

1 ВВЕДЕНИЕ

1.1 Постановка задачи

В результате формирования, передачи и преобразовании с помощью электронных систем, изображения могут быть подвержены различным искажениям, что в ряде случаев ухудшает качество изображения, с визуальной точки зрения, а так же скрывает некоторые участки изображений. Актуальность задач шумоподавления изображений, полученных с помощью фотокамер, видеокамер, рентгеновских снимков или другим не искусственным методом, растет с каждым годом по мере развития информационных технологий.

На текущем этапе современных средств компьютерной техники можно выделить несколько направлений. Распознавание образов - обнаружение на изображении объектов с определенными характеристиками, свойственных некоторому классу объектов. Обработка изображений - преобразует некоторым способом изображение. Визуализация - генерирование изображения на основе некоторого описания. Важную роль играют системы автоматизации всех этих процессов.

Первостепенной задачей такой системы является улучшение качества изображения. Это в первую очередь достигается за счет уменьшения количества шума на изображениях. На данный момент не существуют универсальных алгоритмов, которые позволяют это сделать. Поэтому исследования в этой области не прекращаются и по сей день.

Чаще всего шумоподавление служит для улучшения визуального восприятия, но может также использоваться для каких-то специализированных целей — например, в медицине для увеличения четкости изображения на рентгеновских снимках, в качестве предобработки для последующего распознавания и т.п. Также шумоподавление играет важную роль при сжатии изображений. При сжатии сильный шум может быть принят за детали изображения, и это может отрицательно повлиять на результирующее качество сжатого изображения.[1]

Целью данной квалификационной работы является изучение следующих алгоритмов шумоподавления: BM3D, Non-Local Means, Guided, Bilateral,

Total Variation, Markov Random Field. Так же необходимо провести сравнительный анализ данных алгоритмов. Дополнительной задачей стоит улучшения алгоритма основанного на случайных марковских полях.

Дипломная работа состоит из четырех разделов. В первом описаны модели шумов, классификация алгоритмов шумоподавления, описание выборки изображений, взятых для исследования. Во втором разделе подробно описаны изучаемые методы шумоподавления и подобраны оптимальные параметры для каждого. В третьем разделе приводится описание модификации алгоритма основанного на случайных марковских полях. В четвертом разделе сравниваются алгоритмы.

2 Проблема шумоподавления

При шумоподавлении особенную роль играет его природа. Почему он возник, зависит ли шум от изображения, является он аддитивным или мультипликативным, а так же модель шума.

Источники шума[2]:

- неидеальное оборудование для захвата изображения — видеокамера, сканер и т.п.;
- плохие условия съемки — например, сильные шумы, возникающие при ночной фото/видеосъемке;
- помехи при передаче по аналоговым каналам — наводки от источников электромагнитных полей, собственные шумы активных компонентов (усилителей) линии передачи

Типы шума

- Аддитивный шум
- Мультипликативный шум

Кроме всего прочего, так же различают и модели шума.

- Гауссовский шум
- Белый шум
- Импульсный шум
- Спекл-шум
- Пуассоновский шум

В подавляющем числе случаев шум на цифровых изображениях является аддитивным гауссовым. Так как именно он зачастую появляется

при формировании изображения. Он характеризуется добавлением какого-то значения из нормального распределения с конечным среднеквадратичным отклонением. Наложение шума на изображение можно описать следующим образом.

$$y(i, j) = x(i, j) + n(i, j) \quad (1)$$

где

- y - зашумленное изображение
- x - оригинальное изображение
- n - шум имеющий распределение гаусса
- i, j - координаты пикселя

В свою очередь распределение гаусса можно задать формулой приведенной ниже.

$$G(x) = \frac{N}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{\sigma^2}\right) \quad (2)$$

- σ - дисперсия
- μ - среднеквадратичное отклонение

Задачей шумоподавления является нахождения такого изображения x' , которое было бы максимально похоже на оригинальное изображение x . Наиболее простыми методами являются, так называемые, линейные или локальные фильтры. Они основаны на следующей идеи, пиксели в некоторой окрестности с наибольшей вероятности имеют приблизительно одинаковые значения. К ним относятся алгоритмы, которые берут информацию о новом значении из пикселей в некоторой окрестности: Вох фильтра, фильтр гаусса и медианный фильтр. Данные фильтры не применяются на практике, так как после фильтрации изображения значительно теряется и теряется информация о краях.

