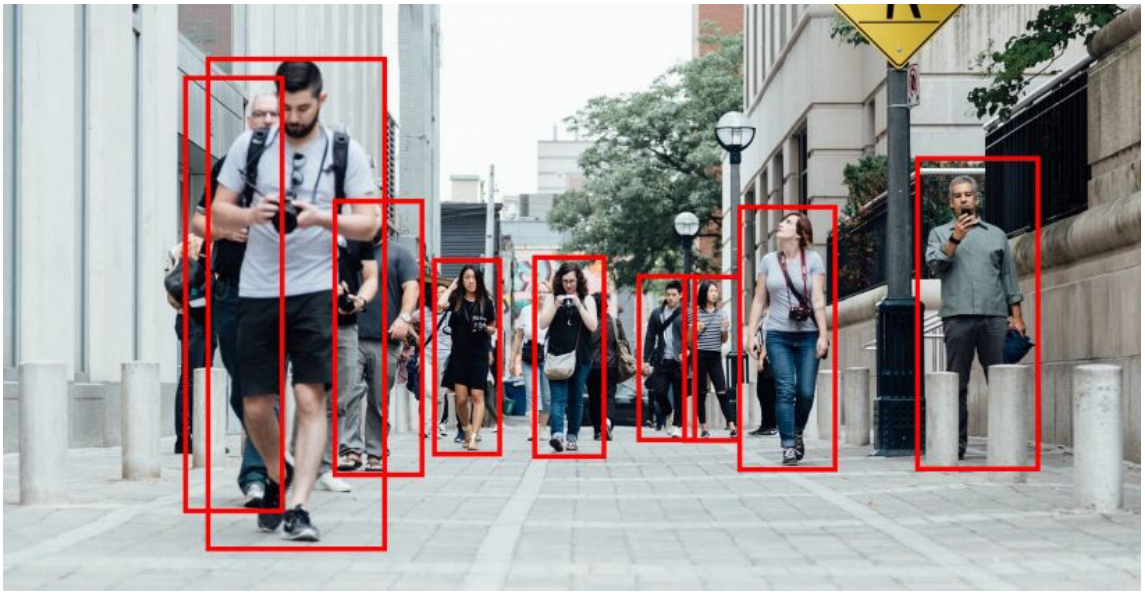
מימוש אלגוריתם לביצוע

Human Tracking

לירן מנצורי, אריאל ורבין, ניתאי דלגושן



מבוא

השלב האחרון במימוש האלגוריתם הוא לייצר מערכת מעקב אחר בני אדם בסרטון. מעקב אחרי בני אדם מהווה מרכיב חיוני במימוש יישומים רבים, כמו מערכות בטיחות, אנליטיקה מתקדמת, ומשחקי וידאו מבוססי תנועה – כמו במקרה שלנו.

"מעקב אחרי בני אדם" מתייחס לתהליך של זיהוי עקבי של בני אדם המופיעים ברצף של פריימים. הכוונה היא להעניק לכל אדם אינדקס ייחודי (ID) שיישאר איתו גם כאשר הוא זז ממקום למקום. זיהוי זה מאפשר לנו לעקוב אחרי כל אדם במרחב לאורך זמן, תוך שמירה על אינדיקציה ברורה לזהות אותו בין פריימים עוקבים.

בהקשר שלנו, אנו זקוקים למערכת מעקב אחר בני אדם לשם **זיהוי השחקנים**: עלינו להבין לאורך המשחק אילו מן האנשים בפריים הם השחקנים, ואיזה אדם מתאים לאיזה שחקן. למשל, נניח שהצלחנו לנתח את הפוזיציה של כל האנשים בסרטון ולהפיק לכל אדם ניקוד. נשאלת השאלה איזה ניקוד צריך להביא לאיזה שחקן, והאם בטעות בזבזנו זמן וחישבנו ניקוד של אדם שבכלל נמצא ברקע ולא משתתף במשחק?

כלומר, נניח שהגדרנו את המשחק להיות עבור 2 שחקנים – שחקן א' ושחקן ב'. על המשחק להשתמש במעקב אחר בני אדם ובעצם למפות את שחקן א' לID של הבנאדם המתאים שמתקבל במעקב, וכן למפות את שחקן ב' לID של הבנאדם השני. כך, בכל פריים אנו ננתח רק את הפוזיציה של האנשים שמתאימים לID של שחקן א' ושחקן ב', ונדע איזה ניקוד להביא לאיזה שחקן.

