

הערכת עומק רחפן מתמונה בודדת - סקירה טכנית מקייפה

1. הגדרת המשימה - זיהוי עומק רחפן בתמונה מונוקולרית

תיאור כללי: המשימה היא לבצע לוקליזציה תלת-ממדית של רחפן באמצעות מצלמה יחידה, בדגש על קביעת מרחקו של הרחפן מהמצולמה (הציר Z). במקרים אחרים, על סמך תמונה בודדת שבה מופיע הרחפן, יש להזות היכן הרחפן נמצא במרחב unlike stereo or multi-camera systems, a single image doesn't have depth information. ¹ provide direct geometric triangulation, so depth must be inferred from visual cues . לשם כך משלבים בדרך כלל שתי משימות: (1) **זיהוי אובייקט** – למשל שימוש ברשת כמו YOLO לאיתור ותיבת תיחום של הרחפן בתמונה, ו-(2) **הערכת עומק מונוקולרית** – שימוש ברשת נפרדת להיזוי מרחק או מפת עומק מitor תמונה בודדת. בחיבור של שני אלה ניתן לקבל את מיקום הרחפן בתלת-ממד. הפרויקט המדובר עובד על נתוני סינטטיים בלבד (מטור Unreal Engine), אך שרשונו **אמת-מידה מושלמת** למיקום הרחפן בכל תמונה, מה שמאפשר אימון ובדיקה יעילים ללא מדידות בעולם האמיתי . התוצאה הסופית המצופה היא מערכת שמקבלת תמונה (ממסדת וידאו או סטילס), מזהה את הרחפן בתוכה, ומעריכה את מרחקו מהמצולמה (כלומר את קואורדינטת ה-Z שלו ביחס למצולמה). בשלב מתќדם יותר, ניתן גם להסיק את המיקום המלא (Z,Y,X) ולבצע עיקוב בזמן אמת. בפועל, הפיפליין המלא יכול **Generation of synthetic data → Object Tracking (קלמן)** ² . בפרויקט שלם, ההתקדמות הראשונית היא בחלק העומק (Z) בלבד, וכך בשלב זה היעד הוא לקבל עבור כל תמונה את מרחק הרחפן. חשוב להבין שהזען אתגר לא טריוני-אי – בהיעדר מידע סטריאוסקופי, המערכת צריכה להסיק עומק מرمיצים ויזואליים (גודל מוגען של אובייקטים, פרספקטיבה, טשטוש אטמוספורי וכו') בדומה לראיות עומק בעין אחת ¹ .

2. תחומי ידע וטכנולוגיות רלוונטיות

הפרויקט משלב מספר תחומי למידה عمוקה וראייה ממוחשבת. חלק מהנושאים יתacen שאתם כבר מכירים, ואחרים דורשים העמקה נוספת. להלן המרכזים שבהם, יחד עם הסבר קצר:

רשתות להערכת עומק מונוקולרי: מדובר ברשתות נירוניות החוזרות מרחק מכל פיקסל בתמונה בודדת. רשת עומק מונוקולרית מסיקה עומק ייחסי – כלומר, היא יודעת אילו חלקים בתמונה קרובים או רחוקים יותר זה מזה, אך לא קנה מידת מוחלט לא ציול נוספים ⁴ . יש להבין את בעיית **עミימות הסקאלה (Scale Ambiguity)** ⁵ : סצנות שונות יכולות להיות למרות מרחקים מוחלטים שונים, ולכן רשת לא מידע נוספת לא יודעת אם הרחפן נמצא במרחק 5 מ' או 50 מ' – היא רק יודעת למשל שאובייקט א' רחוק פי 2 מאובייקט ב' . ישנו מודלים מודרניים מבוססי Transformer שהוכשרו על מיליון תמונות כדי ללמידה רתמי עומק (כגון **ZoeDepth**, **MiDaS**, **Depth Anything** – ראו פירוט בהמשך) ⁵ . אם תחום זה חדש לכם, תצרכו ללמידה על ארכיטקטורות נפוצות כדוגמים כליה.

זיהוי אובייקטים (Object Detection) – YOLO: כדי להתפס את חישוב העומק לרחפן בלבד, יש להזות אותו בתמונה. סביר שתשתמשו בIMPLEMENTATION של מודלי YOLO (You Only Look Once) ליזיהו מורה של הרחפן ומנתן תיבת תיחום סביבו. יתכן וכבר עבדתם עם YOLO או רשת דומה; אם לא, כדאי להכיר מושגים כמו IOU, Bounding Box, והאופן שבו רשתות כאלה מאומנות. בפרויקט הנוכחי, YOLO (למשל גרסה 8/YOLOv8/YOLOv5/YOLOv7) תשמש כמודול ליזיהו מיקום הרחפן בפיקסלים ³ . היזיהו מבוסס תמונה בודדת, ומספק את מרכז הרחפן ותיבת התיחום, המשמשים בהמשך לחישוב המיקום המרחבוי.

גיאומטרית מצלמה והמרת קואורדינטות: נושא חשוב במיוחד הוא להבין כיצד לערוב מתחמונת דו-ממדית - קואורדינטות תלת-ממד. לשם כך דרושה היכרות עם **פרמטרים פנימיים של המצלמה** (Camera Intrinsics) – אורך מוקד (בפיקסלים), נקודת מרכז, ועוד. המטרה של פיקסל למחרך דורשת את הפרמטרים הללו ואת עומק ה-Z המשוער. למשל, ברגע שידועים את Z (מרחק מהמצלמה) ואת מיקום הפיקסל של הרחפן $(x_{\text{pixel}}, y_{\text{pixel}})$, ניתן לחשב את (x, y) המשוחזר $\text{using } x = \frac{x_{\text{pixel}}}{Z} \cdot f_c \text{ and } y = \frac{y_{\text{pixel}}}{Z} \cdot f_c$. לכן, יש לוודא שמקיראים את מודל המצלמה (מודל Pinhole) ולוודא שפרמטרי המצלמה הוירטואלית ב-*Unreal* תואמים לחישובים (עד על כרך חלק 5). היכרות עם המושגים הללו חיונית כדי להפוך עומק מחושב (Z) לקואורדינטה מטרית בעלת משמעות. כמו כן, תצטרכו להבין את ההבדל בין מערכת קואורדינטות **עלומן** למערכת **מצלמה** – המיקום האמתי של הרחפן צריך להיות מיצג במערכת הצירים של המצלמה לצורך לימוד ובדיקה. אם הקונספטים האלה חדשים, מומלץ לערען את הידע באופטיקה גיאומטרית בסיסית.