2.1 Классификация фильтров

В данной ВКР исследуются алгоритмы отнесенные к следующим классам методов шумоподавления:

- Нелокальные: Bilateral, Guided, Non-Local Means.
- Вейвлет-преобразования: BM3D.
- На основе апостериорной вероятности: Markov Random Field.
- На основе вариаций функции: Total variation.

2.2 Dataset

Для исследования были подобраны изображения различных типов, для определения, к каким классам изображения какие фильтры применять изображения. Имеются: фотографии архитектуры, текстуры, изображения с низкой освещенностью, портреты людей, а так же комбинация некоторых. Все изображения взяты с сайта <https://pixabay.com/images/search/people/>. Ниже приведены экземпляры изображений. Изображение архитектуры:



Рис. 1: изображение типа архитектура



Рис. 2: изображение типа плохое освещение



Рис. 3: изображение типа портрет

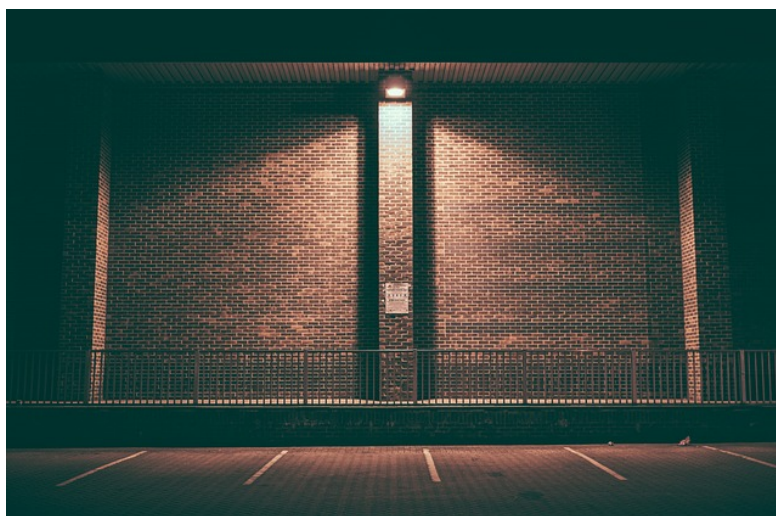


Рис. 4: изображение типа текстура

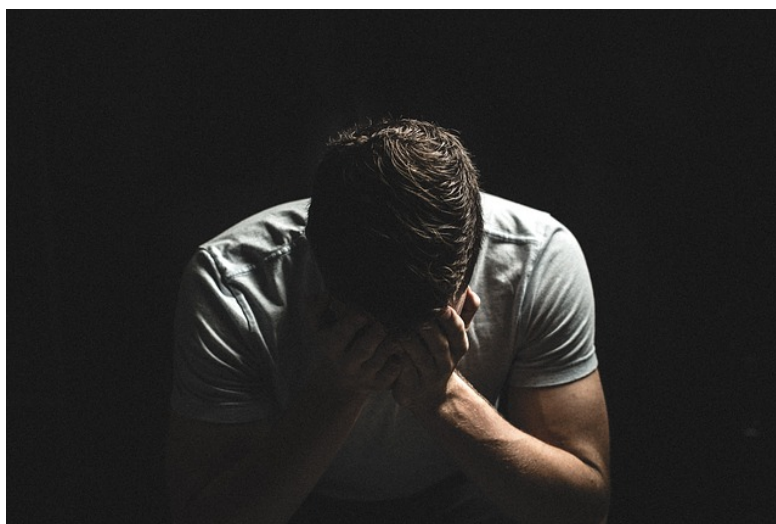


Рис. 5: изображение типа текстура

3 Описание алгоритмов

3.1 Bilateral

Основная идея билатерального фильтра заключается в том, что на текущий пиксель, влияние соседних пикселей снижается как с увеличением расстояния, так и с увеличением разницы в цветовом плане.

$$x(u) = \frac{\sum_{p \in N(u)} W_c(\|p - u\|) W_s(|y(u) - y(p)|) y(p)}{\sum_{p \in N(u)} W_c(\|p - u\|) W_s(|y(u) - y(p)|)} \quad (3)$$

где

- $x(u)$ - новое значение пикселя
- u - пиксель, для которого высчитывается новое значение
- $y(u)$ и $y(p)$ - значения пикселей в исходном изображении
- $N(u)$ - окрестность пикселя u
- W_c - весовая функция расстояния с параметров $\sigma_c = \exp(\frac{-x^2}{(2\sigma_c^2)})$
- W_s - весовая функция цвета с параметров $\sigma_s = \exp(\frac{-x^2}{(2\sigma_s^2)})$

Весовая функция подбирается в зависимости от природы шума. В данном случае выбрана функция Гаусса.