סוגי מערכות מעקב

- **מעקב מבוסס הופעה ויזואלית**: בגישה זו, האלגוריתם עוקב אחרי המראה החיצוני של האובייקט, כלומר צורת הגוף, צבעים, והמאפיינים הוויזואליים של האדם. אלגוריתמים אלה נדרשים להשתמש בנתונים כמו histograms של צבע או טקסטורות כדי להשוות בין אנשים בפריימים שונים. היתרון של גישה זו הוא שהיא שומרת על עקביות גם בתנועות פתאומיות או במקרים שבהם אנשים קרובים מאוד זה לזה. אמנם, גישה זו פחות מתאימה למשימות real-time שכן היא דורשת עיבוד של התמונה והפקת מידע ממנה תוך שימוש בכלים של ראייה ממוחשבת, שעלולים לפגוע במהירות האלגוריתם.

- **מעקב מבוסס מיקום ומהירות**: בגישה זו, אנו משתמשים במידע על המיקום הנוכחי ומהירות התנועה של האובייקטים בין פריימים עוקבים כדי לנבא את מיקומם בפריימים הבאים. שיטות כמו אלגוריתם Kalman Filter מאפשרות חיזוי המבוסס על מידע זה, כאשר המטרה היא למנוע איבוד עקב מעבר מהיר של אנשים בפריימים. גישה זו מותאמת למשימות real-time ויעילה במיוחד כשמדובר בתנועות חלקות והמשכיות, אך היא עלולה להיכשל במקרים של שינויים פתאומיים בתנועה או כשהפריימים מרווחים בזמן.

המודל בו נשתמש לזיהוי בני אדם בתמונה – YOLOv5s-u [1]

על מנת לבצע מעקב אחר בני אדם, דרישה הכרחית היא שימוש במודל למידה לזיהוי בני אדם בתמונה, כלומר מודל אשר בהינתן תמונה יודע לתחום כל אדם בboundary box (או bbox) מתאים. הbbox האלו משמשות בתור הנקודה ההתחלתית באלגוריתם המעקב. כלומר, אם המודל מזהה בנאדם כלשהו ותוחם אותו בbbox, מערכת המעקב מביאה לבנאדם זה ID כלשהו, כך שגם אם הוא יזוז בפריימים הבאים, מערכת המעקב תדע שמדובר באותו האדם ותביא לו את אותו הID לאורך כל הפריימים.

המודל YOLOv5s-u הוא גרסה של המודל הפופולארי YOLO (You Only Look Once) לזיהוי אובייקטים. המודל הותאם במיוחד לעבוד במהירות ובדיוק גבוהים. YOLOv5 פותח ע"י החברה Ultralytics המתמקדת בפיתוח מודלים לזיהוי אובייקטים. משום שהמודל עוצב במיוחד עבור משימות real-time, הוא אידיאלי עבור המקרה שלנו. המודל בפועל משתמש לזיהוי אובייקטים מקטגוריות רבות (כיסאות, כלבים, בקבוקים וכדומה) אך אנו נשתמש בו רק לצורך זיהוי הקטגוריה של בני אדם.

[1] Ultralytics. "YOLOv5." GitHub, 2020, https://github.com/ultralytics/yolov5.

נושא מקדים – Kalman Filter [2]

מסנן קלמן (Kalman Filter) הוא כלי מתמטי חזק המשמש להערכת מצב של מערכת דינמית על סמך סדרת מדידות חלקיות ורועשות. הוא נמצא בשימוש רחב ברובוטיקה, ניווט, כלכלה, עיבוד אותות ובתחומים רבים נוספים. מסנן קלמן פופולרי מכיוון שהוא מספק אמצעי חישובי יעיל לשלב מדידות רועשות כדי לייצר הערכה מיטבית של מצב המערכת. הוא נקרא על שם רודולף א. קלמן, שתיאר אותו לראשונה בשנת 1960.

לצורך הדוגמה, נניח שאנו מנסים לעקוב אחר רכב הנע בכביש מעורפל תוך שימוש בתמונות תקופתיות של מיקומו. עם זאת, תמונות אלו רועשות או כוללות שגיאות בשל הערפל. מסנן קלמן יעזור לנו "לנחש" את המיקום הנוכחי והעתידי של הרכב בצורה מדויקת יותר על ידי שילוב של המידע הרועש עם מודל המתאר כיצד הרכב נע.

