КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
МЕХАНІКО-МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
КАФЕДРА АЛГЕБРИ І КОМП'ЮТЕРНОЇ МАТЕМАТИКИ

Освітній ступінь: магістр

за спеціальністю 111 - математика
за освітніми програмами - математика

кваліфікаційна робота на ступінь магістра математики
на тему "Застосування машинного навчання у задачі знаходження
відповідностей для стерео пари"

студента 2 курсу магістратури
Швеця Максима Сергійовича

Допущений до захисту в ЕК
Протокол №11 засідання кафедри
алгебри і комп'ютерної математики
від 21 квітня 2020 року

Науковий керівник
Доктор фізико-математичний наук
Лавренюк Я. В.

Київ-2020

1 Огляд

В цій роботі розглянемо задачу стереобачення та метод її розв'зання за допомогою машинного навчання запропонований в статті "Efficient Deep Learning for Stereo Matching" [2]. Модифікуємо цей метод та перевіримо його ефективність на датасеті KITTI.

Задача стереобачення важлива в таких облатях як

2 Стереобачення

Задача стереобачення полягає в знаходженні інформації про тривимірні об'єкти використовуючи їх зображення. Зазвичай розглядають два зображення отримані з двох камер розташованих на одній горизонтальній осі, саме з такими данними працює алгоритм описаний в цій роботі.

Маючи два зображення однієї сцени можна зрозуміти наскільки далеко розташовані від камери зображені об'єкти спостерігаючи за зміною їх положення на зображеннях. Чим сильніше змінюється положення об'єкта між зображеннями, тим ближчий він до камери. Приклад цього можна побачити на рис. 1 взятому із датасету MIddlebury [3].

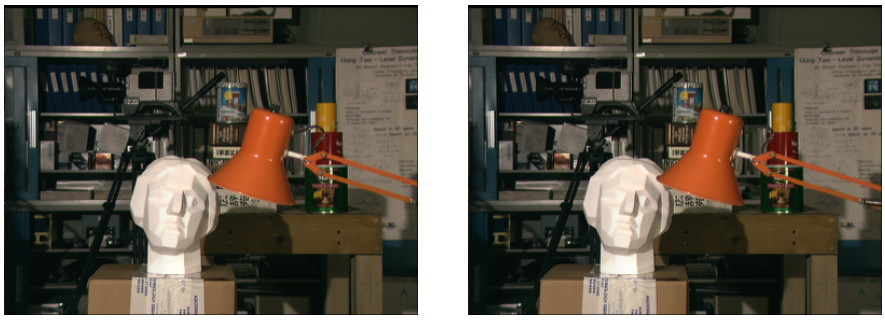


Рис. 1: Настільна лампа розташована ближче до камери і тому ссувається сильніше ніж, наприклад, голова чи стіл

Якщо для деякої точки сцени відомі відповідні їй точки на зображеннях, то ми можемо розрахувати глибину цієї точки за формулою (рис. 2).

$$h = \frac{fb}{d}$$

де f - це фокусна відстань, b - відстань між камерами, d - різниця між кординатами точок зображеннь. Також, можна розрахувати відстань від оптичної вісі камери за формулою

