НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

«До захисту допущено»
Завідувач кафедри

_____
(підпис)
М. В. Грайворонський
(ініціали, прізвище)
“____” _____
2017 р.

Дипломна робота

освітньо-кваліфікаційного рівня “магістр”

за спеціальністю 8.04030101 «Прикладна математика»
на тему «Тема»
Виконав студент 6 курсу групи group
Name

Керівник Rank, Name _____
(підпис)

Рецензент Rank, Name _____
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ — 2017 року

ЗМІСТ

ВСТУП

0.1 Предисловие

Нахождение карты глубины сцены является однии из важнейших задач в области компьютерного зрения. Решение этой задачи может осуществляться несколкими способами. Хоть они все и разные, но принцип у всех один и тот же – чем дальше объект, тем меньше "изменения".

Глаз = любой зрительный сенсор (человеческий глаз/ камера и т.п)

0.2 Способы восприяти глубины

Восприятие глубины (восприятие расстояния) — зрительная способность воспринимать действительность в её трёх измерениях, воспринимать расстояние до объекта.

Данное восприятие формируется при помощи множества так называемых "глубинных признаков". Это такие характеристики, которые у одного и того же объекта на разных расстояних – различны. Их можно разделить на Монокулярные – те, для обнаружения которых будет достаточно и одного глаза / сенсора, и Бинокулярные – полноценное восприятие которых возможно только как агрегация сенсорной информации поступающей с двух глаз. Так же, иногда их делят на Динамические – требующие движения глаза или самого объекта, и Стационарные – для которых подобные действия не нужны.

0.2.1 Монокулярные глубинные признаки

- Параллакс

  Один из самых знакомых нам признаков, и очень широко используемый эффект в задачах нахождения карты глубины. Суть его проста – чем дальше объект, тем меньше он будет смещаться (на проекции) при движении глаза.

  $$.Parallax.png$$

- optical expansion!! перевести

  Если объект движется на/от нас, размер его проекции будет увеличиваться/ уменьшатся.

  $$.Optical_expansion.png$$

- Относительные размеры

  Если известно, что несколько объектов имеют похожие размеры (напр. люди), причём абсолютные размеры могут быть неизвестены, то на основании размеров их проекций можно сделать выводы про относительное расстояние до них (Если один человек на снимке значительно больше остальных, значит, скорее-всего, оно находится ближе всего к камере).

  $$.Relative_sizes.png$$

- Градиент текстуры

  Хорошо различимые мелкие детали текстуры поверхности становятся всё хуже заметными с расстоянием.

  $$.Texture_Gradient.png$$

- Свет и тени

Тени отбрасываемые объектами, то как свет падает и отражается от объектов, позволяют определять форму объектов и их положение в пространстве.

$$.Light_Shadows.pngsiluets, contours, shadows$$

0.2.2 Бинокулярные глубинные признаки

- Бинокулярный параллакс

  При использовании двух изображений одной и той же сцены, полученных под разными углами, можно триангулировать расстояние до объекта с высокой степенью точности. Именно на этом принципе основано 3д-кино и автостереограммы.

$$.Binocular_Parallax.png$$

- Конвергенция

  Присущий живым организмам признак, конвергенция – кинестические ощущения от сокращения глазных мышц при фокусировке на каком-либо объекте.

$$.Convergence.png$$

Это далеко не все существующие глубинные признаки. Более полный список можно найти здесь: TEX[1]

В данной работе будет рассматриваться использование только бинокулярного параллакса. И хоть этот метод сам по себе не даёт абсолютной информации о глубине сцены, зная некоторые дополнительные параметры её можно получить.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Нахождение абсолютного расстояния методом бинокулярного параллакса

1.1.1 О камерах

В нашем случае в качестве ”глаза” используется камера, поэтому для последующих выкладок нам нужно иметь представление о том, как она устроена и как работает.

