דו"ח מסכם פרויקט גמר:

Continual-NeRF-Project

אבישי עוז

avishaioz04@gmail.com

ניר סגל

nirsegal2004@gmail.com

מנחה – סיימון קורמן

קישור לגיט

אוניברסיטת חיפה

החוג למדעי המחשב

תוכנית אתגר 2022-2023

אוניברסיטת חיפה
University of Haifa
جامعة حيفا

תוכן עניינים

מה זה NeRF בקצרה: 3

הגדרת הבעיה: 4

הגדרת הפתרון: 6

פירוט הפתרון שלנו (מימוש): 8

התקנת הפרויקט והרצה: 9

מה למדנו ומה ידענו? 9

המשך הפרויקט בעתיד: 10

צוואר הבקבוק שלנו בפרויקט 10

תוצאה 10

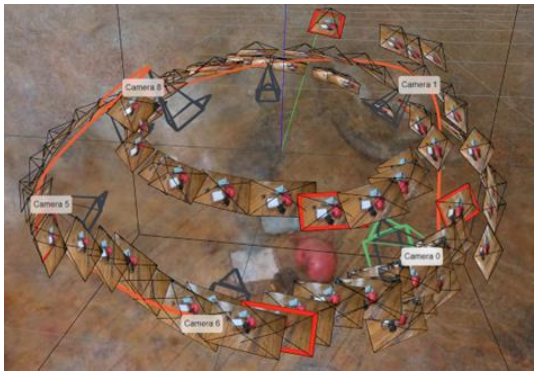
מה זה NeRF בקצרה:

$NeRF$ הוא מודל המקבל בקלט אוסף תמונות המתארות סצנה מסוימת ומייצר מודל תלת ממדי עבור סצנה, כלומר על ידי אוסף תמונות סופי מסוגל ה-*nerf* לייצר תמונות מזוויות חדשות שבכלל לא צולמו.

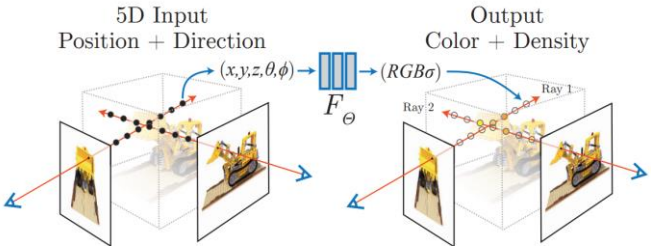
איך הוא עובד בכמה מילים:

המודל מקבל בקלט קואורדינטות של נקודות במרחב והזוויות של המצלמה שממנה צולמו תמונות (עבור כל תמונה נתונים שונים) ומחזיר את הערכה שלו לצבע הפיקסל והצפיפות, כלומר הרשת מקבלת בקלט מספר פרמטרים:

1. $X = (x, y, z)$ קורדינטות $3D$ למרחב
2. (Θ, ϕ) $2D$ כיוון צפייה



וכפלט הרשת מחזירה: $c = (r, g, b)$ צבע הפיקסל, σ – צפיפות הפיקסל

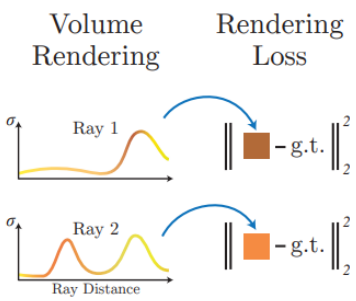


לאחר שהמודל העריך כל פיקסל נוכל לבקש ממנו לייצר תמונה חדשה, כל תמונה מיוצגת על ידי אוסף קרניים היוצאות מהמצלמה כאשר הם מיוצגות ע"י: $r(t) = o + td$

כאשר d הוא ווקטור הכיוון, o הוא מיקום המצלמה, t הוא מיקום הפיקסל על הקרן ו$r(t)$ הוא מיקום הפיקסל במרחב.