$$r = \frac{bx_r}{d}$$

де x_r - відстань точки від центра зображення

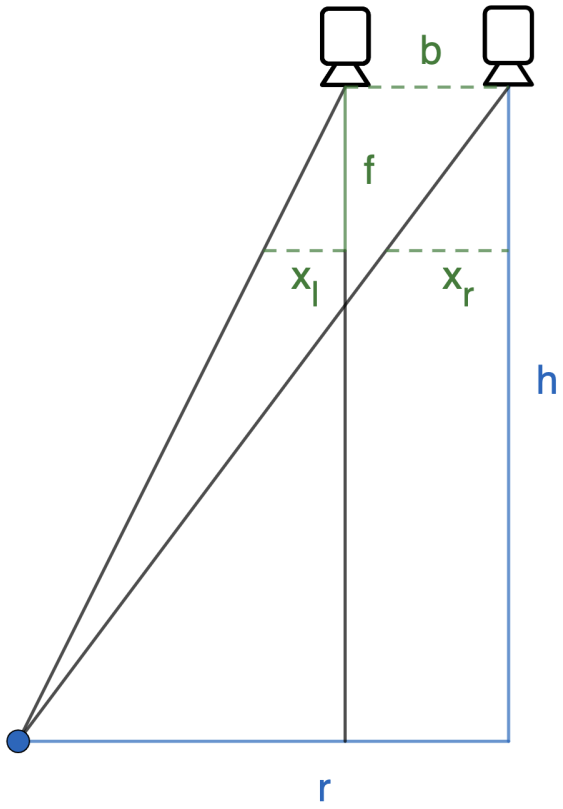


Рис. 2: Відомі нам значення (відстань між камерами b, фокусна відстань f, координати точок на зображеннях x_l, x_r позначено синім. Значення які можемо знайти (глибина зображення - h, відстань від вісі камери - r) позначено зеленим)

Отже, маючи алгоритм який співставляє точкам одного зображення відповідні їм точки іншого зображення ми можемо знаходити відстані до точок сцени. Зручний спосіб представити ці відстані - чорно-біле зображення в якому кожен піксель тим світліший чим більше значення d його зсуву на іншому зображенні. Це зображення називають **мапою зсувів** (англ. disparity map). За попередньої форму видно що зсув обернено-пропорційний глибині, тому об'єкти на такому зображені будуть тим світліші чим ближче вони до камери. Мапу зсувів для стереопари зображенної на рис. 1 можна побачити на рис. 3.

Наш алгоритм приймає два зображення і обчислює мапу зсувів.

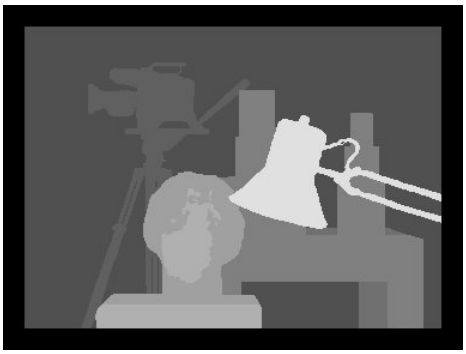
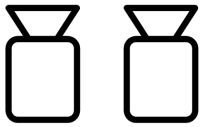


Рис. 3: Лампа знаходиться ближче до камери тому зсув пікселів на яких зображена лампа вищий, а отже на малюнку вони світліші.

2.1 Виправлення зображенньь

Знаходити відповідні пікселі легше за все коли камери зміщені лише горизонтально і направлені однаково. При такому налаштуванні пікселі будуть зсуватися між зображення лише по горизонталі, а отже ми можемо шукати відповідні пікселі лише в одному вимірі.



(а) Зручне розташування

(б) Незручне розташування

У випадках коли цих ідеальних умов неможливо досягти, до зображень можна застосувати процедуру виправлення (англ. rectification), яка полягає в проектуванні зображенньь на одну площину. Результат застосування такої процедури можна побачити на рис. 5.

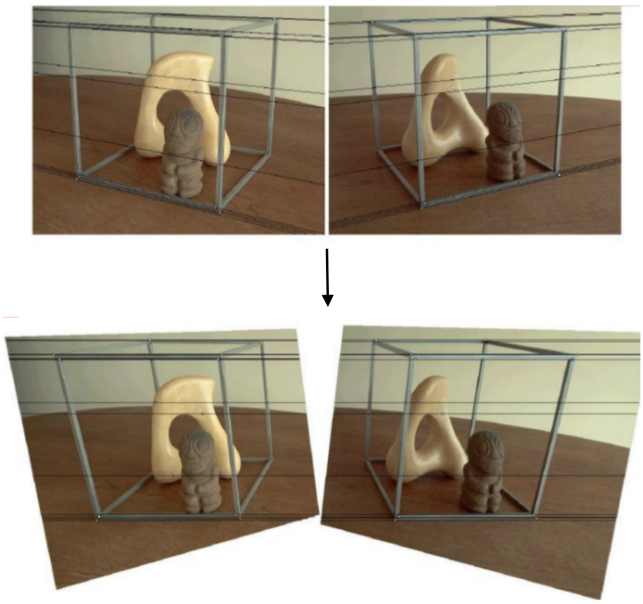


Рис. 5: Приклад виправлення зображень

2.2 Релевантність

Задача стереобачення важлива в таких областях як роботехніка, об'ємна відбудова (англ. 3D scene reconstruction), безпілотне вождіння та побудова доповненої реальності.

