Autocalibration via Rank-Constrained Estimation of the Absolute Quadric

שם הקורס: נושאים מתקדמים בראייה ממוחשבת

סטודנטים: משה מימון, יעקב מישייב

מרצה: ד"ר ירמיהו קמינסקי

תקציר:

בעבודה התבקשנו לבצע אימפלמנטציה למאמר Autocalibration via Rank Constrained Estimation of the Absolute Quadric. במאמר מוצגת שיטה שמאפשרת לנו לקחת שיחזור פרויקטיבי ולשדרג אותו לשיחזור מטרי על ידי קירוב Absolute Dual Quadric. (בנוסף לקליברציה עצמה)

בהצגה זו נציג את יתרונות השיטה ביחס לשיטות אחרות , רקע מתמטי לאלגוריתם, את הקוד של האימפלמנטציה שלנו וסיכום העבודה.

רקע:

כפי שכבר הסברנו בשקף הקודם המאמר מציג אלגוריתם autocalibration. קליברציה , כפי שלמדנו בקורס , היא התהליך של חילוץ הפרמטרים הפנימיים של המצלמה אך בניגוד לקליברציה אוטו-קליברציה לא דורשת אוביקטי קליברציה ותהליכי preprocess דומים. מדובר בשיטה שמקבלת אוסף של תמונות לא מכוילות ומצליחה לחלץ את מטריצות הפרמטרים שלהם. כפי שלמדנו בקורס יש מספר שיטות לבצע כיול (תלוי במידע שנתון לך) אך שיטה נוספת לבצע זו היא על ידי שימוש בAbsolute Conic. אחת התכונות של חתך החרוט הזה שהוא מהווה invariant תחת תזוזות של "קשיחות" של המצלמה (טרנספורמציות אוקלידיות). החשיבות האמיתית שלו היא בAbsolute Dual Quadric שהיא בעצם הדואלית שלו ומקודדת בתוכו מידע לכיול.

הקוניט האבסולוטי מוגדר להיות:

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{pmatrix}$$

רקע - המשך:

נניח כי יש לנו שחזור פרויקטיבי , קרי אוסף של מצלמות פרויקטיביות שנסמן ב $\mathbf{P}^i$ עם "נקודות עניין" שנסמן על ידי $\mathbf{X}^i$. אם כן כדי לבצע את השידרוג נצטרך למצוא את הטרנספורמציה הפרויקטיבית **H** שתיקח אותנו לשיחזור אוקלידי על ידי:

$$\mathbf{P}_M^i = \mathbf{P}^i\mathbf{H}$$
$$\mathbf{X}_M^i = \mathbf{H}^{-1}\mathbf{X}^i$$

אחת הדרכים להגיע ל**H** היא עם ההנחה שמטריצת המצלמה הפרויקטיבית היא במרכז הצירים. אך פה נציג דרך נוספת לקבל את המטריצה הזו ביחד עם **K:** הקוניק האבסולוטי, הסיבה העיקרית שאנחנו מעוניינים בו היא כי התמונה הדואלית שלו מקיימת: $\omega^* = \mathbf{K}\mathbf{K}^T$ כלומר ברגע שחישבנו אותו (על ידי קירוב) כל מה שנשאר לעשות הוא לבצע פירוק צולסקי כדי לקבל את K!

The Absolute Dual Quadric

לפני שנמשיך נציין כמה תכונות חשובות של ADQ המסומן כ Q_∞ :

1. Q_∞ הוא קוניק מנוון; כלומר הוא סינגולרי מדרגה 3.
2. Q_∞ סימטרי וחיובי/שלילי סמי-דפיניט.
3. Q_∞ מתקבע תחת טרנספורמציות קונפורמליות.
4. הגרעין שלו הוא המישור באינסוף: π_∞

אוטוקליברציה בעזרת Absolute Quadric

למטריצת מצלמה $\mathbf{P}^i$, התמונה הדואלית של הקוניק האבסולוטי היא ההטלה תחת $\mathbf{P}^i$ של הקוניק האבסולוטי $\mathbf{Q}_\infty$, כלומר:

$$\omega^{*i} = \mathbf{P}^i \mathbf{Q}^*_\infty \mathbf{P}^{i^T}$$

כעת נוכל להגדיר הגבלות על הכניסות של ω^{*i} כדי לחלץ את ADQ.