מושגי מפתח במסנני קלמן

1. **הערכת מצב (State Estimation)**: בכל רגע נתון אנו רוצים להעריך את "המצב" של המערכת. לדוגמה, במקרה של רכב נע, המצב יכול לכלול את מיקום הרכב ואת מהירותו.
2. **חיזוי ותיקון (Prediction and Correction)**: מסנן קלמן פועל בשני שלבים:

- שלב החיזוי (Prediction Step): בו, המסנן חוזה את מצב המערכת העתידי בהתבסס על המצב הנוכחי ומודל הדינמיקה של המערכת.
- שלב התיקון/עדכון (Correction Step): בו, המסנן מתקן את המצב הצפוי באמצעות שילוב של נתוני מדידה חדשים.

3. **הנחות גאוסיאניות (Gaussian Assumptions)**: מסנן קלמן מניח ששני סוגי הרעש, רעש תהליך ורעש מדידה, הם גאוסיאניים, כלומר מתפלגים לפי עקומה פעמונית. זה מאפשר שימוש בממוצע מטריצת שונות המשותפת כדי לתאר את אי-הוודאות.

תיאור מתמטי של מסנן קלמן

כדי להבין את מסנן קלמן באופן מתמטי, עלינו להגדיר את מצב המערכת, את דינמיקת המערכת, ואת המדידות.

[2] כלל ההסברים נלקחו מהאתרים המצוינים בנספחים

תיאור מתמטי של שלב החיזוי

מסנן קלמן מבוסס על מערכת דינמית לינארית המנוקדת בזמן. מצב המערכת בזמן k מיוצג על ידי וקטור x_k.

הקשר בין המצב הנוכחי x_k לבין המצב הבא x_{k+1} מתואר על ידי:

$$x_{k+1} = A_k x_k + B_k u_k + w_k$$

כאשר,

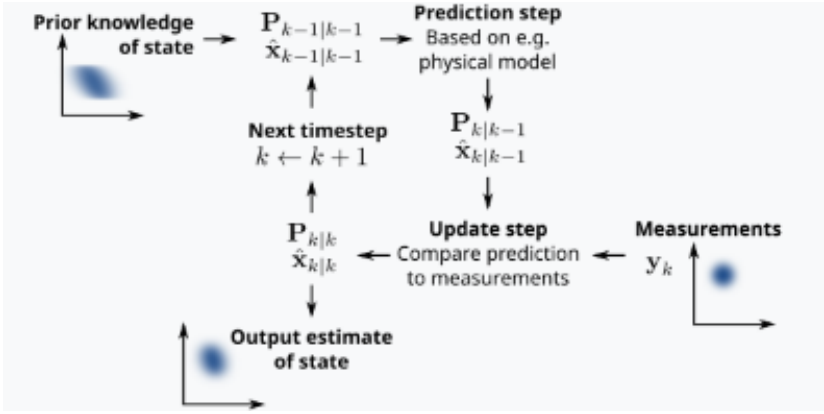
- x_k הוא וקטור המצב בזמן k (לדוגמה, מיקום ומהירות של רכב).
- A_k הוא מטריצת המעבר, המייצגת כיצד המצב משתנה מרגע אחד לרגע הבא.
- B_k הוא מטריצת הקלט, המתארת כיצד קלטי השליטה u_k משפיעים על השינוי במצב.
- u_k הוא וקטור הפקודה (למשל, תאוצה, כוח קלט).
- w_k הוא רעש התהליך, אשר מתווסף לכל מעבר במצב ומייצג שגיאות או אי-ודאות במערכת, אשר מניחים שהוא גאוסיאני עם ממוצע אפס ומטריצת שונות המשותפת Q_k.

תיאור מתמטי של שלב התיקון

המדידה z_k בזמן k קשורה למצב x_k על ידי. $z_k = H_k x_k + v_k$.

כאשר,

- z_k הוא וקטור המדידה בזמן k, כלומר הערכים שנמדדו במערכת.
- H_k היא מטריצת המדידה, שממפה את המצב האמיתי x_k למרחב שבו מתבצעת המדידה.
- v_k הוא רעש המדידה המייצג אי-דיוק במדידה, אשר מניחים שהוא גאוסיאני עם ממוצע אפס ומטריצת שונות המשותפת R_k.