3 Алгоритм

3.1 Базовий алгоритм

Типовий алгоритм розв'язання задачі стереобачення починає з підрахунку cost-функції для кожного можливого значення зсуву. Ми хочемо щоб cost-функція мала низьке значення для зсувів близьких до реального і велике значення для зсувів далеких від реального.

Приклад. *Cost-функцію можна визначити як суму абсолютних різниць.*

$$Cost(x_0, y_0, d) = \sum_{(x,y)\in W(x_0,y_0)} \left| I^L(x,y) - I^R(x-d,y) \right|$$

Тут $I^L(x,y), I^R(x,y)$ *інтенсивності пікселів з координатами* (x,y) *на лівому та правому малюнку відповідно, а* $W(x,y)$ *окіл пікселя* (x,y)*. Отже, ця функція порівнює інтенсивності пікселів в околах* (x_0,y_0) *і* (x_0-d,y_0)*.*

Якщо ці пікселі відповідають одній і тій самій 3D точці, то інтенсивності в їх околах скоріш за все будуть майже однаковими і значення функції буде відносно низьким.

Простий вибір тих зсувів для яких значення cost-функції найбільш низьке зазвичай приводить до поганих результатів (як ми побачимо пізніше), тому результати обчислення cost-функції додатково оброблюються. Методи обробки не залежать від способу підрахунку cost-функції, для нашого алгоритму ми скористаємось добре відомими методами які буде наведено пізніше.

З появою великих датасетів таких як KITTI и Middleburry, в яких для різних стереопар доступна справжня карта зсувів (отримана за допомогою LIDAR чи структурованого світла), стало можливим застосування машинного навчання для того щоб обчислювати cost-функцію. Ми використовуємо датасет KITTI щоб натренувати нейронну мережу що буде розраховувати cost-функцію.

3.2 Структура нейронної мережі

На рис. 6 зображенно структуру нашої нейронної мережі. Далі наведемо визначення використаних блоків:

- *ConvBNReLu(in, out, w, h)* - поєднання декількох інших блоків:
  Convolution(in, out, w, h) $\rightarrow$ *BatchNormalization(out,* 0.001*)* $\rightarrow$ *ReLU*
- *Convolution(in, out, w, h)* - приймає тензор з вимірами $in \times x \times y$, виконує *out* згорток з $in \times w \times h$ тензором і повертає результат у вигляді $out \times x^* \times y^*$ тензора, де
$$x^* = x - w + 1,\ y^* = y - h + 1$$
- *BatchNormalization(in, esp)* - приймає тензор з вимірами $in \times x \times y$ і повертає тензор з такими самими вимірами. Детальніше про цей шар можна почитати в статі [1].
- *ReLU* - застосовує $max(0, x)$ до кожного елемента тензора
- *DotProduct* - підраховує скалярний добуток тензорів
- *LogSoftMax* - приймає вектор $(x_i)_{i=1}^n$ і розраховує
$$\left(log\left(\frac{e^{x_i}}{S} \right) \right)_{i=1}^n$$
де $S = \sum_{i=1}^n e^{x_i}$

Позначення $5*$*ConvBNReLU* значить блок *ConvBNReLU* застосований 5 разів.