Сегодня существует множество разных типов камер, и хоть каждая из них может иметь свои особенности конструкции, базовый принцип работы у них у всех одинаковый. Основа камеры – её светочувствительная матрица. В ней свет преобразуются в поток цифровых данных, непосредственно формируя изображение. После неё обычно находится объектив. Он отвечает за проекцию изображения на матрицу, позволяя регулировать фокусное расстояне,

значение диафрагмы и другие параметры.

Рассмотрим принцип работы камеры на примере камеры-обскуры.

В силу оптических особенностей, конечное изображение в такой камере будет перевернутым. Поэтому для ещё большего упрощения мысленно пере-

несём матрицу камеры направо, на расстояние f. Такое действие никак не повлияет на итоговый результат, но сильно упростит дальнейшие выкладки.

Покажем теперь, как с помощью двух таких камер можно найти абсолютное расстояние до объекта. Пусть пока камеры будут расположены параллельно и сонаправленны друг другу (что делать в случае когда камеры расположены иначе будет указано дальше). Итак, у нас есть такая конструкция:

Где f – фокусное расстояние камеры, Δ – расстояние между камерами, z – искомая глубина изображения, x_1, x_2 – координата объекта на левом и на правом снимке, R, L – оптические центры правой и левой камеры, S – объект.

Тогда из подобия треугольников следует:

$$\begin{cases} \frac{f}{x_1} = \frac{z}{\Delta + x} \\ \frac{f}{x_2} = \frac{z}{x} \end{cases}$$

Выразив Δ из первого уравнения, и x из второго получим:

$$\begin{cases} \Delta = \frac{zx_1}{f} - x \\ x = \frac{zx_2}{f} \end{cases}$$

Подставим x в первое уравнение и выразим из получившегося z

$$z = \frac{\Delta f}{(x_1 - x_2)}$$

Так как Δ и f не зависят от расположения объекта, а являются характеристиками самой камеры, то расстояние до объекта зависит только от разницы

$x_1 - x_2$, то есть только от смещения объекта между двумя снимками. Следовательно для нахождения карты глубин нам нужно найти "сдвиг"для каждого пикселя.

1.1.2 !!Ректификация!!

Но для того чтобы вычислить $x_1 - x_2$ нужно сначала найти эту пару соответствующих пикселей, а это сложная задача. Для каждого пискселя левого изображения, нам нужно найти соответствующий среди пикселей правого изображения. Нужно ли нам перебирать все пиксели правого изображения, или всё таки можно ограничиться каким-то подмножеством пикселей?

Пусть у нас есть изображение какого-то объекта на фоне стены.

6

Всё что мы можем сказать о его рассположении относительно камеры, это что он находится где-то на луче L.

Добавим теперь вторую камеру, на каком-то расстоянии Δ от первой камеры. Если мы теперь проведем лучи из оптического центра правой камеры так, чтобы они пересекали луч L - мы получим, так называемые, "эпиполярные линии". И теперь мы можем сократить множество для поиска соо-

Рисунок 1.1 — построчная карта сдвигов

тветствующих пар со всего изображения до эпиполярных линий на нём. Это существенно облегчает нам задачу.

1.2 Стерео-зрение при полных данных

Для простоты в этой работе рассматриваются только одномерные алгоритмы поиска карт сдвигов. Это означает, что каждую строку изображения мы обрабатываем независимо от других. Такой подход требует предварительной ректификации изображения, однако является более быстрым и дает неплохие результаты (рис. 1.1).

Однако не любая пара пикселей может находиться в соответсвии! Пикселю левого изображения с номером i могут соответсвовать только $i-k$ -е пиксели правого изображения ($k = (0,i-1)$). Вы можете убедиться в этом держа перед собой карандаш и поочередно закрывая то правый, то левый глаз. В правом глазу карандаш будет находиться левее чем в левом.