לסיכום, על מנת ליצר תמונה חדשה אנחנו נסתכל על כל פיקסל בקרן שנשלחת לעבר האובייקט ולפי הערכים לאורך הקרן נקבע את ערכי הפיקסל בתמונה.

נרצה לציין, כי פונקציית ה-loss של המודל, מבוססת על MSE של צבעי הפיקסלים האמיתיים לבין האלו שמסונתזים.

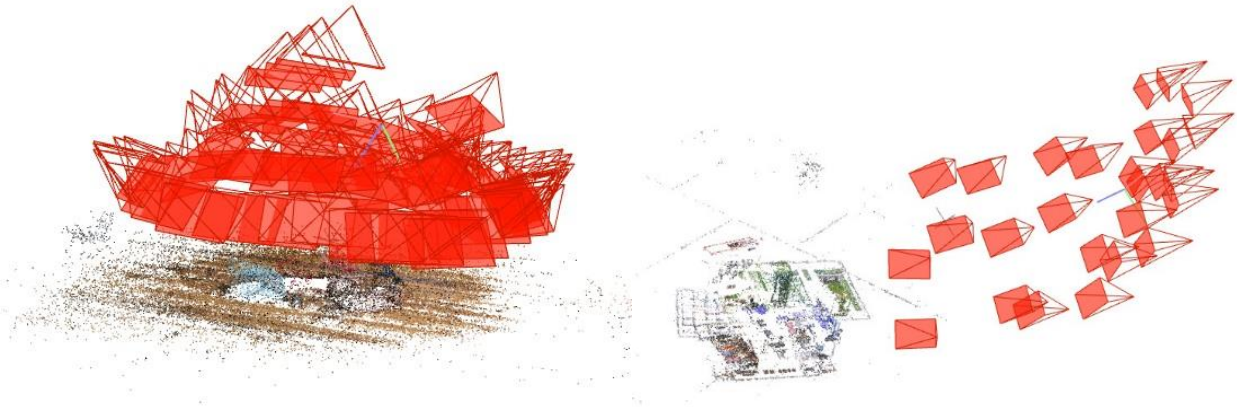


הגדרת הבעיה:

כיום קיימים סוגים רבים של NeRF-ים ולכולם יש בעיה משותפות שבה אנו נרצה להתעסק.

על מנת שה-NeRF ידע לבנות מודל של סצנה מסוימת הוא מבקש לקבל סט תמונות של הסצנה ובנוסף הוא דורש קובץ הנוצר על ידי כלי בשם COLMAP הנותן את מיקומי המצלמות במרחב.

(בתמונה מתואר הפעלת COLMAP על Data שצילמנו כדי להמחיש את הבעיה)



סרטון המציג דוגמה למודל ש-NeRF יצר על הדאטה שלנו.

עניין אותנו לבדוק איך יגיב ה-NeRF כאשר נאמן אותו על סט תמונות של סצנה מסוימת ובמהלך האימון יחול שינוי בסצנה וכתוצאה מכך בתמונות שהוא מתאמן עליהם.

כפי שניתן לראות ב2 התמונות: מצד ימין צולמה סצנה ובה קופסת תיונים ובצד שמאל אותה הסצנה ללא הקופסא.