מעקב וחולנית קלמן (Kalman Filter): מאחר שמדובר ברחפן שעשו לנו,סביר שהמערכת שלכם תידרש לעקב אחריו על פי רצף פרויימרים. מסנן קלמן הוא כל-קלاسي למיזוג הערכות רועשות לאור זמן ולקבלת מסלול חלק. גם אם בשלב ראשון תעבדו על תമונות בוודאות, בשלב המשך (או עברו וידאו) כדאי להכיר את השימוש ב-*Kalman* כדי **החלקת נתוני המיקום והעומק**. בפרויקט הדוגמה, אחרי שחישבו בכל פריטים את המיקום, החילו פילטר קלמן כדי לסנן רעשם (:right) של תיבת הזרוי, או-יציבות בחיזוי העומק) וגם כדי לחזות את מיקום הרחפן בפריטים הבאים במקורה שהזיהוי מתפספס. אם טרם עבדתם עם *Kalman*, מומלץ ללמוד את מודל הממצבים, מטריצות התהיליך והמדידה, וכייזד להזין אלגוריתם קלמן בפייתון (יש לך ספריות מוכנות – ראו סעיף כלים).

עבודה עם נתונים סינטטי (Unreal Engine): עבודה עם נתונים מסונחים מצבה הזדמנויות ואתגרים. **היתרונות:** אפשר לקבל אמת-מידה מושלמת לזרוי ומרחך (*Unreal* יכול לספק את מיקום האובייקט המדויק בכל פריטים), וכן לשולוט ברגעון תנאי הסצנה באופן שיטתי. בפרויקט לדוגמה, ניצלו זאת לצירות וריאציות רבות: צג אוור שונה, תנאי תאורה (בוקר/ערב), רקעים מגוונים, ומרחקי ורחפן שונים כדי שהמודול יוכל לטפל בכל הממצבים. **האתגרים :** יש לשים לב לפער האפער בין עולם סינטטי לעולם האמתי (Gap) – טקסטורות וגרפיקה ממוחשבת עלולות להיות "נקיות" מדי, אך שאמ מתקנים בהמשך להפעיל את המודול על וידאו אמתי, יתכן ויידרש Fine-tuning או אוגננטציה חזקה כדי שהמודול יתמודד עם רעשים אמיתיים. בנוסף, יש לוודא שפומט הנתונים התפעולי (למשל, קבוע תמונה + קבוע NOSE של קואורדינטות) מוגדר היטב כדי שתהיליך האימון יהיה חלק. כאמור חשובה נוספת היא סברון נתונים – מכיוון שב-*Unreal* ניתן להפיק גם וידאו וגם לוג של מיקום הרחפן, חיבים לוודא שידיעים להתאים כל פריטים לתוויות (זמן אמת) הנכונה. **בפרט, המיקום שישמש כתווית לעומק חייב להיות באוטה מערכת צירים של המצלמה** – כמובן, אם *Unreal* נותן מיקום בעולם, יש לתרגם אותו לקואורדינטות יחסיות לצלמה (עד פריט בסעיף 5). אם נושא זה חדש, מומלץ ללמידה על **תהליך Domain Randomization** בסימולציות (שינוי פרמטרים אקרים כדי שהמודול לא יתאים עצמו לפרטים ספציפיים), ועל שיטות יצוא נתונים מ-*Unreal* (למשל שימוש ב-CV-Albedo, או כתיבת סקריפט באנג'ין שմצלם וושומר זוגות של תמונה+נתוני מיקום).

אוגננטציה נתונים (Data Augmentation): זה נושא מוכר בראיית מכונה, ובביר שכך השתמשתם בו, אך שווה להציג אותו בהקשר שלנו. מלבד הגיון הסינטטי שנותן להשיג מותק המונע (אכמור – תנאי תאורה, זווית טישה, מרחקים וכו'), אפשר לבצע אוגננטציה קלאסית על התמונות הסינטטיות בעת האימון. למשל: הוספה רעש, טשטוש קל, שינוי בהירות/ניגודיות, היפוך תמונה וכדומה. מטרת האוגננטציה היא לשפר את יכולת הכללה (generalization) של הרשת כך שלא תלמד פרטיים ספציפיים מדי בסצנות הסינטטיות. במיוחד אם תכוננתם בסוף לבדוק את האלגוריתם על וידאו אמתי, אוגננטציה תסייע לנשר על הפער. יש כלים נוחים בפייתון לביצוע אוגננטציות כמו `Albumentations`, או המודול `torchvision.transforms` (או `MSE` – למשל – למדוד הטעינה של הדטה).

בנוסף לנושאים העיקריים הללו, השתמשו ודי בידע כללי בלמידה عمוקה (אימון מודלים, פונקציות הפס – *Transfer Learning*) – למשל, שימוש במודל עומק עבור רגרסית מרחק, אופטימיזציה, וכו'). יתכן ותישמו **למידת מעבר** (Transfer Learning) – למשל, שימוש במודל עומק טרום-מאומן וההתאמתו לדאטתת שלכם, דבר שדורש הבנה איך לטעון מודל מאומן ולהמשיך לאמן אותו על דאטתת חדש.

לסייעם, הרשימה לעיל מרכזת את עיקר התחומיים שבהם כדאי לודא שיש לכם ידע מספק, ולזהות אילו דרישים למידה נוספת לפני או במהלך לביצוע הפרויקט.

3. כלים, ספריות וסביבת פיתוח מומלצות

פרויקט מסווג זה מחייב שימוש בערכת כלים מתאימה לתחזוקת קוד, אימון רשותות ועיבוד נתונים. להלן הכלים המרכזיים שמצווקו להם (רביהם מהם יתכן וכבר בידיכם):

שפת תכנות וסביבה: כמעט כל המחבר והפיתוח ייעשו ב-[Python](#), שהוא השפה הנפוצה ביותר ללמידה עמוקה וראייה ממוחשבת. מומלץ לבודד בסביבת פיתוח שתומכת בהרצאת קוד אינטראקטיבית, כגון Jupyter Notebook/Lab (לנייסויים מהירים) או סביבות IDE כמו PyCharm ו-[VS Code](#). וDAO שיש לכמ גישה לחומרה גרפית – [GPU](#) – לצורך אימון והרצאת המודלים. מודלי עומק מודרניים כבדים יחסית; למשל, במימוש המלא דוחה על קצב ~25-30FPS על GPU צרכנו (בדומה ל-[RTX 3060](#)) ¹³. אם אין בידכם GPU מקומי חזק, ניתן להשתמש בשירותים כמו Google Colab או Kaggle Notebooks בענן לניסויים.