Но всё же, как нам искать соответствующие пиксели? Какими характеристиками обладают соответствующие пиксели? Логично было бы предположить, что у них одинаковый цвет. Но этого не достаточно. Тогда можно ещё

посмотреть какие свойства есть у соседних соответствующих пикселей. Допустим для пикселя x_i уже найден соответствующий ему пиксель, то есть нам уже известне сдвиг для (i)-го пикселя. Можем ли мы что-то сказать о сдвиге для $(i+1)$-го пикселя? Так как всё же большинство объектов в реальной жизни - гладкие, то велика вероятность, что сдвиг для $(i+1)$-го пикселя не будет сильно отличаться от сдвига (i)-го пикселя. Иными словами в последовательности сдвигов должно быть мало скачков.

Итак, у нас есть два признака похожести пикселей: схожесть цветов и схожесть соседних сдвигов. Введём качественные оценки этих двух характеристик. Схожесть цветов определим как расстояние между ними, в их цветовом пространстве (!опасный момент!). Схожесть же соседних сдвигов определим просто как модуль разности значений этих сдвигов.

Поставим задачу более формально. Пусть множество $\mathcal{L} = \{x_i \mid i = \overline{1,n}\}$ — левое изображение-строка, n - ширина изображения, x_i - цвет пикселя (если у нас черно-белые 8-битные изображения, то $x_i \in [\,0,\ 255\,]$, если же цветные, то $x_i \in [\,0,\ 255\,]^3$). Введём функцию $\mathcal{L}(i)$ — цвет i-го пискеля на левом изображении. Аналогично, $\mathcal{R}$ — правое изображение-строка, а $\mathcal{R}(i)$ — цвет i-го пискеля на правом изображение.

Для каждого пикселя i левого изображения нам нужно найти соответствующий ему пиксель на правом изображении, то есть найти такой сдвиг d_i, что:

1) Мимимизирует **Унарный штраф** $H(i,d) = |\,\mathcal{L}(i) - \mathcal{R}(i-d)\,|$
   (!норма, а не модуль!)
2) Мимимизирует **Бинарный штраф** $g(d,d') = \alpha\,|\,d-d'\,|$
   (где — α коэффициент сглаживания, и подбирается экспериментально)
   Можем построить штрафную функцию $\omega(\overline{d}) = \sum\limits_{i=1}^{n} H(i,d_i) + \sum\limits_{i=1}^{n-1} g(d_i,d_{i+1})$
   Тогда, такая последовательность $\overline{d}$ которая минимизирует штрафную фун-

кцию $\omega(\overline{d})$ и будет нашим решением. Таким образом имеем задачу оптимизации.

Для нахождения эффективного решения, и большего понимания задачи, представим её в несколько ином виде, а именно в виде ориентированного графа.

Множество вершин $V = \{\ \sigma(i,d)\,|\,i = \overline{(1,n)},\ d = \overline{(0,\ maxD)}\ \} \bigcup \{\ S, E\ \}$.
Вес вершины $\sigma(i,d) = H(i,d)\ ,\ i = \overline{(1,n)},\ d = \overline{(0,\ maxD)}$
Веса ребёр:

1) Из S в $\sigma(1,d)$ вес ребра = 0, $d = \overline{(0, maxD)}$
2) Из $\sigma(n,d)$ в E вес ребра = 0, $d = \overline{(0, maxD)}$
3) Из $\sigma(i,d)$ в $\sigma(i+1,d')$ вес ребра = $q(d,d')$, $d, d' = \overline{(0, maxD)}$

Получается такой граф:

10

Тогда, длина пути из вершины S в вершину E через вершины $\sigma(i,d)$ где $i = \overline{(1,n)}$, а $d \in \overline{d}$ будет равна:

$$\sum_{i=1}^{n} H(i,d_i) + \sum_{i=1}^{n-1} g(d_i, d_{i+1})$$

Что в точности повторяет нашу штрафню функцию, причём последовательность $\overline{d}$ минимизирующая эту штрафню функцию, будет отвечать последовательности вершин, которые составляют самый короткий путь из вершины S в вершину E.

Таким образом наша задача сводится к поиску кратчайшего пути на графе.

Решать эту задачу традиционными алгоритмами типа Белмана-Форда или Дейкстры - не самая лучшая идея. Алгоритм Белма-Форда имеет сложность $|V| * |E|$, причём $|E|$ у нас равна $D^2 n$, где D - максимальный диспа-

ритет, n - ширина изображения.