שלבי החיזוי והתיקון באלגוריתם

כפי שציינו, מסנן קלמן פועל בשני שלבים עיקריים: שלב החיזוי ושלב התיקון (עדכון). נראה כיצד שלבים אלו מתבצעים.

- **שלב החיזוי**

חיזוי המצב: אנו חוזים את המצב הבא ($\hat{x}_{k\,|\,k-1}$) על סמך היסטוריית המצבים הקודמים, בהתבסס על הנוסחה,

$$\hat{x}_{k\,|\,k-1} = A_{k-1}\,\hat{x}_{k-1\,|\,k-1} + B_{k-1}\,u_{k-1}$$

כאשר,

- $\hat{x}_{k\,|\,k-1}$: הערכה צפויה של המצב בזמן k בהתבסס על המצב הקודם.
- $\mathrm{A}_{\mathrm{k}-1}$: מטריצת מעבר מצב, שמתארת כיצד המצב מתפתח מ-k-1 לk .
- $\hat{x}_{k-1\,|\,k-1}$: הערכת מצב בזמן $k-1$ לאחר התיקון.
- B_{k-1}: מטריצת קלט, שמדגימה את השפעת הקלטים על המצב.
- u_{k-1}: קלט בזמן $k-1$.

חיזוי מטריצת שונות המשותפת: בהתבסס על הנוסחה,

$$P_{k\,|\,k-1} = A_{k-1}\,P_{k-1\,|\,k-1} + A_{k-1}^{T}\,Q_{k-1}$$

כאשר,

- $P_{k\,|\,k-1}$: היא מטריצת שונות השגיאה הצפויה בזמן k.
- $P_{k-1\,|\,k-1}$: היא מטריצת שונות השגיאה בזמן $k-1$ לאחר התיקון.
- Q_{k-1}: מטריצת שונות רעש התהליך, המייצגת את חוסר הוודאות במודל התהליך.

- **שלב התיקון (עדכון)**

חישוב רווח קלמן: רווח קלמן K_k בעצם קובע עד כמה משקל יש לתת למדידה החדשה (על סמך כמה אנחנו מאמינים לחיזוי שביצענו – אם למשל רמת הוודאות שלנו לגבי החיזוי נמוכה, אזי המדידה הבאה תקבל יותר משקל). חישוב רווח קלמן מתבסס על הנוסחה,

$$K_k = P_{k\,|\,k-1}\,H_k^{T}\left(H_k\,P_{k\,|\,k-1}\,H_k^{T} + R_k\right)^{-1}$$

עדכון המצב:

$$\hat{x}_{k\,|\,k} = \hat{x}_{k\,|\,k-1} + K_k\,(z_k - H_k\,\hat{x}_{k\,|\,k-1})$$

- $\hat{x}_{k\,|\,k}$: הערכת מצב מעודכנת בזמן k.

- z_k : מדידה בזמן k.
- H_k : מדידה צפויה בהתבסס על ההערכה הצפויה של המצב.

עדכון מטריצת שונות המשותפת : ע״פ הנוסחה

$$P_{k \mid k} = (I - K_k H_k) P_{k \mid k-1}$$

כאשר,

- $P_{k \mid k}$: מטריצת שונות השגיאה המעודכנת בזמן k
- I : מטריצת האחדות, שיש לה את אותן ממדים כמו P
- $(I - K_k H_k)$: מייצגת את ההפחתה בחוסר הוודאות לאחר שילוב המדידה

פסאודו קוד של האלגוריתם

על מנת להבין טוב יותר את מהלך האלגוריתם, צירפנו פסאודו קוד שלו.

Initialize:

$\hat{x} = \hat{x}_0$ # הערכת מצב ראשונית

$P = P_0$ # מטריצת שונות השגיאה הראשונית

For each time step k:

שלב החיזוי

$\hat{x}$_prior = A * $\hat{x}$ + B * u # חיזוי המצב : הערכת מצב חדשה בהתבסס על המצב הקודם

P_prior = A * P * A^T + Q # חיזוי שונות השגיאה : עדכון מטריצת השגיאה בהתאם

שלב העדכון

K = P_prior * H^T * $(H * P\_prior * H^T + R)^{-1}$ # חישוב רווח קלמן

$\hat{x} = \hat{x}$_prior + K * (z - H * $\hat{x}$_prior) # עדכון המצב : שילוב המדידה החדשה

