דו"ח התקדמות 1 – 23.07.2023 – למידת נתונים במערכות זמן אמת

המגישים:

דוד ליבר 213910144

מיכאל רודל 326184744

המשימה שלנו:

בהינתן מספר מפות עומק (מערך של x,y דו ממדיים בתמונה והמרחק שלהם מהplane של המצלמה) או עננים נקודות (מערך של x,y,z) (אנחנו מניחים שניתן לעבור מהצגה אחת לשנייה אם יש צורך) אנחנו רוצים אלגוריתם מהיר שמחבר אותם לענן יחיד שמכיל את כל הנקודות, זה כולל מיזוג נקודות משני העננים שמסמנות את אותה נקודה פיזית, ומחיקת outliers.
זוהי בעיה יחסית פתורה בעולם הcomputer vision ואנחנו רוצים לפתור אותה ביעילות תחת ההנחות שלנו שידוע הסיבוב/תנועה בין הנקודות המבט שיצרו את העננים.

הסבר במילים שלנו על כל אחד מהשלבים:

1. המסלול הוא סיבוב של 360 מעלות בגובה מסוים, ואז עלייה אנכית כלפי מעלה, וסיבוב נוסף של 360 מעלות. אז, מה שמקבלים, זה שאם מסתכלים על הסיבוב התחתון, בין כל פריים לפריים, כלומר בין כל מספר קבוע של מעלות, מקבלים motion vectors. מקבלים את הווקטורים האלה הן בסיבוב התחתון והן בסיבוב העליון. הצילום התחתון והעליון נותנים לנו את האפשרות לעשות טריאנגולציה לפריימים שלנו, ובעזרת זה לקבוע עומק של נקודות.
אחת ההצעות שהוצעו, זה בעצם מתוך כל פריים למטה, והפריים המתאים למעלה, לחשב motion vectors ביניהם. הייתה הצעה נוספת, על מנת לחסוך חישוב motion vectors בין פריימים אנכיים מתאימים, שבמקום לעשות סיבוב שלם למטה, ואחר כך למעלה, מקודם לעשות עלייה אנכית, להסתובב במספר קבוע של מעלות, אחר כך לרדת אנכית, לעשות עוד סיבוב קטן, ואז עוד עלייה אנכית, וכן הלאה עד שמשלימים סיבוב מלא – וככה נקבל בחינם את ה-motion vectors בין הלמעלה ללמטה.

2. אלגוריתם הORB:
אלגוריתם ה-ORB (Oriented FAST and Rotate BRIEF) הוא אלגוריתם פופולארי בתחום הראייה הממוחשבת, המשמש לזיהוי תכונות (פיצ'רים), תיאור, והתאמה בין תמונות. הוא הוצג כחלופה יעילה לאלגוריתמי SIFT ו-SURF בשנת 2011 על ידי
Ethan Rublee ,Vincent Rabaud ,Kurt Konolige ,Gary Bradski.

אלגוריתם ה-ORB מורכב מ-2 רכיבים עיקריים:
זיהוי פיצ'רים (פינות, צמתים, שינויי צבע, וכו'), ע"י אלגוריתם ה-FAST (Features from Accelerated Segment Test), ותיאורים ע"י אלגוריתם ה-BRIEF (Binary Robust Independent Elementary Features).

1.1. FAST:
זוהי שיטת זיהוי נקודות עניין (קרי פיצ'רים) בתמונה, כגון פינות, צמתים, וכדומה. אלגוריתם זה בודק מעגל של 16 פיקסלים ב-"שכונה" של פיקסל,

ומשווה את הערך (העוצמה) של הפיקסל המרכזי אליהם. אם למספר מספיק של פיקסלים רציפים יש עוצמות גבוהות או נמוכות יותר מהפיקסל המרכזי, הוא ייחשב לפינה.

אחרי שמצאנו את הפיצ'רים בעזרת FAST, נתאר אותם בעזרת BRIEF. נרצה שהתיאור יהיה עמיד לשינויי תאורה, סיבוב, וקנה מידה (scale).

2.2. <u>BRIEF:</u>
BRIEF יוצר מחרוזת בינארית המקודדת את הקשר בין זוגות פיקסלים ב-"שכונה" של פיצ'ר (קרי נקודת עניין). כדי ליצור את תיאור ה-BRIEF מוגדרת קבוצה של בדיקות בינאריות אקראיות. עבור כל נקודת מפתח, זוגות פיקסלים נדגמים מאזור מוגדר מראש סביב מיקום נקודת המפתח. הבדיקה משווה את העוצמה של פיקסל אחד לשני, והתוצאה מקודדת כ- 1 אם יחס העוצמה עונה על קריטריון מסוים ו- 0 אחרת.

מגבלה אחת של BRIEF היא שהוא רגיש לסיבוב תמונה. כדי להתמודד עם מגבלה זו, ORB מציג שלב נוסף שמחשב כיוון עבור כל נקודת עניין. לאחר מכן מסובבים את מבחני ה-BRIEF בהתאם לכיוון של נקודת העניין, מה שהופך את המתאר לעמיד יותר לסיבובים.

<u>אז בסה"כ, נקבל שזה מה שעושה ORB:</u>

א. <u>זיהוי מהיר של נקודת עניין (פי'צרים):</u>
זהה נקודות מפתח פוטנציאליות בתמונה באמצעות אלגוריתם FAST.

ב. <u>הקצאת כיוון:</u>
לכל נקודת עניין, חשב כיוון המייצג את הכיוון הדומיננטי של תכונות תמונה מקומיות.

ג. <u>יצירת תיאורי BRIEF:</u>
צור תיאורים בינאריים עבור נקודות העניין (הפיצ'רים) באמצעות אלגוריתם BRIEF עם בדיקות מסובבות.

