



המעבדה לעיבוד גיאומטרי של תמונות
Geometric Image Processing Laboratory

ExpressionMime

עורכי הדו"ח:

דור גרנט

רן יחזקאל

תאריך מסירה:

8.11.17

שם המנחה:

מתן סלע

תוכן עניינים

3	מבוא ותקציר.....
4	תיאור המערכת.....
5	תיאור הביצוע.....
9	תוצאות.....
14	מסקנות.....
15	המלצות.....
16	מקורות ספרותיים.....
17	נספחים.....

מבוא ותקציר

מטרת הפרוייקט שלנו הייתה נסיון ליצור מערכת שמסוגלת לזהות פרצופים והבעות פנים בוידאו בזמן-אמת ולהחליף את הפרצוף המקורי מהוידאו בפרצוף ממודל תלת מימדי בצורה שתתאים הכי טוב להבעות הפנים וסיבוב הראש. (ניתן גם לבצע את אותה הפעולה על כמה פרצופים יחדיו, ולא רק על פרצוף יחיד). מסריקה ספרותית שהתבצעה עם מתן, הגענו למאמר שמדבר על הנושא הזה ומספק את כל היסודות התיאורטיים הדרושים בנושא של זיהוי הבעות ותנוחה של פרצופים בזמן אמת. בפרוייקט השגנו בהצלחה רבה את נושא זיהוי הפרצופים והתנוחות של הפרצוף, והצלחנו לעוות מודל תלת מימדי של פרצוף להבעות שונות ואף לרנדור של הפרצוף בהבעות השונות בוידאו. זיהוי הבעות הפנים התגלה כבעיה קשה יותר ולכן הדפסת ההבעה הנכונה לא התבצעה כנכונה. המאמר המדובר מספק בסיס מתמטי רב ואף שיטות מימוש שונות עבור הנאמר בו ולכן ראוי לנסות ולהשתמש במידע המתמטי שבו ולהמירו לתיאוריה המתמטית שמתאימה עבור האפליקציה הזו עוד לפני תחילת העבודה הרטובה.

תיאור המערכת

המערכת כוללת תוכנית מחשב אשר מציגה על המסך את מה שרואה מצלמת הרשת של המחשב בזמן אמת, ומציגה מעל צילום הוידאו הזה את התוצאות בזמן אמת. התוכנה עושה שימוש בספריות החיצוניות dlib ו-OpenCV שצריכות להיות מותקנות גם כן בשביל שהמערכת תעבוד.

תיאור הביצוע

ראשית, התכנית מקבלת קובץ ply אשר מכיל בתוכו את המודל התלת מימדי המתאים. לתכנית כתבנו קורא קבצי ply אשר בהתחלה קרא אך ורק מיקומים של נקודות ואת המשולשים המורכבים מהם, ולאחר מכן שוכלל כך שיוכל גם לקרוא את הצבעים המתאימים לכל נקודה (מה שייצור בסוף את הטקסטורה בזמן הרנדור). הקובץ נקרא לתוך מבנה Mesh מסויים בתכנית אשר שומר את הנתונים כך שיהיו קלים לקריאה. כמו כן, התכנית קוראת את מילון ההבעות שמתאים ל-Mesh שנקרא.

מרגע זה מתחילה המצלמה לעבוד והתכנית מציגה את הפרטים המצולמים על המסך. כבר ברגע זה עלינו להתחיל בעבודת זיהוי הפרצופים בוידאו, כך שנזהה את הפרצופים בכל frame בנפרד. זיהוי הפרצופים נעשה על ידי ה-frontal_face_detector של ספריית dlib. ראשית היה נסיון לזהות פרצופים בכל frame, אך זה לא התאים לדרישות הביצועים שרצינו, שהרי אנחנו מנסים להשיג ביצועים ב-real-time למערכת שלנו. לכן החלטנו לזהות את הפרצופים בוידאו רק פעם בשני פריימים. כמו כן, בשביל לגרום למערכת זיהוי הפנים לעבוד בצורה מהירה יותר – שלחנו לאלגוריתם הזיהוי את הפריים ברזולוציה של חצי על חצי מהרזולוציה המקורית – וכך דאגנו לזיהוי פנים מהיר וכזה שניתן להסתמך עליו – מכיוון שתנועת פרצופים בוידאו היא לרוב רציפה ולכן השינוי במיקום הפרצופים בין שני פריימים עוקבים אינו גדול מדי.