Сохранение краёв на изображении достигается за счёт весовой функции цвета. Если мы считаем новый цвет оказавшись на "темной стороне т.е. где пиксели принимают низкие значения интенсивности, то как раз "темные" пиксели будут вносить больший вклад, в то время как более "светлые" пиксели, практически не будут влиять на результат. Диапазон интенсивности, который будет влиять на итоговое значение задаётся параметром σ_s , при чем зависимость эта прямо пропорциональная. Параметр σ_c определяет радиус окрестности, которая будет влиять на итоговое значение. На практике окрестность пикселя, т.е. её радиус $= 2\sigma_c$.

Рассчитаем сложность алгоритма. Для каждого пикселя, необходимо проверить все пиксели в определенном радиусе. В таком случае сложность алгоритма $= O(nr^2)$, где n - количество пикселей в изображении, r - радиус.

Одним из главных преимуществ данного фильгтра, является его простота, а так же сохранение границ. К сожалению, он так же обладает и рядом недостатков, среди которых:

- Долгое время работы
- Артефакты на краях изображения

Попытаемся подобрать оптимальные параметры, для каждого типа изображений. В дальнейшем будем считать, что среднеквадратичное отклонение шума равно 0.05. В приложении А будут приведены полученные изображения.

3.2 Guided filter

Идея, заключается в том, что бы определить линейную зависимость каждого пикселя итогового изображения x от опорного изображения I в каждом окне ω_k с центральным пикселем k , по следующей формуле:

$$x_i = \bar{a}_i I_i + \bar{b}_i = \frac{1}{|\omega|} \sum_{k|i \in \omega_k} (a_k I_i + b_k) \quad (4)$$

где: $|\omega|$ - количество пикселей в заданном окне ω_k ;

\bar{b}_i и \bar{a}_i - средние значения линейных коэффициентов a_i и b_i во всех окнах, которые включают в себя пиксель i a_i и b_i - линейные коэффициенты для каждого окна ω_k , которые вычисляются следующим образом:

$$a_k = \frac{\frac{1}{|\omega|} \sum_{i \in \omega_k} I_i p_i + \mu_k \bar{p}_k}{\sigma_k^2 + \varepsilon} \quad (5)$$

$$b_k = \frac{1}{|\omega|} \sum_{i \in \omega_k} p_i - a_k \mu_k = \bar{p}_k - a_k \mu_k \quad (6)$$

где: μ_k и σ^2 - среднее значение и дисперсия опорного изображения I в окне ω_k ; \bar{p}_k - среднее значение входного изображения p в окне ω_k ; ε - параметр регуляризации параметра a_k что бы ограничить его рост.

Уравнение 4 можно переписать в след. виде [3]:

$$q = \sum_j W_{ij}(I) p_j \quad (7)$$

где

$$W_{ij}(I) = \frac{1}{|\omega|^2} \sum_{k:(i,j) \in \omega_k} \left(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \varepsilon}\right) \quad (8)$$

Это позволит сделать вычисления не зависимыми от величины окна ω_k . И весь алгоритм может быть представлен следующим образом.

$$\text{mean}_I = \text{boxfilter}(I, r) \text{mean}_p = \text{boxfilter}(p, r) \text{corr}_I = \text{boxfilter}(I * I, r) \text{corr}_I = I$$

Свойства сохранения краев можно объяснить следующим способом, возьмем вспомогательное изображение равное исходному т.е. $I = p$, в таком случае могут быть рассмотрены два крайних случая. В случае, когда окно будет гладким, т.е. значения пикселей будут мало отличаться друг от друга, в таком случае параметр $a_k = 0$, а $b_k = \overline{p_k}$. В противоположном случае, $a_k = 1$, а $b_k = 0$. Влияние параметра ε , можно описать следующим образом. Если среднее отклонение σ^2 меньше, чем ε , то область размывается, в ином случае она остается неизменной.

