УДК 501

ЗАДАЧА СТЕРЕО-ЗОРУ В УМОВАХ ПОСТУПОВОГО НАДХОДЖЕННЯ ДАНИХ

Кириленко І. А.[1, a]

[1] *Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Фізико-технічний інститут*

Анотація

Ключові слова: комп'ютерний зір, стерео-зір, динамічне програмування

Вступ

Від автономної навігації [1] та фотограмметрії citestereoPG до доповненої реальності citestereoAR – задачі стерео-зору мають багато застосувань. Задача стерео-зору полягає у використанні двох або більше камер для отримання даних про відстань до об'єктів citestereoReview. Ідея методу полягає в порівняні зображень об'єкта отриманих під різними кутами та пошуку скалярного поля зсувів (або карти зсувів – від англ. disparity map).

В цій роботі розглядаються одновимірні алгоритми пошуку карти зсувів. Це означає, що кожен рядок зображення ми опрацьовуємо незалежно від інших. Такий підхід вимагає попередньої ректифікації зображення, проте є більш швидким та дає непогані результати (рис. 1).

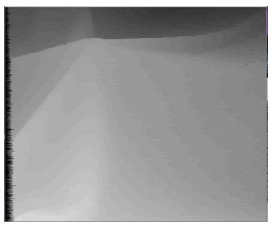


Рис. 1. Результат роботи одновимірного алгоритму

Далі під «зображенням» будемо розуміти деякий рядок вихідного зображення. Перший розділ присвячено постановці задачі стерео-зору та методам її вирішення в умовах повних даних. У другому розділі розглянуто модифікації цих методів для випадку поступового надходження даних.

1. Пошук зсувів при повних даних

1.1. Постановка задачі

Нехай n – довжина зображення, а $I = \{1, 2, .., n\}$ – множина координат пікселів. Ліве і праве зображення задамо як функції $\mathcal{L} : I \to R$ та $\mathcal{R} : I \to R$. Тобто $\mathcal{L}(i)$ – інтенсивність i-го пікселя на лівому зображені, а $\mathcal{R}(i)$ – інтенсивність i-го пікселя на правому зображені. Така постановка природно узагальнюється на випадок кольорових зображень. Також введемо множину зсувів $D = \{0, ..., D_{max}\}$, де D_{max} – значення максимального зсуву, та підбирається експериментально.

Для вирішення задачі нам потрібно кожному пікселю лівого зображення знайти відповідний йому піксель на правому зображенні (2). Тобто для кожного пікселя з номером i лівого зображення знайти таке $d_i \in D$, щоб піксель правого зображення з номером $i - d_i$ – відповідав йому.

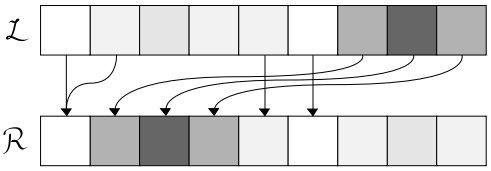


Рис. 2. Пошук відповідних пікселів

Проте, не будь яка пара пікселей може знаходитися у відповідності – пікселю з номером i на лівому зображені можуть відповідати тільки ті пікселі правого зображення з номером j, для яких $i \leq j$. Переконатися в цьому можна тримаючи перед собою олівець та по черзі закриваючи то праве, то ліве око. Для правого ока олівець буде знаходитися лівіше ніж для лівого

Тож треба знайти таку послідовність $\overline{d} \in D^n$, яка б мінімізувала штрафну функцію

$$\omega(\overline{d}) = \sum_{i=1}^{n} h(i, d_i) + \sum_{i=1}^{n-1} g(d_i, d_{i+1}), \qquad (1)$$

$$h(i, d_i) = \mid \mathcal{L}(i) - \mathcal{R}(i - d_i) \mid,$$

$$g(d_i, d_{i+1}) = \alpha \mid d_i - d_{i+1} \mid.$$

Де $h(i, d_i)$ відповідає за схожість кольору пікселів, $g(d_i, d_{i+1})$ – за гладкість поля зсувів, а α – коефіцієнт згладжування.