ספריות ללמידה عمוקה (Deep Learning): הפרויקט צפוי להתבצע ב-[PyTorch](#) (בהתחשב שרוב דוגמאות הקוד והמודלים המוכנים נמצאים שם). PyTorch יספק את התשתיות לבניית רשותות, טעינת נתונים (באמצעות [PyTorch.utils.data.Dataset](#)), אימון לולאת למידה וכו'. מידת הצורך נתן לשקו גם שימוש ב-[Lightning](#) כדי לפשט קוד אימון, אך זה לא חובה. ספריית [TorchVision](#) עם מודלים חזקים ([ResNet](#), [TensorFlow/Keras](#) וכו' – אם תרצו להשתמש ב-Transfer Learning) ועם טרנספורמציות לתמונות. אם מישחו בצוות מגע מרתק של בוגרות בפייתון/[PyTorch](#), ניתן תאזרית למשם, אך מרבית הכלים הקיימים לנו שא (מודלי [YOLO](#), מודלי עומק) זמינים.

כלי זיהוי אובייקטים (YOLO): בשלב זיהוי הרחפן בתמונה, תוכלו להשתמש בIMPLEMENTATIONS מוכנים של [YOLO](#). ספריית [ultralytics](#) (ניתנת להתקנה דרך pip) מספקת ממשק נוח לדגמי [YOLOv5](#)-[YOLOv8](#) בפייתון, כולל אפשרות אימון על נתונים שלכם. תצטרכו לאסוף את התמונות הסינטטיות ולסמן להן תיבות תיחום של הרחפן נתוני [YOLO](#) (ניתן להוציא מ-[Unreal Engine](#) באופן אוטומטי את תיבות התיחום על בסיס מודל התלתל-מדד). לאחר אימון מודל זהה, תוכלו להשתמש בו כדי לקבל עבור כל תמונה את מיקום הרחפן בפיקסלים (נניח $c(x_center, y_center)$ או תיבת תיחום בצורה $[x_min, y_min, x_max, y_max]$). יש גם גרסאות קוד פתוח אחרות כמו ([darknet](#), [C++](#)), אך בפייתון התהילה יהיה פשוט יותר.

מודלים להערכת עומק: כתואר בסעיף 4, מומלץ להתחיל שימוש במודל עומק מוכן. לדוגמה, [MiDaS](#) של Intel או [ZoeDepth](#) של LSIS-InTEL. את v2.1/v3. At PyTorch Hub [MiDaS](#) אפשר לטעון שירות דרך [load](#) ללא צורך לשכפל ריפו: לדוגמה, הטענת המודל הגדול מתבצעת כך: `midas = torch.hub.load("intel-is1/MiDaS", "midas_transforms")`. גם את הטרנספורמציות החדשנות (שינוי גודל וונרטול) אפשר לטעון: `midas("MiDaS")`. באופן דומה, קיימים ריפזיטורי עבור [torch.hub.load\("intel-is1/MiDaS", "transforms"\)](#) ¹⁴. באופן דומה, אין הכרח לבצע אימון מחדש של מודל העומק – אפשר להשתמש במודל מאונן ולישם **CTION SKAALA** כדי לקבל עומק מוחלט (ראו סעיף 5.3 במאמר המצורף) ¹⁵. במידה ותרצו **לאמן מודל عمוק בעצמכם** על הדטה הסינטטי, תזדקקו לספריות הנ"ל בכל מקרה, ובנוסח אויל לכלי מעקב ניסויים ([TensorBoard](#) למשל) כדי לצפות בתוצאות.

ניהול וקדם-עיבוד של נתונים: ספריות כמו [OpenCV](#) ו-[PIL](#) Pillow שימושיות לטיעינת תמונות, המרת פורמטים, ועיבוד בסיסי (חיתוך תמונה לפי bounding box, שינוי גודל וכו'). OpenCV גם מספק פונקציות לצור תיבות תיחום על תמונה – שימושי להדמיה ולבדיקה. אם יש לכם פורטט ויצוא מ-[Unreal Engine](#) (למשל קובצי JSON עם מיקום רחפן לכל פריים), תוכלו להשתמש בספרייה [json](#) הפיתונית או Pandas לקריאת קבצים אלה, ולהיבור בין הרשומות לתמונות.

בנויות מערך הנתונים לאימון תעשה נראית באופן ייעודי: אפשר לכתוב Dataset שפיטו קורא לכל אינדקס את התמונה ואת הערך \$Z\$ המתאים מה-JSON. ועודו שסידור הקבצים (תיקיות, שמות) מובנה והגוני כדי למונע שגיאות. כמו כן, אם אתם מפוקים **DEPTH AMBIGUITY** מ-UrealHub (כל פיקסל), קבצים אלו יכולים להיות בפורמט תמונות עומק או EXR – צריך לוודא שאופן הטענה שלהם נכון (למשל OpenCV יודע לקרוא TIFF/PNG של 16-ビט, NumPy יכולת לקרוא EXR דרך skimage וכו').

ספריות לאוגמנטציה: כאמור, Albumentations היא ספרייה חזקה ופешטה ליישום אוגמנטציות על התמונות בזמן הטעינה. היא משתמשת יפה עם PyTorch (מחזירה תמונות pytorch שאפשר להפוך לטנזור). גם transformstorchvision可以在הספיק לדבירם בסיסיים (היפוכים, חיתוך אקראי, שינוי בהירות/ניגוד). אוגמנטציות אפשר להגדיר בתוך מחלקת Dataset או בשלב יצירתה .batch.

כליים למסנן קלמן ולמעקב: את שלב הקלמן אפשר למשמש "מאפס" עם NumPy, אך קל יותר להשתמש בספרייה כמו **pykalman** או **FilterPy** בפייטון, המספקות שימושי EKF ו-Kalman Filter. מסנן קלמן דורש להגדיר וקבעו מצב, מטריצות תהודה ורעש וכו' – בפרויקט הדוגמה הגדרו את המצב כ-6 מימדים [z, y, x, z, y, x] ושילבו את חישובי Z, Y, X, שציינו קודם ⁶. ספריות אלה יחסכו לכם קצת זמן בניהול האלגברה. לחłówין, יש שימוש Kalman ב-OpenCV (אם עובדים ב-C++), אבל נניח שתעבדו בפייטון.