Куда эффективнее решать эту задачу при помощи ДИНАМИЧЕСКОГО ПРОГРАМИРОВАНИЯ.

Обозначим длину кратчайшего пути из вершины S в вершину $\sigma(i, d)$ как $f_i(d)$. Тогда для любого $d \in D$:

$$f_1(d) = H(1, d)$$
$$f_2(d) = \min(H(1,d') + g(d', d)) = \min(f_1(d') + g(d', d)) + H(2,d)$$
...
$$f_n(d) = \min(f_{n-1}(d') + g(d', d)) + H(n,d)$$

А саму последовательность $\overline{d}$ находим так:

$$d_m = arg\min_{d'}(f_m(d'))$$
$$d_i = arg\min_{d'}(f_i(d') + H(i,d'))$$

1.3 Стерео-зрение при неполных данных

1.4 Стерео-зрение при медленно поступающих данных

1.4.1 Данные приходят по порядку

Пусть сначала все пиксели нам неизвестны (рис. 1.2), поэтому мы не можем вычислить вес ни одной из вершин графа G (**??**).

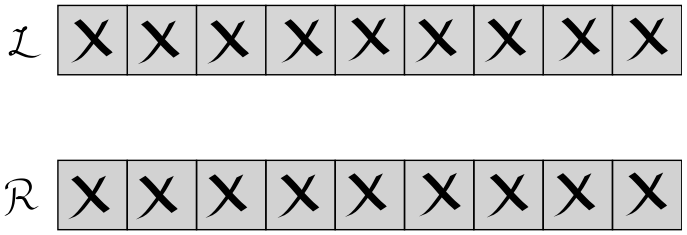

Рисунок 1.2 — Нет данных

Если мы не можем вычислить вес вершины — будем называть эту вершину "*закрытой*" и обозначать ее черным кружочком. В противном случае назовем вершину "*открытой*" и обозначим белым кружком. Для простоты рисунков положим $D_{max} = 2$ и не будем обозначать ребер. Тогда, в ситуации когда все данные нам неизвестны, граф G будет иметь следующий вид (рис. 1.3).

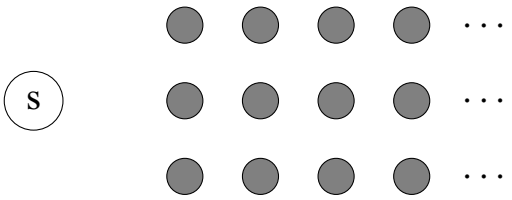

Рисунок 1.3 — граф G в отсутствии данных

Пусть теперь нам по порядку начинают приходить пары из пикселей правого и левого изображений с одинаковыми координатами. Когда нам известен только один первый пиксель каждого изображения (рис.1.4) то мы можем вычислить вес только одной вершины, и граф G будет иметь вид как на рис.1.5.

Когда же нам будут известны уже первые два пикселя обоих изображений, мы сможем найти веса уже трёх вершин, и граф G будет выглядеть как на рис.1.6.