P = (I - K * H) * P_prior # עדכון מטריצת שונות השגיאה

Output $\hat{x}$, P # הפלט בכל שלב זמן : הערכת המצב המעודכנת ומטריצת השגיאה המעודכנת

אלגוריתם SORT

אלגוריתם SORT (Simple Online and Realtime Tracking) שבו השתמשנו עוקב אחרי אובייקטים (במקרה שלנו, אנשים) בתמונה בזמן אמת בעזרת שימוש בנתונים כמו מיקום האובייקט ותנועתו לאורך פריימים עוקבים וחוזה את מיקומו בפריימים הבאים. למעשה, האלגוריתם עושה שימוש ב- Kalman Filter כדי לחזות את מיקום האובייקטים בפריימים עתידיים.

לאחר מכן, כשהאלגוריתם כבר ניבא את מיקום האובייקטים, הוא מנסה לזווג (כלומר, למצוא התאמה של אחד לאחד) בין החיזויים למיקומים האמיתיים של האובייקטיבים, זאת על מנת שאנחנו נוכל לעקוב אחרי השחקנים לאורך המשחק מבלי להתבלבל ביניהם. כלומר,

1. לכל שחקן (בעל ID מסוים הניתן לו), נבא היכן סביר שיהיה בפריים הבא.
2. בהינתן כל המיקומים האמיתיים של כל האנשים בפריים הבא, התאם כל שחקן (בעל ID מסוים) למיקום כלשהו של אדם שזוהה בפריים, קבע שזה אותו השחקן וסמן אותו באותו הID.

ההתאמה הזאת מתבצעת באמצעות מטריצת IoU (Intersection Over Union) כדי להשוות בין המיקומים החזויים לבין המיקומים בפועל (IOU זו מטריקה נפוצה ביותר למדידת חפיפה בין 2 מלבנים. ככל שהIOU גדול יותר, סביר יותר שיש התאמה בין שני המלבנים שמתארים את החיזוי והמיקום בפועל. אזי, מחשבים את הIOU של כל זוג (חיזוי ומיקום) ומבצעים אלגוריתם תכנות לינארי (LP, עם מטריצת עלויות שתהיה ($-IOU$) אותה נרצה למזער – שזה שקול למקסום של ($+IOU$)) שמטרתו למצוא זיווג ביניהם.

מהלך האלגוריתם המקורי

1. **חיזוי בעזרת Kalman Filter**: עבור כל אובייקט שעוקבים אחריו, אלגוריתם Kalman Filter משמש לניבוי מיקום האובייקט בפריימים הבאים.
2. **זיווג בעזרת IOU**: האלגוריתם מתאים בין החיזויים של האובייקטים לבין המיקומים האמיתיים של אובייקטים בפריימים החדשים באמצעות חישוב ה- IOU.
3. **במקרה שלא הייתה התאמה עבור אדם כלשהו שזוהה – ניצור לו מזהה חדש**: האלגוריתם ייתן לאותו אובייקט ID ייחודי, שיאפשר מעקב שלו לאורך הפריימים הבאים. כל אובייקט חדש שלא היה מזוהה קודם לכן מקבל ID חדש. דוגמה לתרחיש כזה היא למשל אם אדם נכנס לפריים של המצלמה.
4. **עדכון המעקב**: האלגוריתם מעדכן את המעקב על פי הזיהוי בפריימים החדשים, ומסיר אובייקטים שזמן העדכון שלהם עבר (כלומר אובייקטים שלא זוהו במשך זמן רב מספיק).

באופן כללי, אלגוריתם SORT תוכנן למעקב בזמן אמת, ומשתמש בגישות מתמטיות פשוטות יחסית (כמו Kalman Filter ו- IOU) על מנת לשמור על יעילות טובה. הבעיה העיקרית באלגוריתם היא פספוס מעקב עקב מצבים מורכבים כמו תנועה פתאומית או כאשר אובייקטים קרובים מאוד אחד לשני.