כעת, עבור כל פרצוף – עלינו לצייר את המודל התלת מימדי בצורה נכונה. בשלבים הראשונים של התכנית רצינו לבדוק שאכן זיהוי הפנים עובד, ולכן הדפסנו מלבנים מסביב לפרצופים. כאשר השלב הזה עבד – עברנו לחלק המרכזי – שינוי המודל התלת מימדי כך שיתאים לתנוחת הפנים.

בשביל לבצע את סיבוב המודל בצורה נכונה ורינדור שלו במישור הדו מימדי של המצלמה ביצענו מספר דברים:

1. מציאת 68 נקודות עוגן (landmark points) בפרצוף, המזהות על ידי dlib.
2. זיהוי אותן הנקודות במודל התלת מימדי (hard coded עבור המודל איתו עבדנו)
3. כעת – עלינו להתאים את סיבוב המודל והגדלה / הקטנה שלו על מנת ש-68 הנקודות יתאימו זו לזו בצורה הטובה ככל הניתן. הדבר נעשה על ידי מציאת מטריצת הטלה מתאימה, עם אלגוריתם של OpenCV אשר יינתן עליו הסבר מאוחר יותר.
4. ברגע זה ניתן לסובב את המודל בדיוק כפי שאנחנו רוצים ולהעביר אותו למישור המצלמה.

5. כעת יש לנו נקודות בדו – מימד אשר ניתן להדפיס למסך ולקבל תוצאה כמתבקש. גם זה נעשה במספר שיטות במהלך הפרוייקט ועליהן נרחיב בהמשך.

שלב (1) נעשה על ידי שימוש במודל למידה מאומן של dlib עבור full_object_detection.

ההתאמה בין הנקודות התלת מימדיות משלב (2) והנקודות הדו מימדיות בשלב (1) נעשה על ידי אלגוריתם איטרטיבי Levenberg-Marquardt optimization. למעשה – באופן איטרטיבי יש נסיון למצוא מטריצת הטלה (אשר מורכבת מהרכבה יחדיו של מטריצת רוטציה ומטריצה טרנזלציה) כך שהפרשי הנקודות שמתקבלים מהנקודות הנתונות בדו מימד קטנות ככל שאפשר (לפי MSE). האלגוריתם שאנחנו משתמשים בו עובד על פי סכמת ה-RANSAC ולכן הוא מאפשר לקחת נקודות אשר נמצאות במקומות יוצאי דופן, שלכאורה לא אמורים להתאים ופשוט להתעלם מהם. כך אם אנחנו מוצאים מספיק נקודות, אשר ההפרשים ביניהן לנקודות מהוידאו קטנים מספיק – מצאנו מטריצת הטלה טובה מספיק ונוכל להמשיך. יש לשים לב שמכיוון שהאלגוריתם הוא איטרטיבי אז יש לו תנאי התחלה מסויימים. עבור הפריים הראשון – תנאי ההתחלה הוא מטריצת 0 עבור מטריצת ההטלה. מכיוון שהתנועה של פרצופים בוידאו היא די חלקה – ניתן להשתמש במטריצת ההטלה מהפריים הקודם כתנאי ההתחלה של האלגוריתם שלנו – ובכך לשפר את התוצאות והביצועים שלנו גם יחד. כמו כן, אנחנו מגבילים את מספר האיטרציות למספר סביר שבו הביצועים נשמרים גם אם הבעיה נתקלת במקרה קשה לפתרון.

מה היא בכלל המטריצה הזאת? איך אנחנו משתמשים בה ולמה זה עוזר לנו? מטרתנו היא לייצג את הנקודות התלת מימדיות הללו בדו מימד, ולכן נעשית הטרנספורמציה שומרת הפרספקטיבה הבאה:

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & r_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

כאשר X,Y,Z הן הקואורדינטות של נקודה במרחב התלת מימדי

u,v הן הקואורדינטות במישור הדו מימדי של המצלמה.

המטריצה 3x3 מייצגת את נתוני המצלמה כך ש:

c_x, c_y הן ה-principal points ולרוב (וכך גם הנחנו) מיוצגות על ידי מרכז המצלמה.

f_x, f_y הן אורכי העדשות ומיוצגות כאורך התמונה בפיקסלים.

s מייצג את הסקאלה שבה יש לחלק את הנקודות בשביל לקבל את הנקודות הרצויות.