סביבת Unreal Engine: למרות זה לא בדיקת "ספרייה" בפייטון, כלי הליבה להפקת הדאטה הוא מנוע Unreal Engine 5 (או 4). ועודו שיש לכם גישה לפרויקט Unreal שבו יש סצנה עם רחפן ומצלמה וירוטואלית. כדאי להשתמש בתוסף **Python 7-Real** המאפשר להריץ סקרופטים שמצלמים תמונות ושמורם מידע. להלן, ניתן להריץ את הסימולציה ולשמר וידאו + קובץ לוג (כפי שתואר במאמר). יתכן ושימוש בחבילת **AirSim** של Microsoft (סימולטור רחפנים מבוסס Unreal) יוכל לספק קישור דורך, שכן הוא מוכן לצלם תמונות מצלמה על רחפן ולתת גם מידע מיקום. כך או אחרת, Unreal הוא חלק בלתי נפרד מניהול הדאטה הסינטטי. בנוסף, תזדקקו לכלי עיברת נתונים – למשל, אם תגלו פיקסליהם חריגים במפות העומק או צורך להתאים את פורמט הקואורדינטות, ניתן שתכתבו סקריפט לתקן/סינון הדאטה.

סיכום כל הפיתוח: השימוש של Python עם PyTorch יתנו מנגנון לרוב צרכי המימוש (אימון מודלים, הרצת רשותות מאומנות בזמן אמת, וכו'). Unreal Engine משמש לייצרת הדאטה. ספריות עזר כמו OpenCV, Albumentations, FilterPy יתאפשרו לעבוד נתונים ומעקב. הקפידו לעבד בסביבה שיש בה מספק זיכרון GPU (מודולי עומק גדולים צורכים ~2GB ומעלה לשרת אחד, 1-YOLO קטן עוד כמה מגה), ובוצעו בדיקות לכל רכיב בנפרד לפני שמחברים אותם לצינור שלם.

4. ארכיטקטורת רשת עומק מומלצת להתחלה

ישנם מודלים רבים להערכת עומק מבט יחיד, אך נרצה לבחור בפתרון **פישוט יחסית ללמידה ולשימוש** – זהה שהויה מספק טוב להוכחת ההיכנות הראשונית. להלן מספר אפשרויות מוביילית, והמלצותנו:

MiDaS (Mixed Depth Estimation): זה מודול חד-עוני שפותח ע"י אונטאל, ונחשב baseline חזק למשימות עומק מונוקולרי ¹⁷. יתրונו הוא שיש דגמים מאומנים זמינים ("off-the-shelf") שניתן פשטוט להוריד ולהריץ, ללא צורך באימון יקר. MiDaS שולב במערכת נתונים רבים כדי להיות רובעטי (מיועד להערכת עומק יחסית כללי). הגרסאות החדשות מבוססות על Transformer (DPT) בתצורת Encoder-Decoder ¹⁸ – ארכיטקטורה מודרנית שנותנת תוצאות מדויקות בהרבה מרשתות CNN מסורתיות. היתרון הגדול: **כל מאוד להשתמש** במודול זהה – טעינה דרΗ PyTorch, וקיבלת מפות עומק איקוניות ישירות מתוך מותנים קטל ¹⁹ ¹⁴. החיסרון: הוא נוטן עומק כדי להמייר עומק יחסית שרירותית. בפרויקט שלכם, זה לא בהכרח עבה – כפי שנפרט בסעיף 5, ניתן לבצע **fine-tune** כדי להמייר עומק יחסית למטרים ¹⁶. MiDaS יהיה בחירה מצוינת להתחלה: תלמדו איך להשתמש בו, אולי תבצעו Fine-tune כל על הנתונים הסינטטיים שלכם (אם תרצו לשפר ביצועים), ותקבלו baseline של עומק. אולי בפועל לא אימון נוסף,

- 5 MiDaS מספקת עומק יחסית המבוסס על רמזי פרספקטיביה, גודל, טקסטורה וכו' שהוא מAMILIONI תמונות מה שיכול להספק כדי לספק סדר גודל נכון של מרחק לרוחפן.

ZoeDepth: מודל חדש (2023) מבית intel-iS (2023) שנועד להתמודד עם בעיית הסקירה ע"י שילוב בין למידת עומק יחסית ועומק מטרתי. **ZoeDepth** משתמש באדריכלות של MiDaS בסיסי, אך מוסיף לה ראש הנקרוא *Metric Bins Module* המאפשר למודל להפיק עומק בסקירה אבסולוטית (מטרים) לאחר שעבר **coil על דאטה מטרי**²⁰. האימון של Zoe כל שלב פרה-טראינינג על דאטא יחסית (כמו MiDaS) ואז Fine-tuning על ערכי עומק מוחלט (ממאגרים כמו NYU, KITTI). בפועל, ZoeDepth מסוגל לתת הערכת מרחק ישירה במטרים ללאcoil נוסף, בתנאי שהנתונים דומים לדאטא שבו הוא אומן. השימוש ב-Zoe יכול לחסוך לכם את שלבcoil הסקירה, אך יש לשים לב: אם תנאי הסצנה שלכם (ኖפים או טווחי מרחק) שונים מאוד מהדאטא שעליו Zoe אומן, עדין יתכן צורך לכיד או לאמן אותו מחדש על הדאטא המקורי. ZoeDepth זמין בקוד פתוח (ראו GitHub לעיל) ודרכו תוספת הראש וכונראה דגם backbone גדול, אבל קר שהערכתה שלו די נוחה. הוא מעת **כבד יותר** MiDaS (עקב תוספת הראש וכונראה דגם backbone גדול), אבל עדין רץ בזמן (עד ~10-15 FPS על GPU בינווני לפי דיווחים). למידה שלו מעט מורכבת יותר ממודל סטנדרטי כי צריך להבין את עניין *the-discriminative*, אך תיעוד המודל והמאמר קיימים. Zoe מומלץ אם אתם שואפים לדוק גובה יותר בעומקים המוחלטים כבר מהתחלה, או אם תרצו לננות גישה עדכנית לשימושה תוצאות state-of-the-art. אפשר להתחילה עם MiDaS ואחר קר לעבורו *ZoeDepth* כדי לסייע. שימושם של Zoe כבר לא מפottaח אקטיבית ע"י אונטול (לפי GitHub), אך זה לא קוריטי לשימושם.

Depth / Depth Anything / Depth Everything: מודל עומק בסגנון "מודל יסוד" (Foundation Model) שפורסם לאחרונה, המכומן על כוכות דאטא אדריה – לפי ההארה, **1.5 מיליון תמונות עם תווית עומק ועוד 62 מיליון תמונות ללא תווית**²¹. הרעיון שהוא יכול לשמש כמודל עומק כללי וחזק מאוד כמעט לכל סצנה. היתרון: אמור לו לספק אינטואיטיביות מרובות בהערכת עומק, אולי טובות אף יותר מ-MiDaS/Zoe. החיסרון: מודל כבידת (בהתחם לכמות הנתונים), פחות נפוץ בשימוש בקהליה הנוכחי להיום, וDOI רץ לאט יותר. כמו כן, יתכן ש-"*Depth Anything*" מוגדר כמיון (depth-anything github), אך להערכתה ראשונית יתכן זה overkill.

