקיים באג בקוד ששחררו אשר לא אפשר תמיכה ביותר משני מצלמות, באג זה תוקן על ידינו.
- סנכרון זה לא צלח מספיק וזאת מכיוון שרמת הדיוק תלויה במספר התמונות ותלויה בחוסר תזוזה של המצלמות ולכן החלטנו להכין מטריצה שעליה ישבו המצלמות ללא תזוזה.
- הכנת ה-case למצלמות באמצעות מדפסת תלת המימד.
- הכנת המטריצה.
- כעת כאשר יש לנו מטריצה שבה המצלמות לא יכולות לזוז הטרנספורמציה הינה מדויקת (עד כדי זוויות קטנות שנוצרות מלחץ על קבל ה-usb) והצלחנו להגיע למצב שבו הרחבנו את התווך בכפי 4 לעומת מצלמה אחת(ליניארית בכמות המצלמות) אך עדיין רואים 4 זוגות ידיים אחת על השנייה.





Leap Motion Matrix



**שלב ב' – הצגת יד יחידה על המסך:**

- על מנת להציג יד יחידה על המסך החלטנו להציג את היד כפי שרואה המצלמה הקרובה ביותר ליד בכל רגע. החלטנו כך מכיוון שלמצלמות יש תווך שבהן הן מפסיקות לעבוד בצורה מדויקת ומתחילות להחזיר קירובים של היד, לעומת זאת במרחקים קרובים היד מדויקת הרבה יותר.
- לאחר שהצלחנו להראות רק זוג ידיים אחד ששטח העבודה שלהם גדל בהרבה נתקלנו בבעיה של "קפיצת" היד, מצב בו במעבר בין אזור של מצלמה אחת לאזור של מצלמה שנייה היד משתגרת טיפה לכיוון המצלמה השנייה, קפיצה זאת מתרחשת מכיוון שהמצלמות לא מספיק מדויקות והן אף משלימות באמצעות אלגוריתמים פנימיים חלקים ביד שלא נראים בכלל או לא מספיק טוב ולכל אחת מהמצלמות השלמה קצת שונה.

**שלב ג' – פתרון בעיית הקפיצה – החלקה של מעבר היד בין אזורים של מצלמות שונות:**

- בשלב זה נתקלנו בבעיה שאנו לא בטוחים בהסבר שלה: שימוש בפונקציית הספרייה של Unity.Vector3 הניבה תוצאות לא נכונות, החזירה 0, עבור מרחקים קטנים סדר גודל של (0.01,0.01,0.01). אנו חושבים שזה מתרחש מכיוון שכנראה נעשה שימוש ב-float אשר יוצר שגיאה נומרית בחישוב המרחק או שפשוט מזניחה את סדר הגודל של  $0.01^2$ . כאשר הכנו פונקציית מרחק משל עצמנו אשר תחילה ממירה את ערכי הווקטורים ל-double ורק לאחר מכן מבצעת את החישוב קיבלנו תוצאות נכונות.

- ניסינו מספר גישות לפתרון הבעיה:

1.  $New Frame = Avg\{Prev Frame, Current Frame\}$  – במקום להציג את הפריים הנוכחי הגדרנו פריים חדש שהוא ממוצא הפריים הנוכחי עם הפריים הקודם, פתרון זה לא תרם וזאת מכיוון שהמעבר קורה בפריים יחיד והעין האנושית לא תוכל אפילו להבחין בשינוי שעשינו.

$$2. New Frame = Avg_{\beta}\{Prev Frame, Current Frame\} = \beta(Prev Frame) + (1 - \beta)(Current Frame)$$

בשיטה זאת ניסינו לתת משקל  $\beta$  משתנה למוצע, שיטה זאת כמובן גם לא צלחה מאותה סיבה כמו מקודם.

3. כעת הבנו שאי אפשר להתייחס לבעיית הקפיצה כמעבר חד אלא צריך לתת הדרגתיות על מנת "להחליק" את המעבר. בדקנו מהי ההגדרה המספרית של "קפיצה" זאת אומרת מדדנו את המרחק הממוצע בין שני פריימים כאשר יש קפיצה ואת המרחק הממוצע ללא קפיצה וקיבלנו שבממוצע הקפיצה היא בסדר גודל של 2-3 ס"מ בעולם האמיתי.