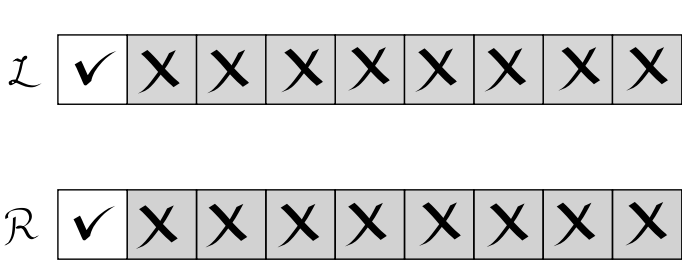

Рисунок 1.4 — Известны только первые пиксели

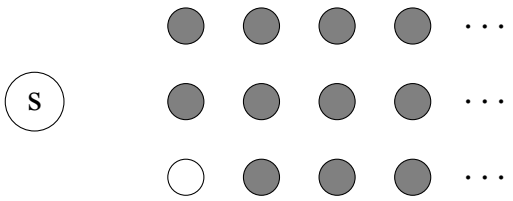

Рисунок 1.5 — Граф G когда известны только первые пиксели

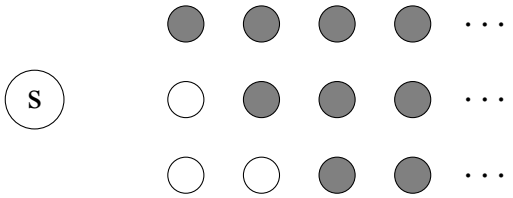

Рисунок 1.6 — Відомі перші два пікселі зображень

По мере прихода следующих пикселей в графе G будут открываться вершины на диагонали над уже открытыми вершинами. А как только нам станут известны $D_{max} + 1$ первых пикселей обоих изображений — то все вершины в первом столбике станут открытыми (рис. 1.7), и мы можем приступать к предвычислениям

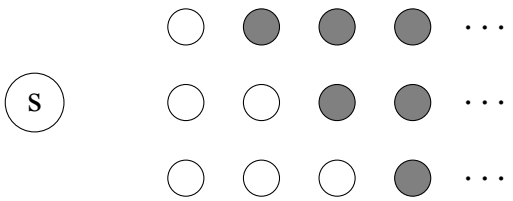

Рисунок 1.7 — Известны первые $D_{max} + 1$ пікселей

Предположим кратчайший путь проходит через какую-то вершину $\sigma(2,d)$ во втором столбике. Тогда точно можно утверждать, что он проходит через вершину $\sigma(1,k)$ первого столбика, где

$$k = \arg\min_{l=0}^{D_{max}} \big(h(1,l) + g(l,d)\big).$$

Тогда все остальные возможные пути из S в $\sigma(2,d)$ нам не нужны, и мы их отбрасываем, а вместо них вводим ребро $< S, \sigma(2,d) >$, с весом

$$g^{(1)}{}_d = \min_{l=0}^{D_{max}} \big(h(2,l) + g(l,d)\big),\ \forall d \in D.$$

Таким образом мы уменьшили количество рёбер в графе G на D_{max}.

Но мы не знаем через какую именно вершину второго столбика проходит кратчайший путь, поэтому вынуждены провести эту операцию для всех вершин $\sigma(2,d)\ \forall d \in D$.В результате такой оптимизации мы сократим количество рёбер на $(D_{max})^2$ (рис. 1.8 $\rightarrow$ рис. 1.9).

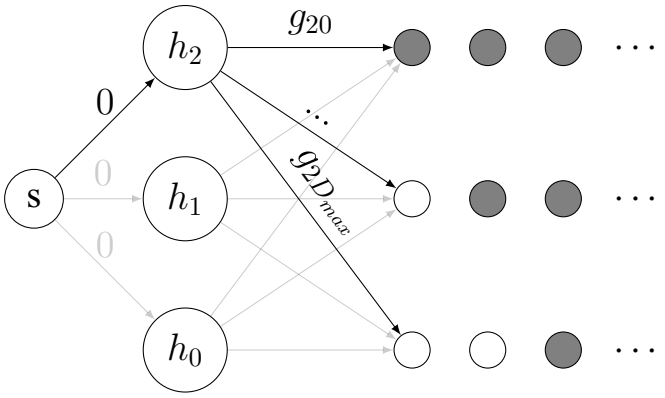

Рисунок 1.8 — граф G до оптимизации

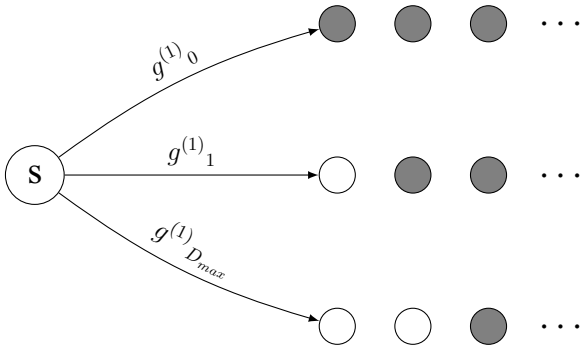

Рисунок 1.9 — граф G после оптимизации

По пришествию первых $D_{max} + 2$ стерео-пар пикселей проводим аналогичные действия, но теперь уже учитываем предыдущую оптимизацию. Для каждой вершины $\sigma(3, d)$ $d \in D$ находим вершину $\sigma(2, k)$, где