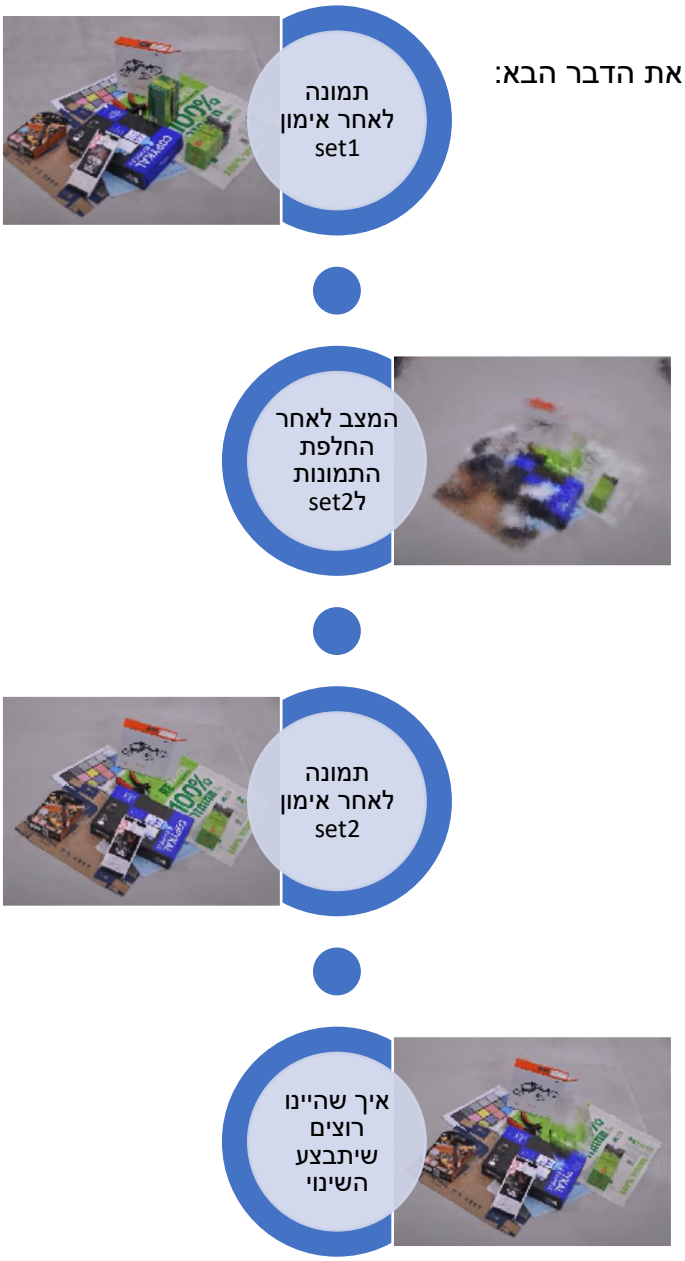
מצאנו שכאשר חל שינוי אפילו קטן כמו בדוגמה שניתן לראות לעיל יש קפיצה משמעותית בשגיאת הלמידה (ב-loss) וכל התמונה המודל הופך "למורעש" ונלמדת מחדש.

תמונה המתקבלת בלמידה ברגע המעבר בין הסטים השונים, נשים לב כי רוב הסצנה אינה שונה אולם כל התמונה משתבשת

סרטון המתאר NeRF שהתאמן על Set1 המתואר בתמונה מימין ולאחר מכן המשיך את האימון על Set2 המתואר בתמונה משמאל.

הבעיה שאנו רוצים לפתור היא האופן שבו השינוי בסצנה משתלב בלמידת המודל, המטרה שלנו היא להכניס את השינוי בצורה חלקה ככל שניתן ולמנוע את הרעש על התמונה כולה, כלומר להצליח להשפיע על המודל רק באזור שבו חל השינוי.

בעצם למנוע את הדבר הבא:



הגדרת הפתרון:

את ההסבר על הפתרון שלנו נחלק ל2 שלבים:

שלב הראשון: לזהות את האזור שהשתנה בסצנה,

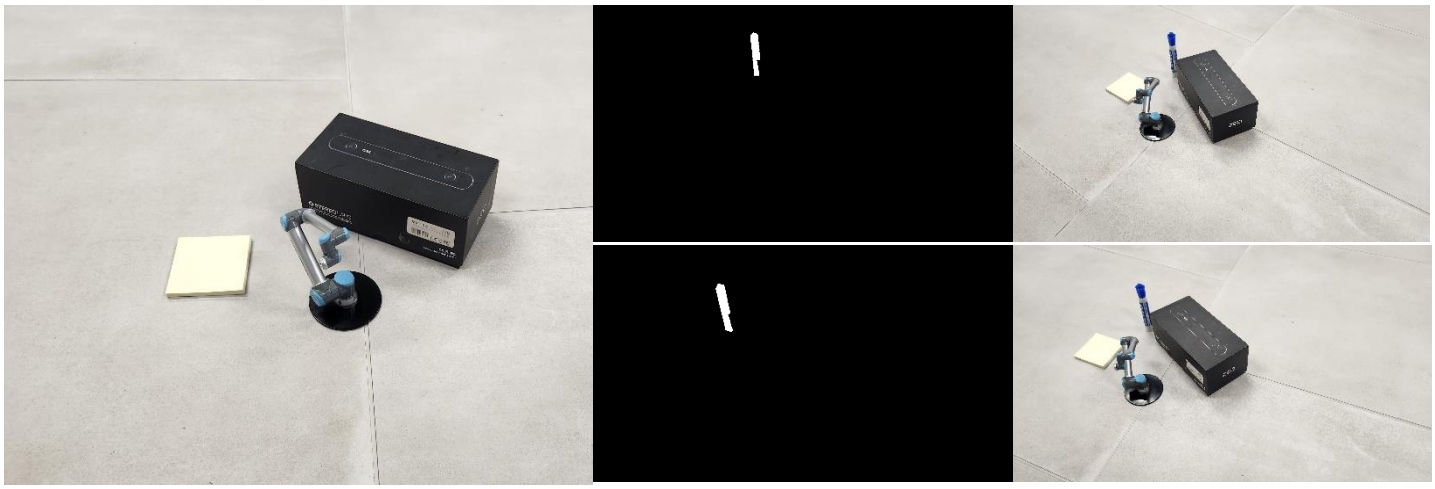
לאחר שהמודל התאמן על הסצנה המקורית הוא ידע לסנטז (לייצר) תמונות שלא קיימות בדאטה כלומר להוציא תמונה מזווית שלעולם לא צולמה.

נשתמש ביכולת הזאת באופן הבא, ניקח את מיקומי המצלמות של הסט השני (הסט לאחר השינוי) ונבקש מהמודל לסנטז את התמונה התואמת למיקום המצלמה ובכך נקבל 2 תמונות ממיקומים זהים, כך שהתמונה הראשונה צולמה באמת ושייכת לסט השני (עם השינוי) ותמונה שנייה שקיבלנו על ידי המודל ומתאימה לסט הראשון.

דוגמה ל2 תמונות זהות ממיקומים שונים: (סרטון שסונתז על ידי NeRF)

לאחר שנקבל את 2 הנקודות נמצא את הפיקסלים השונים בין התמונות ונבנה מסיכה בינארית כך שהפיקסלים השונים יהיו 1 והפיקסלים הזהים יהיו 0 ובכך יצרנו מסיכה שמסווגת איזה אזור שונה בין התמונות.

דוגמה להצגת ההבדל בין 2 תמונות עם שינוי:

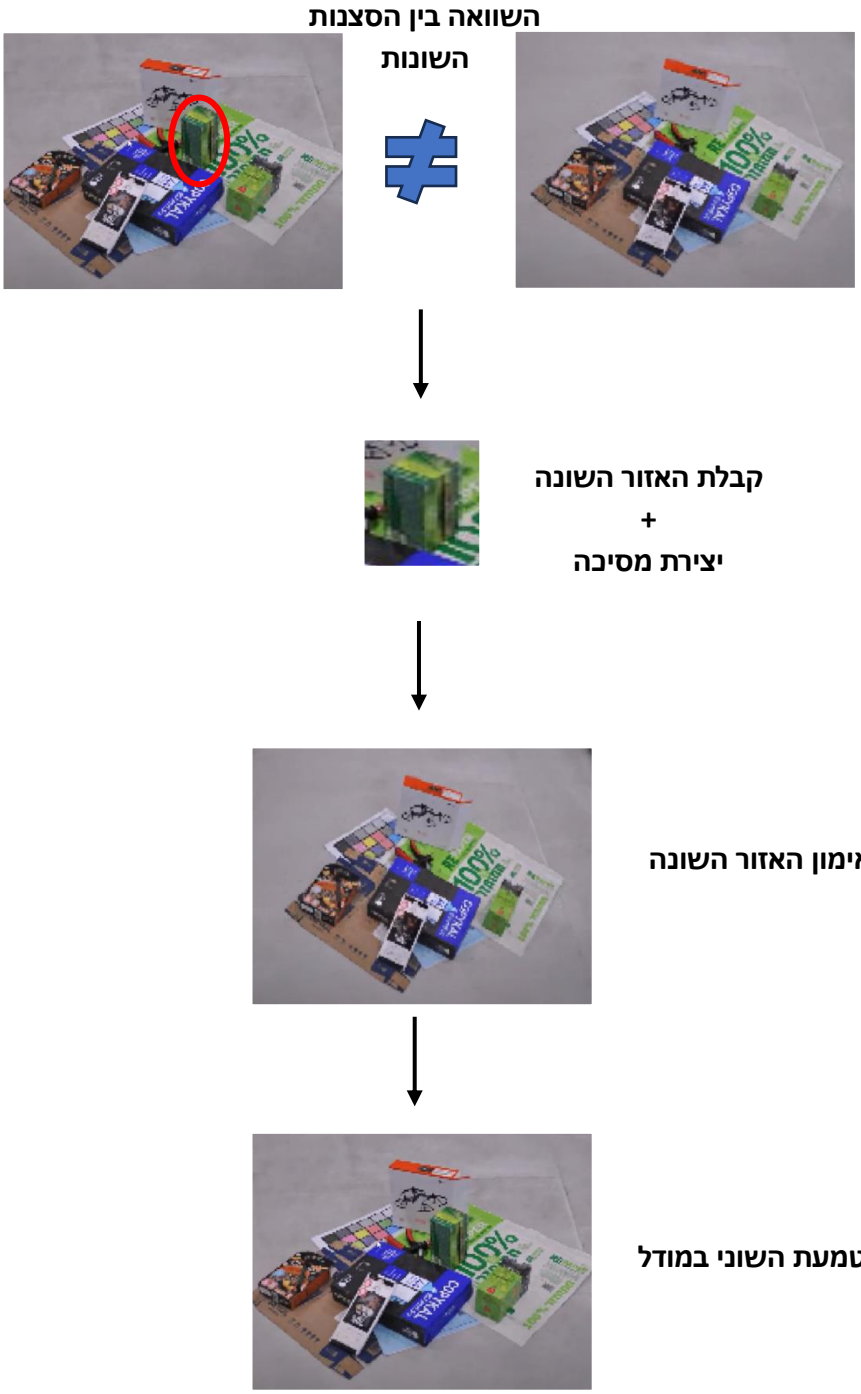


שלב השני: לאמן את המודל רק באזור שסווג כשונה.

לאחר שיצרנו לכל תמונה בסט עם השינוי מסיכה שמתארת את הפיקסלים שהשתנו ,ניקח את המסכות האלה ונכניס אותם לקובץ ה-transform שמועבר לפני האימון ומתאר את מיקומי המצלמות עבור התמונות.

כעת כאשר נמשיך את האימון הפיקסלים שבחרנו מכל תמונה יהיו רק מקבוצת הפיקסלים שבמסכה יהיו "1" כלומר הפיקסלים שהשתנו בין הסצנות ,כלומר נגביל את עולם האימון שלנו רק לאזור שהשתנה בעבור כל תמונה ונקבל את המודל שאנחנו רוצים- כלומר הפיקסלים שהיו באזור השונה ילמדו מחדש וישתנו בהתאם.