המלצה מעשית: התחלו עם **MiDaS** או **ZoeDepth**. MiDaS יספק לכם את הדרך הפשטת ביוטר לקבל תוצאות ולעבור את כל השרשרת הטכנית (טעינת תמונה → הרצת רשת → הפקת עומק → רוחפן). ZoeDepth יכול להיות הצעד הבא אם זוקרים לשיפור בדוק המטרתי. שניהם יכולים לשילוב בקוד (כמה שורות טעינה והערכת CPI שתואר)¹⁹. בהמשך, לאחר שתתקבלו תוצאות בסיסיות, תוכלו לבחון אם כדאי לאמן מודל מותאם אישית. למשל, אפשר **לאמן רשת קלה עצמאם** שתעשה גרסיה של מרחק הרוחפן מתמונה: לקחת מודל כגון ResNet-18 TorchVision מוקן (TorchVision) ולשנות את השכבה האחורונה שלו להוציא נוירון יחיד, ולאמן על תמונות (או קורופים של הרוחפן) עם תווית \$Z\$ אמרתית. זה אמור "פושט" אך דורש כמה דיאטה רבה כדי להגיע לדוק גובה, בעוד שמודלים כמו MiDaS כבר למדו מיליאוני דוגמאות של רמזי עומק⁵. لكن, הגישה היברידית (זיהוי + מודל עומק אומן + coil) היא קיצור דרך יעיל. לסיום, כמודל עומק ראשון – S MiDaS (v3 עם DPT-large) מומלץ כפתרון שקל ללמידה ולהתחילה להפיק תוצאות.

5. דגשים מיוחדים בעבודה עם דאטא סינטטי מ-Unreal Engine

כעת נתרץ בהיבטים ייחודיים לפרויקט CPI שהוגדר – שימוש בעיבוד Unreal Engine להפקת הדאטא, ודרישות הנובעות מכך לצורך הערך עומק מוצלח:

ישור מערכות קוואורדינטות ואמת המידה: אחד היתרונות של Unreal Engine הוא יכולת לקבל קוואורדינטות אמת של האובייקטים בסצנה². ואכן, בקובצי הלוג-JSON שלכם כנראה יש לכל פריטים את מיקום הרוחפן. **חשוב לוודא שהמיקום הזה מומר למערכת המצלמה.** במקרים אחרים, אם Unreal מספק מיקום ב-*World Space* (נעיה קוואורדינטות גלובליות במרחב המשחץ), יש לבצע טרנספורמציה למערכת הצירים של המצלמה (Camera Space) כך שה-Z הוא למעשה המרחק מהמצלמה. במאמר המצורף מודגש: "המיקום חייב להיות במרחב המצלמה, לא במרחב העולם"²³. הדרך לעשות זאת היא: לקחת את וקטור המיקום בעולם של הרוחפן, גורוע ממנו את וקטור המיקום

בعالם של המצלמה, אז להחיל טרנספורמצית סיבוב הפוכה של המצלמה²⁴. פعلاה זו תניב וקטור \$(Z, Y, X)\$ במרחב המצלמה, כאשר \$Z\$ הוא העומק הisher קדימה. dazu שתהיליך זה מושם נconaו שאתם מכנים את תווות ה-Z לאימון/בדיקה – לאחרת תכinoisו שניאת כויל משמעותית. הערה נוספת: כדי לבצע את רישום הנתונים בצדרא timestamps סינכרונית ומדויקת – לדוגמה, לרשותם עבור כל פריים את מספר הפריים והמיוקם, ולא להסתמך על timestamps שיכולים לגרום להסתה¹¹.

התאמת פרמטרי המצלמה הוירטואלית: כדי שהערכת העומק והמיוקם תהיה נכונה, **המצלמה בסימולטור חייבת להתנהג כמו המצלמה האמיתית המשוערת**. משמעות הדבר היא להגדיר נכון את הפרמטרים הפנימיים: אורך מוקד, גודל חישון, רזולוציה, וכו'. כאמור מפורט כיצד הגדרו מצלמה-iReal: לדוגמה, בחירת אורך מוקד (35-45 מ''), גודל חישון ידוע, ורזולוציה 1920x1080²⁵. מtower אלה מחשבים את זווית הראייה (FOV) ומפיקים את מטריצת המצלמה \$A\$²⁶. בעת חישוב מוקום מ-3D-X של הרחפן מתוך העומק, אתם חיברים להשתמש באוטם \$Ax_c, f_c, x_c, y_c\$ של המצלמה הוירטואלית²⁷. אם יהיה פער, תקבלו שנית קנה-מידה (למשל, העומק יכול להיות נכון אבל ה-3D-X יצא לא מדויקים, או להיפך). **לכן, יש לדאוג שפרמטרי המצלמה וירטואלית תועדו ומכוראים לכם**, ולישם אותם בכל חישוב (ובתחליך הכיוון, אם יש). יתרה מזאת, אם אתם מתכוונים להזין את פרמטרי המצלמה לרשות כלשהו (יש מודלים שיכולים לקבל FOV כקלט), וודאו שהדבר עשה. לסייע, מצלמת ה-iReal צריכה להיות כויל-פנימי "קורקע אמת" עבור הפרויקט – השקיעו זמן בהגדرتה לבדוק כפי שתרצו לבצע את החישובים. המקור מזכיר שככל חוסר-התאמת יוביל להטיות שיטתיות²⁸.

כויו עומק והמרה לעומק מוחלט: כפי שנזכר, רשותות כמו MiDaMi מספקות עומק ייחסי. מכיוון שבשרותכם נתונם סינוטיטים עם מרחוק אמיוני, אפשר לנצלם כדי לכייל את הרשות. כאמור מציעים תחילה **כויו סקאללה (Scale)** (**Calibration**) : להריץ את רשות העומק על מגוון דוגמאות סינוטיטיות (נניח פרויימים שבהם הרחפן במרחביהם ידועים מ-20m, 50m, 100m, וכו'), לרשותם עברו כל תמונה את ערך העומק שהרחפן במרחב עומקPredictedDepth (PredictedDepth) ניתן להתחאים פקטור \$z \approx k \cdot \text{TrueDistance}^{true} \times \text{predicted}^{predicted}\$²⁹. למעשה, מונחים שהרשות טועה בקנה-מידה קבוע (למשל מכפילה הכל ב-0.1), ואז \$z\$ יהיה ההופכי של הטועות. את \$z\$ מוצאים למשל ע"י גרסית מינימלי ריבועים על הנתונים²⁹. לאחר מכן, כל פעם שהרשות חזה עומק, נכפיל אותו ב-\$z\$ כדי לקבל מטרים. כאמור הראו שיש לשמר חלק מהנתונים לאימוט – כדי לוודא שהכיוון תקף לטוחה רחוב. **תאיליך זה פשוט לביצוע** (ניתן למשבב כמה שורות עם NumPy/סקיפי) אך קרייטי לבדוק מוחלט. **דgesh**: אם הרשות השתמשו היא ZoeDepth או דומה שמרаш מנסה לתת עומק מטרי, עדין כדאי לבדוק אם יש סטייה ולישם פקטור תיקון אם צריך.