על מנת ליצור מעבר חלק יותר החלטנו שבכל מעבר אנחנו נכניס בין לבין עשרה פריימים שאנחנו ניצור אשר מקיימים:

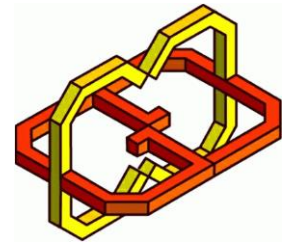
$$Frames = \{Avg_{\beta}\{Prev Frame, Current Frame\} | \beta = \{0.1, 0.2, \dots, 1.0\}\}$$

ונציג אותם בזמן המעבר, כמות פריימים זאת נבחרה אחרי מספר ניסיונות בכמויות גדולות יותר אשר גרמו להאטה רגעית של המערכת בזמן קפיצה וגם היו נראים יותר מידי זמן לעיין מכיוון שהתרחשו למשך מספר פריימים רב.

- גישה 3 הייתה המוצלחת ביותר מכל הגישות.



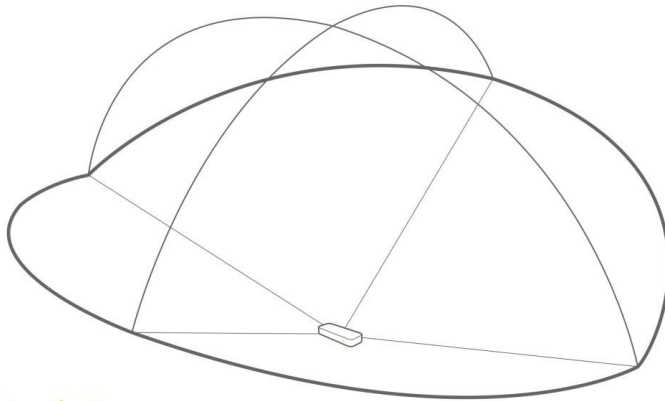
Leap Motion Matrix



## תוצאות

שיפור התווך של המצלמה מוצג בשני הסרטונים הבאים:

- [סרטון](#) ראשון מדגים ביצועים עם מצלמה אחת.



### Interaction Area

2 feet above the controller, by 2 feet wide on each side (150° angle), by 2 feet deep on each side (120° angle)

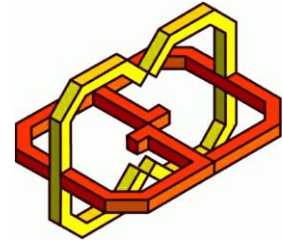
- [סרטון](#) שני מדגים ביצועים עם 4 מצלמות.

שיפור זה מגדיל את הטווח ליניארית במספר המצלמות.

- [סרטון](#) המדגים את המעברים בין מצלמות. כל מעבר מוצג כהחלפת צבע של היד.
- [סרטון](#) המציג את הכיול הדינמי.



Leap Motion Matrix



## המלצות

- שיטה נוספת שחשבנו להחלקת הקפיצה הינה שלזמן מסוים לאחר קפיצה לתת משקל משתנה לפריים הקודם בממוצע אשר עם הזמן יורד, שווה ערך ללומר שהתמונה הקודמת נכונה יותר מהחדשה לזמן מסוים ולאחריו התמונות החדשות מתחילות להיות נכונות יותר.
- בעבודה עם מצלמות Leap Motion לקחת בחשבון שהמטרה העיקרית של המצלמה היא להיות Responsive ולא כמה שיותר מדויקת ולכן יש וויתר על הדיוק.
- לכייל את המערכת לא באמצעות הנתונים שמגיעים על מיקומי הידיים מכיוון שנתונים אלה עוברים כבר באלגוריתם של המצלמה שמבצע השלמות של חלקים ביד שאינו רואה והשערות. עדיף לבצע זאת באופן מדויק יותר באמצעות חפץ שלא עובר עיבוד מקדים של המצלמה.
- להתקדם עם הפרוייקט גם למימד הנוסף (ציר Z). בסרטון האחרון שהצגנו הראנו POC עבור אפשרות 12.