תהליך האימון כפי שאנחנו מתכננים:



פירוט הפתרון שלנו (מימוש):

פסואדו קוד עבור רעיון המימוש:

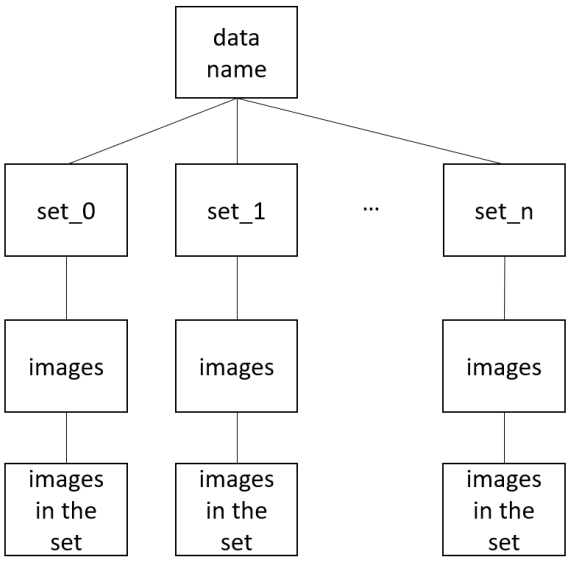
```
f (trained_model, images, camera_intrinsics):

    rendered_images = trained_model(camera_intrinsics)

    masks = thresh(loss(rendered_images, images))

    new_model = train(trained_model, images, masks)

    return new_model
```

אנו משתמשים בקוד פתוח של ספריית GitHub בשם nerfstudio אשר נותנת ממשק יחסית נוח להתעסק עם NeRF-ים שונים ולהוסיף מתודות חדשות.

בחרנו להתעסק עם NeRF הנקרא Nerfacto שהוא מודל המכיל בתוכו רעיונות מסוגי nerf-ים שונים כאשר הוא מביא איזון בין ביצועים טובים לקוד יחסית נוח לעבודה.

יצרנו קוד שממש את הפסואדו קוד שהצגנו, בעזרת docker של nerfstudio. הקוד שלנו משתמש בdocker בכדי לאמן את המודל, ואנחנו כתבנו את כל המסגרת בכדי שהמודל יתאמן כמו שאנחנו רוצים. בנוסף ביקשנו מהמשתמש שיוסיף פרמטרים נחוצים בrun configuration: מספר סטים שהוא יתאמן, האם לדלג על האימון הראשון אם כבר התאמנו עליו, והיכן נמצא הדאטה.

הדאטה ההתחלתי יראה כך בתיקיות:



בעיה שנתקלנו במהלך מימוש הפתרון ולא לגמרי הצלחנו לפתור בסוף הייתה סטייה של ה-colmap בעיה זו נותרב פתוחה עבורנו והצלחנו לפתור אותה, בעצם כאשר הפעלנו את ה-colmap על הדאטה שלנו שהיה מורכב מחיבור של הסט המקורי והסט עם השינוי נוצרו מיקומים לא מדויקים עבור התמונות ובכל התמונות שייצרנו מהמודל בעזרת המיקומים היו עם סטייה דבר שהקשה על יצירת המסכה והשוואת התמונות.

התקנת הפרויקט והרצה:

1. להתקין WSL2 עם Ubuntu 22.04.
2. להתקין docker desktop.
3. בכדי לדבג:
   a. לערוך את הקובץ env. כך:
      לערוך את CUDA_ARCHITECTURES ליכולת המתאימה של הכרטיס בגרפי.
   b. לערוך את הקובץ docker-compose.yaml כך:
      הvolumes יהיו תואמים לנתיב המוחלט, כך שאנחנו ממפים את הפרויקט שלנו מהמחשב לcontainer.
   c. להתקין listener על הdocker socket.
   d. ליצור remote interpreter שמבוסס על docker-compose.yaml.