חישוב עומק הרחפן מתוך מפת העומק / תיבת התיחסום: נשאלתם לגבי "הסקת עומק מתוך ה-3D bounding box על האובייקט בלבד". אכן, לאחר שיש לנו תיבת תיחסום של הרחפן וזהו המטרה העיקרית, אין טעם לקחת בחשבון את כל שאר חלקי התמונה עבור חיזוי העומק הסופי. יש כמה דרכים לעשות זאת:

1. שימוש במפת עומק מלאה: בגישה זו, נריץ את רשות העומק על כל התמונה ונקבל **מפת עומק דו-ממדית** (ערך עומק לכל פיקסל). לאחר מכן, נחותור מתוך מפת העומק את האזור של תיבת התיחסום של הרחפן, ונחשבanza עומק מייצג – למשל, אפשר ללקחת את החזון או הממוצע של ערכי העומק בתוך התיבה בתווך עומק הרחפן. החזון עשוי להיות עמיד לרעות/פערים (אם חלק מהתיבה יכול רקע או עצמים אחרים). לחפותן, אפשר ללקחת את **ערך העומק בפיקסל המרכזי** של התיבה כהערה (בහנחה שהמרכז נופל על הרחפן עצמו). כך עשה פחوت או יותר המימוש כאמור: קיבלו עומק מנורמל לרחפן מהמודול, כיילו אותו, ואז השתמשו בו לחישוב המיוקם³¹. שיטה זו קלה מאוד למימוש (מפת עומק כבר זמינה מהרשות).

2. רשות עומק ממוקדת אובייקט: כאן משלבים את התיבה כמידע קלט למודול. אפשרות אחת – **חיתוך (Crop)** התמונה לאזור הרחפן לפני העברת התמונה לרשות העומק. בכך הרשות תתמקדך רק באובייקט. זה יכול לעזור אם הרקע מאוד מבלב או מכיל עומקים שונים מאוד. עם זאת, איבוד הקונטקט שלול דזוקא לפגוע, כי חלק מרמוני העומק קשורין לרקע (למשל גודל הרחפן יחסית לSUBJECT, קו אופק וכו'). אפשרות אחרת היא להזין את התמונה כולה וגם למתת לרשות סוג של מסיכת תשומת לב – למשל, להוסיף עירץ קלט נוספת המסקן את האזור המעניין (התיבה) באחד. יש עבודות שמלבות כך מידע קשיה לתוך רשות

(בדומה ל-map (attention)). זה מתקדם יותר ודורש שינוי בארכיטקטורה, כך שלא נמליץ על כך לשלב התחלתי. ברמת התחליה, מומלץ דזוקא להשתמש בגישה הראשונה: **קבלו את מפת העומק המלא והפיקו ממנה את עומק הרחפן**. זה מהיר, לא דורש אימון נוספת, ומנצל את מלאו כוחו של מודל העומק הכללי. אם תגלו שהרשות מושפעת מאוד מפרטים שלא קשורים לרחפן, תמיד תוכלו לנסות לגזור את התיבה ולבצע **אימון רשות ייעודית** על קורופים של הרחפן. למשל, אימון רשות קתנה שתיקח תמונה קרווף של הרחפן ותלמד לרגום באופן ישר את \$2 (כפי שהצעתי לעיל כחלופה). אבל כדי להשאיר זאת אפשרות לשיפור מאוחר יותר, לאחר שהמערכת הבסיסית עובדת.

3. שימוש בממד הרחפן היודויים: שיטה קלאסית לחישוב עומק אובייקט היא לנצל את גודל האובייקט בפיקסלים בהשוואה לגודלו הפיזי. אם ידוע לכם למשל שהרחפן הוא בקוטר 1 מטר, ותיבת התיחסום שלו הצפיטה היא 50 פיקסלים, אפשר בקרוב לחשב מרחק: $\text{Distance} = \frac{\text{Size_real}}{\text{Size_pixels}}$ (לפי מודל המצלמה הפינוולוי) ³². ואכן, כאמור ציינו שיטה זו כבודקה **משנית** לאימום תוצאות הרשות ³³. בפועל, שיטה גיאומטרית כזו יכולה לתאם ערך מוקוב מאד, שתלווי בבדיקה בתנוחת הרחפן (אם הוא לא ניצב בדיקוק, התיבה היא לא בדיקוק היטל פשוט) וכן מניח שהרחפן כולם נראה. לכן, ניתן להשתמש בהז-**"Sanity check"** – למשל, אם רשות העומק נתנה 100 מ' אך לפי חישוב גודל זה אמור להיות ~50 מ', מדובר שיש בעיה במקורה הזה. אולי משלבים את העירזה במסנן הקלמן בתור מדידה נוספת, או לפחות כינטור. אבל לא הייתה מסתWER על זה כפתרון יחיד, במיוחד אם אין לכם מודל תלת-ממד מלא של הרחפן.

חלוקת טווחים ואייסוף נתונים: עם נתונים סינטטיים, אתם שולטים ביפוי הדוגמאות, ולכן לא צריך כיסוי טוב של כל טווחי המרחק הרלוונטיים. המחקר המצויר מצין שכדי להשקייע יותר ותרד דוגמאות בטווחים גדולים יותר, כי טווח העומק גדולו עם המרחק ³⁴. למשל, אם תחום העניין הוא 150-20 מטר, ודאו שיש מספיק פרויימרים ב-100-150m (כיו' שם הרשות תתקשהści כי הרבה). כמו כן, גנווון תנוחות: הרחפן במרכז הפריים, בצדדים, בתנוחה לכיוונים שונים – כדי שהרשות לא תלמד שהתמונה תמיד בנניה באופן אחד. רצוי גם **גנון רקעים**: שמיים, בניינים, עצים ¹⁰, שכן הרקע משפיע על ניגודיות וקונטראסט מול הרחפן. בניית **תוכנית טיסה** לייצור הנתונים היא רעיון טוב – למשל, במאמר השתמשו בתבניות כמו סריקה ישירה למצלמה, תנעה היקפית סביב המצלמה, גיריד בזווית שונות וכו' ³⁵ – כל זאת כדי לכנות מצבים רבים. שימו לב גם לפרמטרים כמו מגן אויר (ערפל מקשה על עומק – הוסיף סצנות בערפל קל) ¹⁰.