[a] mr_thor@gmail.com

1.2. Зведення задачі до пошуку найкоротшого шляху на графі

Представимо задачу пошуку послідовності $\overline{d}$, що мінімізує штрафну функцію (1), як пошук найкоротшого шляху через орієнтований зважений граф $G =< \mathcal{V}, \mathcal{E} >$ (3). Множина вершин якого

$$\mathcal{V} = \{\sigma(i,d) \,|\, i \in I,\ d \in D\} \cup \{S, E\}.$$

Вершина $\sigma(i,d)$ має вагу $h(i,d)$, $i \in I$, $d \in D$. Множина ребер

$$\mathcal{E} = \bigcup_{d \in D} < S, \sigma(1,d) > \ \cup$$
$$\cup \bigcup_{d \in D} < \sigma(n,d), E > \ \cup$$
$$\cup \bigcup_{\substack{i=1..n-1 \\ d \in D \\ d' \in D}} < \sigma(i,d), \sigma(i+1,d') > .$$

Ваги ребер

- З S в $\sigma(1,d)$ – 0, $d \in D$
- З $\sigma(n,d)$ в E – 0, $d \in D$
- З $\sigma(i,d)$ в $\sigma(i+1,d')$ – $g(d,d')$, $d, d' \in D, i \in I$

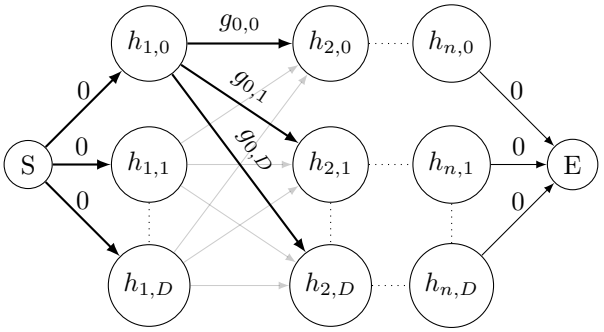


Рис. 3. Граф G

Тоді послідовність $\overline{d}$ – послідовність вершин через які проходить найкоротший шлях з S в E, і буде мінімізувати штрафну функцію $\omega(\overline{d})$ (1).

Позначимо довжину найкоротшого шляху з вершини S в вершину $\sigma(i,d)$ як $f_i(d)$. Тоді $\forall d \in D$

$$f_1(d) = h(1,d),$$
$$f_2(d) = \min_{d' \in D} \Big(f_1(d) + g(d',d) \Big) + h(2,d),$$
$$\vdots$$
$$f_i(d) = \min_{d' \in D} \Big(f_{i-1}(d) + g(d',d) \Big) + h(i,d).$$

Тоді елементи послідовності $\overline{d}$ знаходимо за формулами:

$$d_n = \arg\min_{d' \in D} \big(f_n(d') \big),$$
$$d_i = \arg\min_{d' \in D} \big(f_i(d') + g(d', d_{i+1}) \big) \ i = \overline{n-1, 1}$$

2. Пошук зсувів при поступово надходжуючих даних

Метод описаний в частині 1 потребує наявності всіх даних до початку обчислень. В ситуації, коли дані надходять повільно, цей метод не є самим ефективними. Доводиться чекати надходження всіх даних і тільки потім починати обчислення, адже ми не можемо розрахувати ваги деяких вершин. Замість цього можна проводити деякі розрахунки над частиною даних, яка вже надійшла, цим самим зменшивши час роботи алгоритму після отримання всіх даних.

2.1. Випадок 1 – дані надходять по порядку

Нехай на початку нам невідомі жодні пікселі (рис. 4), тож ми не можемо обчислити вагу жодної з вершин графа G (1.2).

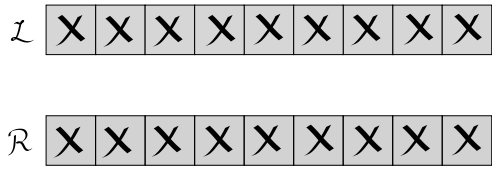


Рис. 4. Немає даних

Будемо називати вершину «закритою» якщо ми не можемо обчислити вагу вершини, а якщо можемо – назвемо її «відкритою». Тож на початку всі вершин графу G закриті.

Нехай тепер нам поступово, по порядку надходить по одному пікселю кожного зображення. Коли нам відомий лише перший піксель кожного зображення (рис.5) то граф G матиме лише одну від-

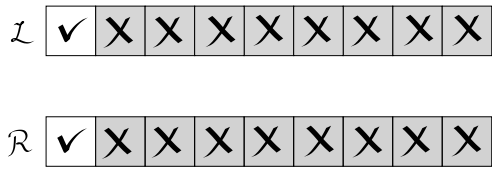


Рис. 5. Відомі перші пікселі обох зображень

криту вершину $\sigma(1,0)$. Коли нам буде відомо два перші пікселі, то нам стануть відомі ще й вершини $\sigma(1,1)$ та $\sigma(2,0)$. Таким чином, при надходженні перших k пікселів обох зображень, в графі G будуть відкриті вершини $\sigma(i,d_i)$, де $i \in I, i \le k$, а $d_i \in D, 0 \le d_i \le \min\big(k-i, D_{max}\big)$. І тільки коли k буде більше за D_{max}, всі вершини в першому стовпчику $\sigma(1,d)$ $\forall d \in D$ будуть відкритими, і ми зможемо починати оптимізацію:

Нехай $G^0 =< \mathcal{V}^0, \mathcal{E}^0 >=< \mathcal{V}, \mathcal{E} >= G$. Нехай також $s^0(d) = 0$, $\forall d \in D$. Для відновлення послідовності $\overline{d}$ введемо матрицю $\hat{\mathcal{P}}$ розмірності $n \times (D_{max}+1)$.

$$p_{1,d} = 0, \forall d \in D.$$

Як тільки перші $D_{max} + t$, $(1 \le t < n - D_{max})$ стерео-пікселей стають нам відомі, шукаємо граф G^t:

1) $\forall d \in D$:
$s^t(d) = \min_{d' \in D} \big(s^{(t-1)}(d') + h(t,d') + g(d',d) \big)$.
$p_{t+1,d} = \arg\min_{d' \in D} \big(s^t(d) \big)$.
2) $\mathcal{V}^t = \mathcal{V}^{t-1} \setminus \{\sigma(t,d) \mid d \in D\}$.

3) $\mathcal{E}^t = \mathcal{E}^{t-1} \setminus$
$\setminus \bigcup\limits_{d \in D} < S, \sigma(t,d) > \setminus$
$\setminus \bigcup\limits_{\substack{d \ in D \\ d' \in D}} < \sigma(t,d), \sigma(t+1,d') > \cup$
$\cup \bigcup\limits_{d \in D} < S, \sigma(t+1,d) > .$
А вага ребра $< S, \sigma(t+1,d) >$ – $s^t(d)$
4) $G^t =< \mathcal{V}^t, \mathcal{E}^t >$

Кожен новий граф G^t матиме на $(D_{max})^2$ менше ребер, ніж граф G^{t-1}. Таким чином, на момент приходу останніх пікселів, нам треба буде опрацювати лише D_{max} ребер. А якщо б ми спочатку чекали приходу всіх даних, а тільки потім починали обчислення, нам треба було опрацювати $(n-1)(D_{max})^2$ ребер.

Коли ж нам стануть відомі всі n пікселей обох зображень, нам залишиться тільки знайти

$$d_n = \arg\min_{d' \in D} \left(s^{(n-D_{max}-1)}(d') + h(n,d') \right),$$

та відновити послідовність $\overline{d}$ через матрицю $\hat{\mathcal{P}}$ –

$$d_i = p_{i+1,d_{i+1}}, i = \overline{n-1,1}.$$

2.2. Випадок 2 – повністю відоме одне з зображень

Нехай на початку нам відоме лише ліве зображення (рис. 6), у такому разі ми теж не можемо обчислити вагу жодної з вершин графа G. Будемо використовувати ті ж позначення як і в 2.1. Тоді граф G буде мати вигляд як і на (рис. 4).

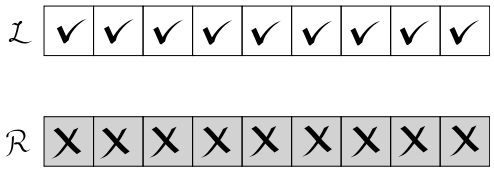


Рис. 6. Відомо лише одне з зображень

Нехай тепер нам стає відомий якийсь один (наприклад другий) піксель правого зображення (рис. 7).

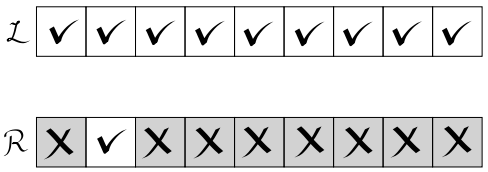


Рис. 7. Відомий другий піксель правого зображення

Тоді у графі G одразу всі вершин у другому стовпчику стануть відкритими ().