שיפור האלגוריתם עבור המקרה הפרטי שלנו

במקרה שלנו, שהמשחק הוא משחק ריקוד שבו התנועות מתבצעות במהירות וישנם שינויים פתאומיים רבים, היינו צריכים להתאים ולשפר את אלגוריתם SORT כך שכמעט ולא יבצע שגיאות במעקב אחרי השחקנים ובזיווגם. לשם כך ביצענו מספר שינויים:

- בתור התחלה – האלגוריתם מתבצע **בדיוק** כמו האלגוריתם SORT המקורי שתואר לעיל – עוקבים אחרי כל האובייקטים ומזווגים אותם לפי ה- IOU ביניהם.
- בשלב מסוים בריצת המשחק, האלגוריתם יקבל מערך שמכיל את המזהים (ID) של השחקנים בלבד. כעת, נרצה לעקוב רק אחריהם ולזווג רק אותם למיקומים האמיתיים שנקבל.
- יחד עם זאת – אנחנו לא נרצה לעולם למחוק מעקב של שחקן – גם אם הוא לא זוהה למשך מספר רב של פריימים, אנחנו נשתמש בתחזיות שנקבל מ- Kalman Filter, עד אשר השחקן יזוהה שוב. באותו אופן גם לא נרצה להוסיף מעקב של אובייקט חדש – כי בשלב הזה כבר אנחנו מניחים שאנחנו יודעים מי השחקנים ואי אפשר להוסיף כעת שחקן.

בנוסף, ברגע שקיבלנו את רשימת השחקנים, אנחנו נבצע את הזיווג בין החיזויים למיקומים באופן הבא:

1. אם הצלחנו לזווג באמצעות IOU כמו שצריך, כלומר כל מעקב של שחקן זווג למיקום אמיתי – אז מעדכנים את המעקב בהתאם לזיווג החדש וממשיכים כרגיל.

2. אחרת, אם יש מעקב השייך לשחקן כלשהו (<u>לפחות אחד</u>) אשר לא זווג לאף אחד מהמיקומים האמיתיים (נדגיש כי ברוב המקרים זה לא יקרה, אך עדיין יש אפשרות שכן. למשל, אם הייתה בעיה בזיהוי המיקומים האמיתיים של מודל הלמידה, או אפילו אם היה שינוי משמעותי במיקום של השחקנים בין הפריימים, למשל קפיצה מהירה בריקוד). במקרה כזה:

   a. אם זה הפריים הראשון שזה קורה (במובן שבו בפריים הקודם זה לא קרה), אנחנו נכנסים למצב שקראנו לו *Recovery*. במצב הזה, אנחנו לא נעדכן את הזיווג <u>לאף שחקן</u>, אלא מקדמים את כל המעקבים של השחקנים בהתאם לחיזוי, עם אותו זיווג כמו מקודם. עבור פריים זה אנו נקבע שהמיקומים של כל השחקנים הם המיקומים שהאלגוריתם חזה עבורם, ללא התחשבות בזיהוי של אף אחד.
   b. בזמן שאנו בRecovery, אנחנו נרצה לזווג בהתאם ל(MSE (Mean Squared Error, כלומר נרצה למזער את תוחלת ריבועי ההפרשים בין החיזויים למיקומים האמיתיים. באופן הזה, גם אם היו איזשהם קפיצות מהירות של השחקנים, ברוב המקרים הMSE עדיין יחזיר את הזיווג הנכון, כי בניגוד ל- IOU, הוא תמיד יחזיר פתרון שלוקח בחשבון את רמת ההתאמה של כל השחקנים ויחזיר את התוצאה הכללית הטובה ביותר, בניגוד לIOU שיכול להיות שיחזיר תוצאה שתהיה חד משמעית עבור שחקן ספציפי, אבל שגויה לגמרי עבור שחקן אחר.

לאחר שהצלחנו להתאים את כל השחקנים מחדש, נצא ממצב הRecovery ונמשיך כרגיל.

כלומר, סך הכל, ביצענו את השינויים הבאים:

1. **הוספת מערך שמכיל את המזהים של השחקנים בלבד (protected):** שחקנים צריכים לקבל ID קבוע שמיוחס להם לאורך כל המשחק, גם אם לא זוהו בפריימים מסוימים. לכן הוספנו שלב שבו אנחנו מקבעים את השחקנים ומחליטים שהחל מעכשיו הם לא יוסרו, גם אם הם לא זוהו בפריימים מסוימים. המערכת תמשיך לעקוב אחריהם על בסיס תחזיות של Kalman Filter בלבד, עד לחידוש הזיהוי בפריימים הבאים.
2. **Recovery mode ו- MSE:** במקרים בהם השחקנים לא מזווגים למשך מספר פריימים, האלגוריתם נכנס למצב התאוששות. במצב זה, האלגוריתם לא מסתמך על ההתאמה הרגילה של IOU אלא על חישוב MSE (Mean Squared Error), שמודד את הסטייה בין החיזוי לבין המיקום בפועל של כל אובייקט. כך, ניתן להחזיר את השחקנים למעקב באופן מדויק גם לאחר תקופה בה לא זווגו בפריימים.