הערכת הביצועים על DATA סינטטי: יתרון של DATA סינטטי הוא שיש לכם Ground Truth מדויק לכל פרט. הגדרו מראש ממדדים לבדיקת הביצועים: למשל **שגיאת מרחק ממוצעת** (Mean Absolute Error) במטרים, ואולי **שגיאהיחסית (%)** עבור טווחי מרחק שונים ³⁶. כך תוכלו לדעת עד כמה המערכת מדויקת בכל טווח. כאמור דיווחו, למשל, על שגיאהיחסית ~8-12% לטווחים 50-100m ³⁷ ³⁸, זהה יכול לשמש לכם כאבן בוחן. כמובן, משום שה拯 סינטטי, יתכן שתגעו אף לדיויקים טובים יותר, תלוי בכמה מגנון הדטה. שמרו חלק מהנתונים כסט בדיקה שלא ראתה הרשות מעולם (למשל סצנות אחרות ב-real Unreal או זווית חדשה) כדי לוודא הכללה.

לסיכום חלק זה, עבודה עם real נתונים لكم שליטה מלאה – נצלו זאת לקבוע את תנאי הניסוי באופן אופטימלי, אך בד בבד דאגו להתאים את הנתונים והמודלים לצורה עקבית (מערכות צירים, כיווצים, פורמטים). תשומת לב לפרטים הללו בשלב ההכנה תחסוך כאבי ראש כשתריצו את המערכת המשולבת.

6. הצעה לIMPLEMENTASI התחלתי – SHILOB קוד לדוגמה

לאחר הסקירה התיאורטית, הנה הצעה לפROYKT התחלתי והוכחת יכולת, צעד-אחר-צעד, עם רכיבי הקוד העיקריים:

שלב א': אימון/טעינת מודל זיהוי (YOLO) וקבלת Bounding Box – אספו מספר תמונות סינטטיות וצרו קובץ אימון עבור YOLO (פורמט YOLO): לכל תמונה קובץ טקסט עם קטגוריה וקווארדינטות תיבת מנומולות. אימנו מודל YOLO קטן (למשל 55YOLOv5) על זה – או דלגו על אימון אם יש לכם דרך אחרת לקבל את תיבת הרחפן (אולי מנוע המשחק יכול לתת box bounding ישירות). לאחר שיש מודל, כתבו קוד Python שמשתמש בו כדי לקבל את תיבת התיחסום מהתמונה. עם ספריית ultralytics זה נראה如下:

```

from ultralytics import YOLO
model = YOLO('path/to/custom-yolo.pt') # טען מודל מאומן
results = model(image) # הרצ על תמונה בודדת
boxes = results[0].boxes.xyxy.cpu().numpy() # נניח מחזיר מערך [x1,y1,x2,y2]

```

קבלו את התיבת (או התיבה הרלוונטיות אם אייכשו יש יותר).

שלב ב': הרצת מודל عمוק על התמונה המלאה – נשתמש, למשל, ב-MiDaS MiDaS הטרום-מאומן. בעזרת PyTorch Hub אפשר לטען את הדגם והטרנספורמציות בקלות :

```

import torch, cv2
midas = torch.hub.load("intel-isl/MiDaS", "MiDaS") # טען מודל גדול DPT-Large
midas_transforms = torch.hub.load("intel-isl/MiDaS", "transforms")
transform = midas_transforms.default_transform # טרנספורמציה מתאימה
for img in images:
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    midas.to(device).eval()

    img = cv2.imread("synthetic_frame.jpg")
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # המודל מצפה לRGB
    input_batch = transform(img_rgb).to(device)
    with torch.no_grad():
        prediction = midas(input_batch)
        prediction = torch.nn.functional.interpolate(
            prediction.unsqueeze(1),
            size=img_rgb.shape[:2], # התאם בחזרה לגודל התמונה המקורי
            mode="bicubic",
            align_corners=False
        ).squeeze()
    depth_map = prediction.cpu().numpy()

```

הקוד הנ"ל יפיק לנו מטריצת עומק `depth_map` בגודל התמונה. הערכים הם ביחס שריורי (לא במטרים). אפשר להדפיס כמה ערכים כדי להתרשם מהסקלה.

שלב ג': חילוץ עומק הרחפן מתוך המפה – נשתמש בתיבת התיחום שקיבלו בשלב א'. נניח שהוא במשתנה `box` בפיקסלים (ברזולוציית התמונה המקורי). נחלץ את איזור העומק וניתן הערכתה:

```

x1, y1, x2, y2 = map(int, box)
drone_region = depth_map[y1:y2, x1:x2]
drone_depth_pred = np.median(drone_region) # עומק יחסית חזוי של הרחפן
print("Predicted relative depth:", drone_depth_pred)

```

כך קיבלנו את עומק הרחפן כמו שחזזה מודול MiDaS.icut, בהינתן ערך זה, נוכל **לכפול** אותו. אם למשל יש לנו מה-Unreal את המרחק האמיטי של הרחפן בפריים זהה (נניח 50 מטר), נוכל לחשב פקטור כיוול $s = \frac{\text{distance}}{\text{drone_depth_pred}}$. כמובן, נעשה זאת על קבוצה וርבה של דוגמאות וננתאים s אופטימלי (למשל ממוצע או גרסיה). נוכל גם להעריך את טווח הרשת לפני כיוול.

שלב ד': כיוול סקאללה ואמיות – אספו N דוגמאות (תמונהות) עם רחפנים במרחקים ידועים שונים. הריצו את הקוד משלבים א-ג על כלן, וצרו רשימה של זוגות: $(\text{pred}_1, \text{true}_1), (\text{pred}_2, \text{true}_2), \dots$. באמצעות NumPy אפשר בקלות לחשב את פקטור הכוול שמשמעותו:

```
import numpy as np
preds = np.array([pred1, pred2, ...])
trues = np.array([true1, true2, ...])
s_optimal = np.sum(preds * trues) / np.sum(preds**2)
```

זו נוסחת גרסיה ליניארית פשוטה למציאת s כך $s = \frac{\sum \text{pred} \cdot \text{true}}{\sum \text{pred}^2}$. השתמשו ב- $\approx 70\%$ מהנתונים זהה (כמו שמצוין המאקרו ³⁰) וביתר כדי לוודא שהשגיאה *indeed* קטנה ממשועותית אחרי כיוול. תוכלו לישם את התיקון בכל הרצה: $s_{\text{optimal}} = \frac{\sum \text{pred} \cdot \text{true}}{\sum \text{pred}^2}$.Cut s_{optimal} אמור להיות הערכת מרחק במטרים.