כעת – מטריצת ה- 4×3 שנוטרה היא מטריצת ההטלה שאנחנו מחפשים, והיא מורכבת ממטריצת רוטציה (3×3 השמאלי) ומטריצת טראנזלציה (העמודה הימנית). למעשה, עבור כל נקודה אנחנו נרצה שיתרחש כי היא תוכפל במטריצת הרוטציה, ואז אליה תתווסף הטרנזלאציה. למעשה – הדבר מסובב את כל הנקודות באופן הנדרש ואז מזיז אותן במרחב על מנת להגיע למיקום המתאים. לאחר מכן הסקאלה נגזרת על ידי חילוק התוצאות כך ש- $z=1$ וכך מקבלים u, v מתאימים.

זו המטריצה שאנחנו מוצאים ואז מכפילים בה ובמטריצת הנתונים של המצלמה ומקבלים את הקואורדינטות המבוקשות – במישור העדשה. אלו למעשה הקואורדינטות שעליהן יש להדפיס בפריים.

בשלב ראשוני יותר בפרוייקט – החלטנו להסתפק בהדפסה פשוטה של הנקודות בשביל לראות שאכן ההטלה עובדת כראוי ועם ביצועים מדוייקים ומהירים, ולכן ה"רינדור" שלנו היה אך ורק הדפסה של כל הנקודות לאחר העברתן בצורה מתאימה למישור העדשה, בצבע קבוע.

כאשר ראינו שהשלב הזה עבר בהצלחה, הוספנו את פרסור הטקסטורה מקבצי ה-ply כך שכל נקודה קיבלה צבע משלה. זה כמובן לא מספיק – עלינו לצבוע את כל המודל, גם את המקומות ללא הנקודות – ולכן יש לעבור לשיטת רינדור חכמה יותר.

שיטת הרינדור שהשתמשנו בה נקראת z-buffering. שיטת הרינדור דורשת, פרט למידע על הנקודות בדו-מימד שהדפסנו עד עכשיו, גם את המידע על נקודות התלת מימד המקוריות (ולאחר הסיבוב שביצענו, על מנת לדעת מה אנחנו "רואים" מנקודת המבט של המצלמה, ולהדפיס בהתאם), וכמובן – מידע על המשולשים של ה-Mesh.

עבור כל משולש ב-mesh, אנחנו מוצאים את קואורדינטת ה-z של הצנטרואיד של המשולש (מרכז הכובד של המשולש, נקודת מפגש התיכונים). זאת קואורדינטת ה-z שמייצגת את המשולש על מנת לגלות האם הוא נראה לנו מנקודת המבט שלנו או שמא הוא מוסתר על ידי משולש אחר. כעת, אנחנו מוצאים רק את המשולשים הנראים מהזווית שלנו על פי הקואורדינטות שלהן (כל נקודה במודל נמצאת בתוך משולש אחד לפחות – אנחנו נקבע איזה משולש אנחנו צריכים להציג עבור הנקודה הזו). מרגע זה – כל נקודה מקבלת את המשולש המתאים לה, והיא נצבעת בצבע המתאים למשולש (שמחושב כממוצע משוקלל של צבעי שלושת הקודקודים של המשולש עבור הנקודה). מרגע זה ואילך, כל נקודה בפרצוף נצבעת בדיוק בצבע שבו היא צריכה להצבע במודל, וקיבלנו רינדור נקי, מדוייק ויפה.

למעשה – בין (4) ל-(5), הייתה אמורה להיות נקודה נוספת והיא מציאת ההבעה המתאימה עבור הפרצוף של המודל התלת מימדי, עיוות המודל בהתאם, והדפסה כרגיל. הנסיון הזה לא צלח לנו, אך נתאר את הפעולות שביצענו על מנת לנסות ולהגיע לתוצאות.

עבור המשימה, היה ברשותינו מילון הבעות שמתאים למודל הפרצוף התלת מימדי שעבדנו עליו. המילון הכיל 84 הבעות, כך שחיבור בפורמט המתאים בין הבעה במילון לקואורדינטות המתאימות בתלת מימד, ייתנו לנו הבעה שונה של הפרצוף. הבעה שונה לכל אחד מ-האפשרויות שבמילון. המטרה היא כמובן לא להיות מוגבלים ל-84 הבעות הללו, אשר שונות מאוד זו מזו – ואנחנו נרצה דווקא לקבל איזשהו שילוב של כמה הבעות יחדיו בשביל לקבל את הפרצוף עם ההבעה הכי דומה לזו של הפרצוף במצלמה.

הדרך לעשות את זה היא מציאת וקטור מקדמים כך שהכפלה שלו במילון (שניהם מיוצגים כמטריצות) ייתן לנו נתונים שהוספה שלהם לנקודות המודל יגרום לקבלת הבעה דומה בצורה אופטימלית לפרצוף שבמצלמה.