בחינת השינויים עבור מצבי קצה

אנו הבחנו במספר מצבי קצה אשר גורמים לאלגוריתם SORT המקורי להתבלבל.

- **קושי**: פספוס פתאומי של השחקן במודל הלמידה גורם להפסקת המעקב אחר אותו אדם. ברגע הבא שבו המודל יחזור לזהות את האדם – SORT יחשוב שמדובר באדם חדש ויביא לו ID חדש.
פתרון: כעת, גם אם שחקן מתפספס לרגע במודל הלמידה, אנו נשתמש בחיזוי שחושב על מנת להעריך את המיקום שלו. בסיכוי סביר מיקום זה אכן יהיה איפה שהוא נמצא ויאפשר למודל הHPE להעריך את הפוזיציה שלו ולתת לו את הניקוד הנכון. בזמן שהשחקן אינו מזוהה, שימוש בMSE יבטיח כי הזיווג של שאר השחקנים לא יפגע.

- **קושי**: נניח אם יש מצב שבו הריקוד כולל תנועה פתאומית ימינה. כל אדם יעמוד בדיוק במיקום של השחקן שהיה בימינו בפריים הקודם. לכן, ערך הIOU בין <u>הזיהוי</u> של שחקן <u>לחיזוי</u> של השחקן מימינו, יצא גבוה מאוד. ואלגוריתם SORT יחשוב שמדובר בשחקן מימין למרות שזהו לא המצב. SORT עוד יחשוב שהאדם השמאלי ביותר נעלם, והופיע אדם חדש מימין.
פתרון: במצב כזה כאמור SORT לא ימצא זיווג עבור השחקן השמאלי, לכן נעבור למצב של Recovery ונזווג על פי מטריקת MSE אשר משום שהיא מחשבת את המרחקים <u>בריבוע</u>, היא תבטיח שהזיווגים יהיו נכונים: נניח מצב של 2 שחקנים, כאשר בפריים הבא שני השחקנים לוקחים צעד פתאומי של d פיקסלים ימינה כך ששחקן A עומד במקום של שחקן B. התאמה שגויה תיתן ערך של $(2d)^2 + 0^2 = 4d^2$ לערך הMSE. לעומת זאת התאמה נכונה תיתן ערך $d^2 + d^2 = 2d^2 < 4d^2$ עבור ערך הMSE.

אתחול המשחק – זיהוי השחקנים

נשים לב שאנו עדיין צריכים להבין בתחילת השחקן אחרי אילו מהאנשים להתחיל לעקוב. לשם כך, עבור זיהוי השחקנים בתחילת המשחק, מימשנו דף המבקש מכל מי שמעוניין לשחק להרים ידיים. השתמשנו במודל הHuman Pose Estimation לצורך זיהוי האם בנאדם מרים ידיים. לאחר שמספיק אנשים מרימים ידיים, שמרנו את האינדקסים של כל אחד מהם אשר התקבל מאלגוריתם המעקב והתחלנו את המשחק.

נספחים - מקורות מידע עבור החלק של מסנן קלמן

1. Welch, G., & Bishop, G. "An Introduction to the Kalman Filter." University of North Carolina at Chapel Hill.
https://www.cs.unc.edu/~welch/kalman/
אתר זה מספק מדריך מפורט ומעמיק על מסנן קלמן, עם הסברים מתמטיים ואינטואיטיביים.

2. "Kalman Filter." Wikipedia.
https://en.wikipedia.org/wiki/Kalman_filter
ערך ויקיפדיה זה מספק סקירה כללית על מסנן קלמן, כולל הרקע ההיסטורי, הניסוח המתמטי, ויישומים שונים.

3. Visually Explained: Kalman Filters
https://www.youtube.com/watch?v=IFeCIbljreY
סרטון הסבר עם דוגמאות וניסוחים מתמטיים ואינטואיציות.

4. "Kalman Filter Overview." MathWorks.
https://www.mathworks.com/help/control/ref/kalman.html
דף זה מסביר על מסנן קלמן ומדגים כיצד להשתמש בו באמצעות MATLAB, כולל דוגמאות קוד.