Главное особенностью данного фильтра является то, что его сложность зависит только от количества пикселей в изображении, т.е. $O(n)$ где n - количество пикселей

3.3 Non-Local mean

Обычно, новое значение пикселя берется как некое усредненное значение его соседей. К таким фильтрам относятся: билатеральный фильтр, box-фильтр и фильтр гаусса. Развитием этой идеи стал фильтр NL-mean. Суть его заключается в том, что бы искать похожие куски изображения на всей области изображения и брать некое среднее значение.

$$x_i = \sum_j w(i, j) v(j) \quad (9)$$

где

- x_i - итоговое значение пикселя.
- $w(i, j)$ - весовая функция, которая сравнивает две области изображения, с центром в пикселе i и j

- $v(j)$ - интенсивность пикселя j

Уравнение 9 указывает зависимость между значением итогового пикселя и всех пикселей исходного, зашумленного изображения.

$$w(i, j) = \frac{\exp(-\frac{\|N(i)-N(j)\|_{2,a}^2}{h})}{\sum_{j \in I} \exp(-\frac{\|N(i)-N(j)\|_{2,a}^2}{h})} \quad (10)$$

где

- h - параметр указывает степень фильтрации, чем он меньше, тем менее похожие патерны изображения будут влиять на итоговый пиксель
- $N(i)$ и $N(j)$ - соседи пикселей i и j
- a - стандартное распределение гауссовского фильтра

Не трудно заметить, что сложность данного алгоритма $O(n^2(r * 2 + 1))$, это приводит к значительным временным затратам даже для изображений малых размеров. Для уменьшения сложности вычисления, можно сделать следующие упрощения: можно заранее рассчитать все веса, так как область одного и того же пикселя может использоваться несколько раз, можно уменьшить окно поиска похожих патчей, т.е. теперь похожие патчи буду искаться не во всем изображении, а в некотором окне. Так же можно сравнивать расстояния не для каждого пикселя, а через один, два и т.д. Тогда сложность будет $= O(n * N * (r * 2 + 1))$, где N размер большого окна.

3.4 BM3D

Фильтр BM3D можно считать более лучшей версией алгоритма Non-Local mean. В нём так же одним из ключевых моментов является усреднение значений посредством поиска похожих блоков. В алгоритме можно выделить два основных шага. Каждый шаг в свою очередь разбивается на 3 этапа. Первый этап называется группировка (*grouping*), в котором для каждого патча (часть изображения фиксированного размера), отбираются другие похожие патчи. Вторым этапом является совместная (*collaborative*) фильтрация. Третьим этапом является агрегация, где для каждого пикселя вычисляется новое значение, с учетом того, что данный пиксель мог

находится в различных патчах. Далее патчем будет обозначаться часть исходного изображения с фиксированным размером.

3.4.1 Первый шаг

Группировка

Всё изображение разбивается на патчи (блоки), которые имеют размером - N_{ht}^{Psize} . Далее для каждого патча, обозначим его P , производятся следующие действия. В окне размера - N_{ht}^{Wsize} , в котором центром является P , с шагом - N_{ht}^{Step} ищутся похожие блоки 11.

$$G_{ht}(P) = \{Q | d_{ht}(Q, P) \leq h_{ht}\} \quad (11)$$

где

- $G_{ht}(P)$ - множество блоков схожих с P
- Q - один из блоков в окне
- $d_{ht}(Q, P) = \frac{\|\gamma_{\lambda_{ht}^{2D}}(P) - \gamma_{\lambda_{ht}^{2D}}(Q)\|_2^2}{(N_{ht}^{Psize})^2}$
- $\gamma_{\lambda}(x) = \begin{cases} 0 & |x| \leq \lambda \\ x & |x| > \lambda \end{cases}$ - жесткая пороговая фильтрация
- λ_{ht}^{2d} - для низких значений шума $\sigma < 40$ обычно равняется нулю, подробнее [4]
- σ^2 - дисперсия шума
- h_{ht} - порог, для того, что бы считать два блока похожими.

Максимальное количество подходящих блоков так же ограничивают, за это отвечает параметр - N_{ht}^{Pmax} . Найденные блоки объединяются в 3D-блок, далее обозначаемый $G_{ht}^{3D}(P)$, в котором блоки располагаются по мере уменьшения схожести.