דבר נוסף שמקודד בתוך $\mathbf{Q}_\infty$ הוא הטרנספורמציה האוקלידית שלנו! אכן לאחר שקירבנו את ADQ נוכל על ידי פירוק לערכים עצמים פשוט לשחזר את **H** על ידי:

$$\mathbf{Q}^*_\infty \sim \mathbf{E}\Lambda\mathbf{E}^T$$

אז נגדיר את הטרנספורמציה האוקלידית שלנו להיות: $T \sim \mathbf{E}\Lambda^{\frac{1}{2}}$

(עם החלפת הערך העצמי האפסי ב1) .

הגדרת הבעיה

בניגוד לאלגוריתמי קליברציה אחרים שמשתמשים בabsolute quadric , פה נקרב אותו ישירות אך עם התרומה העיקרית ,ביחס לאלגוריתמים אחרים, שנאלץ אותו להיות positive-semidefinite ולא מדרגה מלאה **בתוך תהליך האופטימיזציה**.

אם כן נניח כי יש לנו פונקצית יעד f , שנבחרה ותלויה בפרמטרים של $\mathbf{Q}_\infty$. אם כן הבעיה כמו שמוגדרת לעכשיו היא מהצורה:

$$\min_{\mathrm{rank}(\mathbf{Q}^*_\infty)<4,\mathbf{Q}^*_\infty\succeq 0} f(\mathbf{Q}^*_\infty)$$

נזכור שבגלל שQ סימטרית הבעיה היא בעלת 10 משתנים חופשיים. כעת נרצה להמיר את הבעיית אופטימיזציה הזו לבעייה עם אילוצים קונקרטים (קרי פונקציות) ויותר מזה עם אילוצים פולינומים.

הגדרת הבעיה - המשך

כדי להבטיח שהמטריצה תהיה מדרגה לא מלאה ניתן לדרוש שהדטרמיננטה שלה היא אפסית , כלומר בעצם לדרוש איפוס של פולינום מדרגה 4. כדי להבטיח את positive semi definiteness ניתן לבצע זאת על ידי קריטריון סילבסטר ולוודא שהמינורים הראשיים (או יותר נכון הדטרמיננטות שלהם) הם אי-שליליות. כלומר אוסף של פולינומים מדרגה 3 לכל היותר.

לכן הצלחנו להפוך את הבעיה ל:

$$\begin{aligned} \min \quad & f(\mathbf{Q}^*_\infty) \qquad (10) \\ \text{subject to} \quad & \det(\mathbf{Q}^*_\infty) = 0 \\ & \det\left[\mathbf{Q}^*_\infty\right]_{jk} \geq 0, \quad j = 1,2,3 \quad k = 1, \cdots, \binom{4}{j} \\ & \|\mathbf{Q}^*_\infty\|_F^2 = 1. \end{aligned}$$

(הערה: בגלל שהמטריצה היא מוגדרת על לכדי סקלר משתמשים באילוץ האחרון לשמר נורמה אחת)

הגדרת objective function

כרגע שהגדרנו את הבעיה באופן פורמלי עם האילוצים כל שנשאר לעשות הוא לבחור את f.

עם השנים לאורך המחקר השתמשו בפונקציות יעד שונות כאשר הtrade-off ביניהם הוא בין שימור של תכונות גיאומטריות בעלות משמעות ובין שימור אופטימליות של הפתרון.