$$k = \arg\min_{l \in D} \left(g_l^{(1)} + h(2, l) + g(l, d) \right).$$

Отбрасываем все пути из S в $\sigma(3, d),\ d \in D$, и вводим рёбра $< S, \sigma(3, d) >$ $d \in D$ с весами

$$g^{(2)}{}_d = \min_{l \in D} \left(g_l^{(1)} + h(2, l) + g(l, d) \right),\ \forall d \in D.$$

Этим самым снова уменьшив общее количество рёбер графа G на $(D_{max})^2$.

Таким образом, при как только становятся известны первые $D_{max} + m$ стерео-пар пикселей $(m \geqslant 2)$, отбрасываем все пути из S в $\sigma(m + 1, d)$ $d \in D$, заменяя их рёбрами $< S, \sigma(m + 1, d) >\ \forall d \in D$, с весами

$$g^{(m)}{}_d = \min_{l \in D} \left(g_l^{(m-1)} + h(m, l) + g(l, d) \right),\ \forall d \in D, \tag{1.1}$$

уменьшая количество рёбер графа G ещё на $(D_{max})^2$.

Для восстановления конечной оптимальной последовательности $\overline{d}$, нам для каждой вершины $\sigma(i,d),\ \forall\ i = \overline{2,n},\ d \in D$ нужно запоминать предыдущую ей "*оптимальную*" вершину $\sigma(i-1,k)$, где

$$k = \arg\min_{l \in D} \left({g^{(i-1)}}_l + h(i-1,l) + g(l,d) \right).$$

Для этого введем матрицу предыдущих оптимальных вершин $\hat{\mathcal{P}}$ размера $n \times D_{max}$, где $\sigma(i-1,p_{ij})$ – предыдущая $\sigma(i,j)$ оптимальная вершина, $\forall i \in I, j \in D$. Элементы этой матрицы находятся следующим образом:

- $p_{1j} = 0,\ \forall j \in D$ поскольку в вершины первого столбца можно попасть только из начальной вершины S, и предыдущая оптимальная вершина для них всегда одна. Конечно, можно просто исключить эти элементы из матрицы, но их наличие приводит к более красивому и простому определению остальных элементов матрицы $\hat{\mathcal{P}}$.
- $p_{ij} = \arg\min_{l \in D} \left({g^{(i-1)}}_l + h(i-1,l) + g(l,j) \right)$.

Элементы p_{ij} нужно вычислять при приходе $D_{max} + i$ пары пикселей, так как при приходе следующей пары, рёбра $g_l^{(i-1)}$ используются для расчёта рёбер $g_l^{(i)},\ \forall l \in D$ согласно формуле 1.1 и отбрасываются.

После нахождения всех элементов $\hat{\mathcal{P}}$, можем восстановить все элементы последовательности $\overline{d}$ —

$$\begin{aligned} d_n &= \arg\min_{l \in D} \left({g^{(n-1)}}_l + h(n,l) \right), \\ d_{n-1} &= p_{n,d_n}, \\ &\dots \\ d_i &= p_{i,d_{n-1}}. \end{aligned}$$

1.4.2 Одно из изображений известно

2 ПРАКТИЧНІ РЕЗУЛЬТАТИ

3 FINAL REMARKS

ВИСНОВКИ

В результаті виконання роботи вдалося.

REFERENCES

ПЕРЕЛІК ПОСИЛАНЬ

1 wikipedia. The TEXbook / wikipedia // Computers & typesetting. — test, 1984.