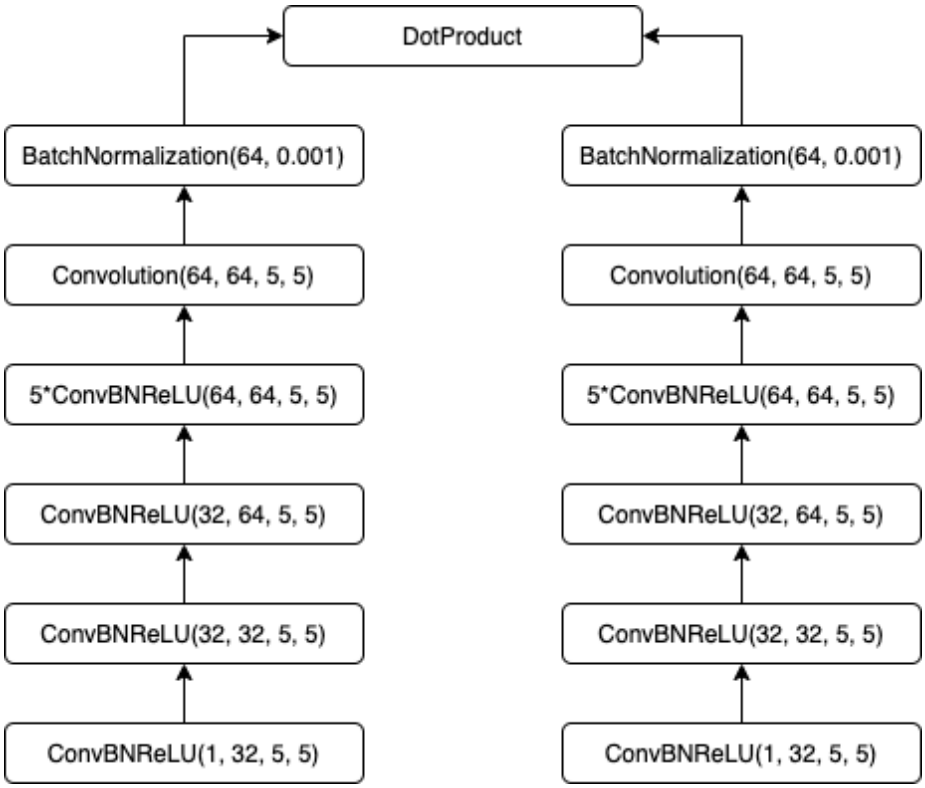


Рис. 6: Структура нейронної мережі

3.3 Застосування нейронної мережі

Щоб застосувать нейронну мережу до зображень зі стереопари ми подаємо їх у вигляді тензорів з вимірами $c \times w \times h$, де $c = 1$ якщо зображення чорно-біле (один канал кольору) і $= 3$ якщо зображення кольорове (три канали кольору). Будемо позначати ці тензори $P^L_{c,w,h}$ і $P^R_{c,w,h}$ для правого і лівого зображення відповідно. Також, оскільки кожен згорточний шар нейронної мережі зменшує ширину та висоту тензора, ми додаємо до зображень прослойку так щоб на виході ширина та висота тензора була рівна ширині та висоті зображення. Отже, в нашому випадку ми передаємо в нейронну мережу тензори $P^L_{c,w+36,h+36}, P^R_{c,w+36,h+36}$, бо ми маємо 9 згорточних шарів, кожен з яких зменшує виміри на 4, і отримуємо на виході тензори

$$O^L_{64,w,h} = (o^l_{i,j,k})^{64,w,h}_{i=1,j=1,k=1}$$

$$O^R_{64,w,h} = (o^r_{i,j,k})^{64,w,h}_{i=1,j=1,k=1}$$

Отримавши ці тензори, рахуємо cost-функцію за формулою

$$Cost(x, y, d) = -\langle O^L(x, y), O^R(x - d, y)\rangle$$

де $O^L(x,y) = (o^l_{i,x,y})_{i=1}^{64}$, $O^R(x,y) = (o^r_{i,x,y})_{i=1}^{64}$, а $\langle \cdot, \cdot \rangle$ позначає скалярний добуток.

Як інтерпретувати цю cost-функцію? Кожна гілка нейронної мережі знаходить деякі характерні риси в частині малюнка навколо пікселя. Які саме риси буде шукати мережа визначається під час тренування, оскільки ми використовуємо однакові параметри для обох гілок, вони будуть шукати однакові риси. Кожне з 64 чисел на позиції (x,y) показує наскільки властива одна з 64 рис частині малюнка навколо (x,y). Отже, кожна i-та компонента вектора $O^L(x,y)$ показує наскільки властива цій частині зображення i-та риса, так само для правого зображення і вектора $O^R(x,y)$. Чим більш схожі риси описані в цих векторах, тим більшим буде скалярний добуток і тим менше буде cost-функція

Література

[1] Sergey Ioffe та Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[2] W. Luo, A. Schwing та R. Urtasun. "Efficient Deep Learning for Stereo Matching". в: *International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[3] D. Scharstein та R. Szeliski. "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms". в: *International Journal of Computer Vision* 47.11 (2002), с. 7—42.