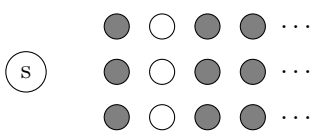


Рис. 8. граф G при відсутності даних

При такій конфігурації можлива оптимізація. Припустимо, що найкоротший шлях проходить через вершини ${h^{(1)}}_i$ та ${h^{(3)}}_j$ (рис. 9). Тоді ми точно знаємо що найкоротший шлях проходить через вершину ${h^{(2)}}_k$, де

$$k = \arg\min_{l=0}^{D_{max}} \left(g_{il} + {h^{(2)}}_l + g_{lj} \right) \quad (2)$$

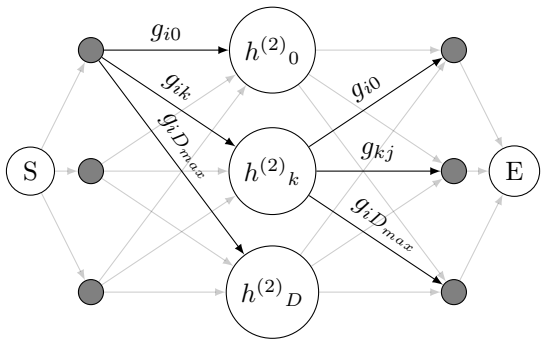


Рис. 9. Граф G до оптимізації

Та на практиці ми ще не знаємо через які вершини проходить найкоротший шлях, тому проводимо цю операцію для кожної пари вершин $< {h^{(1)}}_i, {h^{(3)}}_j > \forall i,j \in D$. Для кожної вершини ${h^{(3)}}_j$ запам'ятовуємо вершину ${h^{(2)}}_k$, де k визначається формулою (2). Після цього можемо замінити всі вершини у другому стовпчику та шляхи через них на ребра, що з'єднують вершину ${h^1}_i$ з ${h^3}_j$ $\forall i,j \in D$ (рис. 10). Ваги цих ребер

$$g'_{ij} = \min_{l=0}^{D_{max}} (g_{il} + h_l + g_{lj}) \; \forall i,j \in D.$$

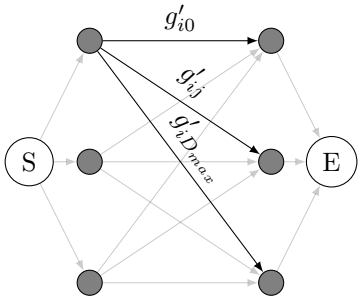


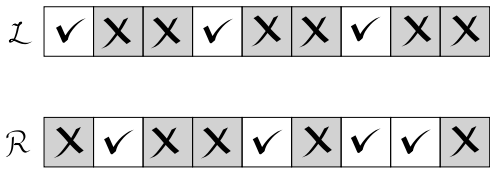
Рис. 10. Граф G після оптимізації

При надходженні наступного пікселя з номером m правого зображення, будемо проводити аналогічні операції: для кожної вершини ${h^{m+1}}_j$ запам'ятовувати , де

Така оптимізація дозволяє нам зменшувати кількість ребер в графі на $(\max D)^2$ при кожному новому відкритому пікселі другого зображення. Таким чином, коли нам відкриється останній невідомий піксель, наш граф матиме лише $2 \max D$ ребер.

2.3. Випадок 3 – дані надходять хаотично

Нехай дані надходять у випадковому порядку.



У такому випадку порядок відкриття вершин у графі нам невідомий. Припустмо що в нас відкрита одна вершина.

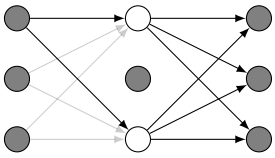


Рис. 11. До оптимізації

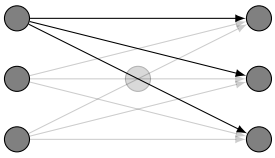


Рис. 12. Після оптимізації

Кількість ребер буде зменшуватися тільки тоді, коли відкрито більше половини вершин в стовпці.

3. Приклади цитування літератури

На кожне джерело має бути посилання в тексті тез. Приклад оформлення посилань на підручник або книгу [**?**]. Якщо посилання йде на конкретну сторінку в книзі, то її слід вказувати таким чином [**?**, стор. 4]. Якщо посилання йде книгу або монографію з кількома авторами, то посилання оформляються в такому вигляді [**?**]. Якщо посилання на кілька джерел, то треба їх перераховувати через кому [**?**, **?**].

Посилання на статтю в журналі [**?**, **?**].

Висновки

В цьому розділі узагальнюються головні підсумки обговорення результатів.

Перелік використаних джерел

1. Murray Don, Little J.J. Using Real-Time Stereo Vision for Mobile Robot Navigation // Auton. Robots. — 2000. — 04. — Vol. 8. — P. 161–171.