שלב ה': המרת לקוואורדינטות מרחביות (אופצונלי): אם תרצו בשלב הזה גם להפיק את מיקום ה- X, Y, Z של הרחפן, זה פשוט כביש עומק מותוקן. יש להכניס את ערך Z ואת מרכז התיבה (או מרכז המסה של הפיקסלים של הרחפן) לנוסחאות:

$$X = \frac{(x_{\text{center}} - c_x) \cdot Z}{f_x}, Y = \frac{(y_{\text{center}} - c_y) \cdot Z}{f_y}$$

כאשר (x, y, z) הם מרכז התמונה (מחזיות הרוחב/גובה) ו- (c_x, c_y, c_z) אורך המוקד בפיקסלים בכיוון אופקי/אנכי. אותן כדי לחשב לפי הגדרות המצלמה שלהם (ראו חלק 5, "התאמת פרמטרי מצלמה"). כך תקבלו נקודה תלת-ממדית. אפשר להשוות זאת לנקודות ה-GT-מ-Unreal כדי לוודא שהכל נכון.

שלב ו': שילוב מסנן קלמן (לניתוח המשכיות בווידאו): אם תעבדו על רצף תמונות (וideo), מומלץ לעתוף את התוצאה של שלב ה' בפילטר קלמן כדי להחליק את המסלול. ניתן להשתמש בספריית FilterPy:

```
from filterpy.kalman import KalmanFilter
kf = KalmanFilter(dim_x=6, dim_z=3) # example: state [X, Y, Z, Vx, Vy, Vz],
measurement [X, Y, Z]
# בהתאם למודל התנועה (למשל תנועה קבועה + רעשים) ...
# F, H, Q, R, gדרת מטריצות
# 초기ול מצב התחלתי ע"ס מדידה ראשונה
for measurement in measurements: # loop over frames
    pred_state = kf.predict()
    kf.update(measurement) # measurement could be [X_meas, Y_meas, Z_meas]
    filtered_state = kf.x
```

ברגע שהקלמן מכיל (ניתן להיעזר במדריכים או בקוד מוכן), הוא יקטין הבדלי מסגרת-למסגרת. כאמור צוין שהקלמן מסיים לנטרל ריצודים ואף לחזות בהיעדר זיהוי בפריים בודד ⁸. בשלב ראשוןiali תבדקו את המSEN offline על נתוני מסלול שלמים.

שלב ז': הרצת הכל ובדיקה הביצועים: כתע כשל הרכיבים קיימים – זיהוי YOLO, הערכת עומק, כיוול, המרה -(אופציונלי) קלמן – שלבו אותו לתוכנית אחת שרצה על רצף פרויימים. בצעו השוואת מול Ground Truth בכל שלב: לפני כיוול ואחריו, לפני קלמן ואחריו. חישבו את ה-MAE, אחוזי שגיאה, וכו'. לדוגמה, יתכן ותראו שבטווות 20-50m השגיאה המוחלטת היא ± 2 (כ-4%), ובטווות 100-150m השגיאה ± 12 (כ-12-18%)⁴⁰. נתונים אלה דומים למה שהושג במאמר, ומהווים אישור שהשיטה עובדת בתחום המצוופה. אם תראו שగיאות גדולות בהרבה, חיזו לשלבים קודמים לבדוק היכן הפער – האם הזיהוי *bbox* לא מדויק? האם מודול העומק מתבלבל בתנאים מסוימים? האם הכוול לא מתאים לכל הטעות (יתכן שתצטרכו כיוול מקרים – *piecewise* – כפי שנראה אם סטיית הסקלה לשונה בין קרוב לרחוק).⁴¹

לסיכום, לאחר השלמת השלבים לעיל, תהיה לכם מערכת התחלתית מתפקדת: היא תקבל תמונת Unreal ותחזיר מרחוק רחפן. המשך סקר את ההיבטים העיקריים – מהגדרת המשימה, דרך הנושאים הטכניים ללמידה, הכלים למימוש, המודלים המומלצים ועוד לדגשים מיוחדים בסביבת Unreal. בהמשך, תוכלו להעמיק בכל ריבוב: לשפר את רשת העומק (למשל Fine-tune על הסצנה הספרטיפית, או ניסיון מודלים אחרים), לשכלל את מודול הזיהוי (אולי מעקב אובייקט בין פרויימים), ולנסות את המערכת על וידאו אמיתי כדי לבחון הכללה. בהצלחה רבה בפרויקט הגמר שלכם! שתיהה למידה פוריה ו מוצרחת.

מקורות וקריאה נוספת:

- מסמך טכני Monocular Drone Detection and 3D Localization – Technical Paper –
הסינטטי, כולל פירוט הארכיטקטורה, ניתוניים ותובנות ⁷ ² ¹ ועוד. מומלץ לקרוא אותו לעומק כהשראה ישירה.
- קוד ופרויקטים רלוונטיים בGIT: isl-org/ZoeDepth – קוד *isl*-*MiDaS* – intel-isl/MiDaS – קוד *yolov5* – קוד *YOLOv5* לאימון והרצה.
- בלוגים ומדריכים: הסבר על *ZoeDepth Medium* על ⁴² ¹⁷ MiDaS והטמעתו, בלוג המסביר את הגישה המשלבת תארוטי.

תהנו מהתהילה הפיתוח, ואל תשחחו לטעוד את ההתקדמות והניסויים לצורך הדוח הסופי. בהצלחה!

37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 16 13 12 11 10 9 8 7 6 5 4 3 2 1

Monocular Drone 3D Localization - Technical Paper (2).pdf ⁴¹ ⁴⁰ ³⁹ ³⁸

file:///file_0000000027b87207b355682aba141446

GitHub - AbirKhan96/Intel-ISL-MiDaS ²² ¹⁹ ¹⁵ ¹⁴

<https://github.com/AbirKhan96/Intel-ISL-MiDaS>

| Monocular Depth Estimation using ZoeDepth : Our Experience | by Bhaskar Bose ⁴³ ⁴² ²⁰ ¹⁸ ¹⁷

Medium

<https://medium.com/@bhaskarbose1998/monocular-depth-estimation-using-zoedepth-our-experience-42fa5974cb59>

Depth Anything ²¹

[/https://depth-anything.github.io](https://depth-anything.github.io)