את הקוד אפשר להריץ מהrun configuration של הIDE שמשתמשים (השתמשנו בpycharm), וניתן לערוך פרמטרים שינתנו לקוד בrun configuration עבור דברים נחוצים לקוד.

מה למדנו ומה ידענו?

הגענו לפרויקט בלי היכרות עם הנושא או עם נושאים דומים והשקענו הרבה זמן ללמוד על הנושא על ידי קריאת מאמרים, במהלך העבודה גילינו כי לימוד תאורטי אינו מספיק ונדרשנו ללמוד הרבה דברים כגון איך להתקין ,להפעיל ולעבוד עם הרבה כלים שונים.

במבט לאחור אנחנו יכולים להגיד שלמדנו המון מהעבודה: הרבה דברים טכניים, איך לעבוד בצוות ואיך נראה מהלך עבודה על פרויקט שמלא באי ודעות ושינויי כיוון.

המשך הפרויקט בעתיד:

בסופו של דבר לא הצלחנו לפתור את הבעיה באופן שרצינו, אנו חושבים שהמשך עבודה על הבעיה בהחלט אפשרי ויהיה ניתן "לתקוף" אותה מעוד כיוונים לדוגמה יצירת אזור תלת מימדי במרחב שמסווג את האזור שהשתנה ולא עם מסיכות עבור כל פריים כמו שאנחנו ניסינו.
בנוסף כיוון מעניין לעבודה לאחר שפותרים את הבעיה הוא שילוב הפתרון במערכות Realtime של NeRF.

צוואר הבקבוק שלנו בפרויקט

ניתן לחלק את העבודה שלנו במהלך השנה על הפרויקט ל2 חלקים מרכזיים:
חלק ראשון שאפשר לקרוא לו חלק יבש, במחצית הראשונה קראנו מאמרים על סוגי NeRF-ים שונים ולמדנו את הנושא שהיה חדש לנו דבר שלקח לנו יותר זמן מהתכנון המקורי.
וחלק שני שבו התחלנו להתנסות ולעבוד על NeRF-ים קיימים, בשלב הזה החלטנו שאנו רוצים לעבוד עם קוד הנקרא – instant ngp לאחר תקופה יחסית ארוכה שהבנו איך להתקין ולפעיל את הקוד, התחלנו לחשוב על פתרונות והבנו שהקוד שאותו בחרנו לא מתאים לדרישות שלנו ויהיה מאוד קשה לעבוד איתו, בעקבות זאת החלטנו לחפש קוד יותר נוח שיענה על הדרישות שלנו, מצאנו כמה קודים שונים שהיו נראים לנו מתאימים אולם במהלך ההתקנה והעבודה איתם נתקלנו בבעיות רבות מה שעיכב אותנו מאוד ועלה לנו בהרבה זמן, לבסוף החלטנו לעבוד על קוד פתוח הנקרא nerf-studio שהושק רק בתחילת 2023 כלומר לא היה קיים בתחילת הפרויקט שלנו.
העבודה עם הקוד החדש בהחלט לא הייתה קלה בהתחלה אבל הרגשנו שלמדנו מהפעם הקודמת והצלחנו להתקדם ולהתמודד עם הקשיים שהיו במעבר ועם הקוד החדש.

תוצאה

כפי שהסברנו בחלק המקדים בדוח נתקלנו בבעיה שלא הצלחנו לפתור במיקומי המצלמות ההתקבלו מה-colmap אשר היה בהם סטייה קטנה שגרמה לבעיה ביצירת המסכות וכתוצאה מכך רצינו לבדוק את הפתרון שלנו עם שימוש במסכות שנוצרו באופן ידני, לצערנו **התוצאות** שקיבלנו לא היו טובות אך והיה תחושה שהכיוון שבו בחרנו ללכת לא יעבוד ולא היה לנו זמן לנסות כיוון אחר.