ד. <u>התאמת נקודות עניין (פיצ'רים):</u>
השווה בין התיאורים הבינאריים של נקודות מפתח בתמונות שונות כדי למצוא התאמות.

ORB ידוע וביעילותו ביישומים בזמן אמת בשל אופיו הבינארי, המפחית את השימוש בזיכרון ומאיצה את חישוב ההתאמה. הוא מספק איזון טוב בין דיוק למהירות, מה שהופך אותו לבחירה פופולרית במשימות ראייה ממוחשבת שונות, כגון תפירת (הדבקת) תמונה, זיהוי אובייקטים, וחישוב מרחק חזותי.

<u>motion vectors:</u>
motion vector מחושב ע"י מציאת התאמה בין חלון (אצלנו החלונות הם בגודל 16 X 16), בזמן t, וחלון בזמן t+1, כאשר t הוא האינדקס של פריים בווידאו. כלומר:

$$\vec{v} = \arg\min \|I(x,y,t) - I(x - v_1, y - v_2, t-1)\|$$

במקרה שלנו, שבו הפריימים מצולמים ממצלמת raspberry pi, ה-motion vectors מחושבים חומרתית – כלומר נתונים לנו (בין כל 2 פריימים סמוכים).

3. את קואורדינטות ה-(x,y) אנו יודעים מהמיקום של הרחפן, ואת קואורדינטת ה-z אנחנו יכולים לחשב מכיוון שיש לנו 2 צילומים מגבהים שונים (בעזרת טריאנגולציה). כך נבנה ענן הנקודות שלנו לכל פריים (והשותף שלו מלמעלה).

לכל נקודה בפריים ברחפן העליון, נמצא אותה באותו פריים התחתון, נחסר ביניהן, וכך נקבל את ה-disparity map (נציג את ה-disparity map כ-intensity map, כך שככל שהמרחק קטם יותר, הפיקסל יותר בהיר).

4. אחרי שיש לנו את כל העננים (עבור כל פריים משותף (למעלה ולמטה יש ענן נקודות)), נרצה לחבר אותם, כדי לקבל ענן שמתאר את כל המרחב שבו הופעל הרחפן.
כמה הצעות לפתרון וכמה בעיות:
   א. אם נסכום את כל העננים, אז נקבל סה"כ ענן צפוף מאוד ועם שכפולים מיותרים, כך שלא נוכל להבחין באובייקטים.
   ב. יש "רעשים" שצריך להשלים: לדוגמה, ייתכן שבפריים מסוים זוהה חפץ כלשהו (נניח שולחן), ובפריים הבא הוא לא זוהה כי המצלמה כבר הייתה באמצע של השולחן אז לא זוהה הבדל בפיקסלים.
   ג. יש רעשים שצריך לנקות: אם זיהינו פינה בפריים מסוים, אבל היא סתם פינה שלא תקדם אותנו למשהו (נגיד שינוי צורה של קיר) – צריך שלא נזהה את זה כפינה (פיצ'ר) – אופציה שהוצעה זה לקחת את החציון של הפיקסלים, אבל זה לא טוב כי כנראה בתמונה יש 2 אזורים כך שהחציון לא ייתן לנו משהו טוב, אז אולי צריך לקחת 10% מהפיקסלים (שבהם הערכים (ההתאמות) הכי גבוהים).
   ד. אופציה נוספת, היא לבנות סכמת voting, המבוססת על "מכירת חיסול", בצורה של עץ. לדוגמה:

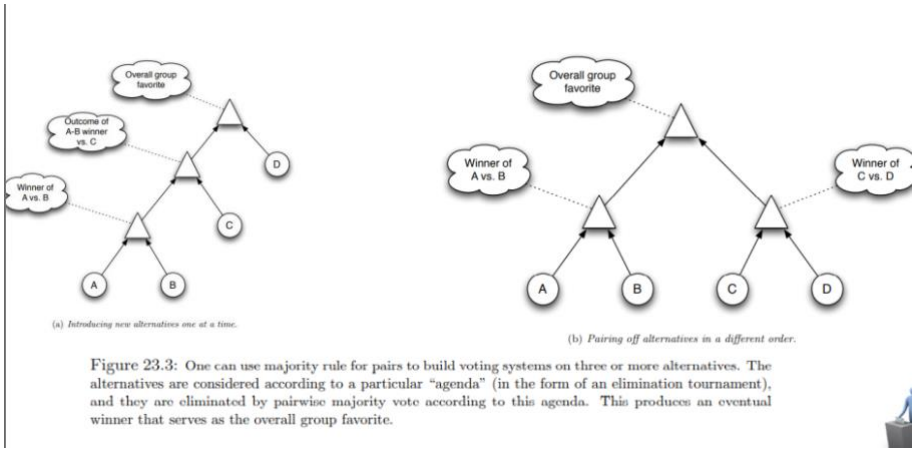

Figure 23.3: One can use majority rule for pairs to build voting systems on three or more alternatives. The alternatives are considered according to a particular "agenda" (in the form of an elimination tournament), and they are eliminated by pairwise majority vote according to this agenda. This produces an eventual winner that serves as the overall group favorite.

כאן אפשרי לבחור בין כל כמה פיקסלים "חשודים/בעייתיים", בין כל שני פריימים. הבעיה היא שמערכת בחירות כזאת היא לא טרנזיטיבית בהכרח (עם זאת, כאשר מגיעים לשורש, ניתן לעשות תיקון כלשהו). דהיינו, במקרה שלנו אם פעם אחת הרחפן יתחיל מזווית אחת, ופעם שנייה מזווית אחרת, ייתכן שנקבל ערך שונה לפיקסל ה-"בעייתי".