במאמר שעליו הסתמכנו היו מספר נוסחאות מתמטיות בשביל לנסות ולמצוא את וקטור המקדמים הזה, אותן מימשנו, אך התוצאה אינה צפויה ופרט לעיוות טוטאלי של הפרצוף – הנוסחא לא פותרת את בעיית מציאת ההבעה האופטימלית, למרות ניסיונות שונים כמו נירמול וקטור המקדמים, שינוי מילון ההבעות כך שיכיל וקטורים אורתונורמליים – אף פתרון לא נמצא על מנת למצוא באמת את ההבעה הרצויה.

עם זאת – עבור המקרה המנוון שבו רק אחת מהכניסות בוקטור המקדמים היא 1, וכל השאר 0 – אנחנו מצליחים לעוות את המודל כנדרש ולרנדר את ההבעה הזו, מה שנותן לנו יכולת הדפסה של 85 הבעות שונות – אך אין לנו יכולת של זיהוי הבעה אופטימלית.

תוצאות

להלן מספר תמונות מהפעלת התכנית.

תמונה של פרצוף יחיד בפריים, בלי שום פריטים על הפנים, והסתכלות ישירה למצלמה.



תמונה עם הסתכלות למעלה



הסתכלות לצד



גם למעלה וגם לצד



פרצוף מוטה לצד (לא מסתכל לצד, רק מוטה)



הסתכלות לצד, מרחק גדול מהמצלמה



פרצוף עם הבעה



מסתכל לצד, הפעם עם משקפיים



פריים של שני פרצופים יחדיו



פריים של שני פרצופים יחדיו, הפעם עם הבעה שונה



מסקנות

לבעיית זיהוי הפנים וזיהוי תנוחת פרצוף קיימים פתרונות תיאורטיים רבים, אשר חלקם מודגמים במאמר שמוצג בסקירה הספרותית. הבסיס המתמטי הוא רחב ומתוצאות הפרוייקט ניתן לראות כי התיאוריה אכן ניתנת ליישום פרקטי, ואף ליישום איכותי ומהיר אשר רץ בזמן-אמת.

מהתמונות ניתן לראות כי התכנית אף עושה את עבודתה בצורה איכותית גם כאשר הפרצוף נמצא בזווית יחסית גדולה מהמצלמה. כמו כן, אובייקטים כמו משקפיים לא מפריעים כל כך לזיהוי והדפסת הפנים. גם כאשר מציגים מספר פרצופים יחדיו – התכנית פועלת בצורה איכותית, מהירה ומסוגלת להראות את כל הפיצ'רים שלה באותה המידה.

המלצות

בשביל שיפור פעולת המערכת והגעה לתוצאות איכותיות יותר, כדאי לנסות ולהוסיף את זיהוי ההבעה בזמן אמת. על מנת לעשות זאת יש לקרוא ולהבין לעומק את הרקע המתמטי והתיאורטי של נושא זיהוי ההבעה והתנוחה של פרצוף, כפי שמתואר במאמר שעליו הסתמכנו.

כמו כן, ניתן לנסות ולאמן מודל זיהוי landmark points חדש, ולא להשתמש בזה שמסופק על ידי dlib.

רעיון נוסף שאפשר לממש בשביל זיהוי ההבעות הוא שימוש בלמידה עמוקה על מנת לזהות את הבעת הפנים בצורה איכותית ומהירה. בעיה בזה היא כמובן מציאה של תמונות / סרטונים אשר בהם מתוייגים הבעות הפנים, דבר שדורש משאבים רבים.

מקורות ספרותיים

1. Camera Calibration and 3D Reconstruction. OpenCV Docs - https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html
2. High Fidelity Pose and Expression Normalization for Face Recognition in the Wild. Xiangyu Zhu, Zhen Lei, Junjie Yun, Dong Yi, Stan Z. Li. - http://www.cbsr.ia.ac.cn/users/xiangyuzhu/papers/CVPR2015_High-Fidelity.pdf, <http://www.cbsr.ia.ac.cn/users/xiangyuzhu/projects/HPEN/main.htm>
3. Z-buffering and Z-culling. Wikipedia - <https://en.wikipedia.org/wiki/Z-buffering#Z-culling>

נספחים

קישור ל-BitBucket עם קוד הפרוייקט -

[/https://bitbucket.org/faceandroid/faceandroid](https://bitbucket.org/faceandroid/faceandroid)

יש לשים לב שבעקבות בקשה של מתן – המילונים אשר משמשים לעיוות המודל ויצירת הבעות פנים לא נמצאים פה. כמו כן, יש להסתכל על הbranch בשם final עבור הקוד הסופי.