תחת הנחות מסוימות (כמו שהskew הוא 0 וה aspect ratio קרוב ל1) ניתן לדרוש קשר אלגברי שמתקיים על ידי הכניסות של DIAC כך שK שלנו תהיה אלכסונית עם aspect ratio 1.
פונקציה יעד שתקיים זו היא:

$$f(\mathbf{Q}^*_\infty) := \sum_i (\omega^{*i}_{11} - \omega^{*i}_{22})^2 + {\omega^{*i}_{12}}^2 + {\omega^{*i}_{13}}^2 + {\omega^{*i}_{23}}^2 .$$

objective function - המשך

חשוב לציין שאין זו האופציה היחידה ובמאמר שלנו לבד מצויינות עוד שני פונקציות אלטרנטיבות ל-f שאציג פה:

$$f(\mathbf{Q}^*_\infty) := \sum_i \frac{(\boldsymbol{\omega}^{*i}_{11} - \boldsymbol{\omega}^{*i}_{22})^2 + \boldsymbol{\omega}^{*i}_{12}{}^2 + \boldsymbol{\omega}^{*i}_{13}{}^2 + \boldsymbol{\omega}^{*i}_{23}{}^2}{\boldsymbol{\omega}^{*i}_{33}{}^2}. \tag{12}$$

$$f(\mathbf{Q}^*_\infty) := \sum_i \|\boldsymbol{\omega}^* - \lambda_i \mathbf{P}^i \mathbf{Q}^*_\infty \mathbf{P}^{i\top}\|.$$

Implementation

לאחר שהגדרנו את בעיית האופטימיזציה והצגנו את כל הרקע הנדרש נראה כעת את המימוש בקוד. להלן מימושים של האילוצים השונים:

```
def constraint_rank_degenerate(upper_tri):
    # Return the determinant of Q.
    Q = transform_into_matrix(upper_tri)
    return np.linalg.det(Q)


def constraint_minor_det(upper_tri,i):
    # Return the determinant of the i,j minor of Q.
    Q = transform_into_matrix(upper_tri)
    minor = matrix_minor(Q,i)
    return np.linalg.det(minor)


def matrix_norm(upper_tri):
    # Return the Frobenius norm of Q
    Q = transform_into_matrix(upper_tri)
    return np.norm(Q)
```

מימוש בעיית האופטימיזציה:

```
def minimize_dual_quadric(cameras):
    """ Function estimates Q_infinity given the cameras matrices."""
    x0 = np.zeros((1,10))
    # Minimize function is set default such that constraint is equal to 0 and ineq means >= 0.
    constarints = ({'type':'eq',   'fun':lambda q: constraint_rank_degenerate(q)},
                   {'type':'eq',   'fun':lambda q: (matrix_norm(q)-1)},

                    # Constraints on the principal minors of the matrix to guarantee positive-definite.
                    {'type':'ineq', 'fun':lambda q: constraint_minor_det(q, 1)},
                    {'type':'ineq', 'fun':lambda q: constraint_minor_det(q, 2)},
                    {'type':'ineq', 'fun':lambda q: constraint_minor_det(q, 3)},
                    {'type':'ineq', 'fun':lambda q: constraint_minor_det(q, 4)})
    f = lambda q: objective_function(q, cameras)
    res = minimize(f,x0,method = 'Nelder-Mead')
```

המשך הפונקציה

```
if res.success:
    Q = transform_into_matrix(res.x)
    ws = np.array([camera.dot(Q).dot(camera.T) for camera in cameras])
    return Q, ws
else:
    print ('Optimization failed.')
    print(res.message)
    return None, None
```

מימוש הקליברציה ושידרוג לשחזור אוקלידי

```
def calculate_euclidean_homography(Q):
    # Calculate the Euclidean homography by the eigendecomposition of dual absolute conic.
    sigma, v = np.linalg.eig(Q)
    return v.dot(np.sqrt(sigma))

def calculate_Ks(ws):
    Ks = []
    for w in ws:
        Ks.append(np.linalg.cholesky(w))
    return ws
```