Совместная фильтрация

К $G_{ht}^{3D}(P)$ применяется 3D преобразование - r_{ht}^{3D} , трансформирующие его в частотную область, после применяется жесткая пороговая фильтрация $\gamma_{\lambda_{ht}^{3D}}$. После данных операций происходит обратное 3D преобразование - $r_{ht}^{3D^{-1}}$. Данные действия описываются следующей формулой.

$$G_{ht}^{3D'}(P) = r_{ht}^{3D^{-1}}(\gamma_{\lambda_{ht}^{3D}}(r_{ht}^{3D}(G_{ht}^{3D}(P)))) \quad (12)$$

Агрегация

Теперь для каждого пикселя в изображении, необходимо рассчитать новое значение. При этом нужно учесть, что пиксель мог находиться в нескольких $G_{ht}^{3D'}$. Итоговая формула следующая.

$$y^{basic}(x) = \frac{\sum_P w_p^{hard} \sum_{Q \in G_{ht}^{3D'}(P)} \chi_Q(x) u_{P,Q}(x)}{\sum_P w_p^{ht} \sum_{Q \in G_{ht}^{3D'}(P)} \chi_Q(x)} \quad (13)$$

где

- $y^{basic}(x)$ - значение пикселя после первого шага
- $w_p^{ht} = \begin{cases} (N_p^{hard})^{-1} & N_p^{hard} \geq 1 \\ 1 & N_p^{hard} < 1 \end{cases}$
- N_p^{hard} - количество ненулевых пикселей после $\gamma_{\lambda_{ht}^{3D}}(r_{ht}^{3D}(G_{3D}(P)))$
- $\chi_Q(x)$ - равняется 1, если $x \in Q$

3.4.2 Второй шаг

На основе y^{basic} можно получить более лучшее шумоподавления, применяя фильтр Винера. Патч, для которого произведены дальнейшие действия обозначим P^{basic}

Группировка

Будем считать, что количество шума после первого шага сведенно к минимуму. Определим формулу, для поиска подходящих блоков на втором

шаге.

$$G_{wie}(P^{basic}) = \{Q | d_{wie}(Q, P^{basic}) \leq h_{wie}\} \quad (14)$$

где

- $G_{wie}(P^{basic})$ - множество блоков схожих с P^{basic}
- Q - один из блоков в окне
- $d_{wie}(Q, P^{basic}) = \frac{\|P^{basic}-Q\|_2^2}{(N_{ht}^{Psize})^2}$
- h_{wie} - порог, для того, что бы считать два блока похожими.

Совместная фильтрация Объединим блоки найденные на этапе группировка второго шага в $G_{wie}^{3D}(P^{basic})$. Затем применим 3D преобразование - r_{wie}^{3D} , как в первом шаге. Применим фильтр Винера: поэлементно умножим $G_{wie}^{3D}(P^{basic})$ на коэффициент сжатия Винера - w_P . После, применим обратное 3D преобразование - $r_{ht}^{3D^{-1}}$. Опишем эти действия следующей формулой.

$$G_{wie}^{3D'}(P^{basic}) = r_{wie}^{3D^{-1}}(w_P \cdot r_{wie}^{3D}(G_{wie}^{3D}(P))) \quad (15)$$

где

$$\bullet \quad w_P = \frac{|r_{wie}^{3D}(G_{wie}^{3D}(P^{basic}))|^2}{|r_{wie}^{3D}(G_{wie}^{3D}(P^{basic}))|^2 + \sigma}$$

Агрегация

Агрегация на втором шаге проводится по следующей формуле.

$$y(x) = \frac{\sum_P w_p^{wie} \sum_{Q \in G_{wie}^{3D'}(P)} \chi_Q(x) u_{P,Q}(x)}{\sum_P w_p^{wie} \sum_{Q \in G_{wie}^{3D'}(P)} \chi_Q(x)} \quad (16)$$

где

- $y(x)$ - итоговое значение пикселя
- $w_P^{wie} = \|w_P\|_2^{-2}$

3.4.3 Оптимальные параметры

Согласно [6], оптимальными параметрами будут следующие:

Обозначение параметра	$\sigma \leq 40$	$\sigma > 40$
N_{ht}^{Pmax}	16	16
N_{wie}^{Pmax}	32	32
N_{ht}^{Psize}	8	8
N_{wie}^{Psize}	8	8
λ_{ht}	2.7	2.7
h_{ht}	2500	5000
h_{wie}	400	3500

3.5 Total Variation

Алгоритм основан на следующем предположение. TV у зашумленного изображения больше чем у оригинала. Соответственно, необходимо найти такое изображение, которое минимизирует TV. Если ограничиться лишь данным условием, то найденное изображение будет полностью черным. Поэтому вводится дополнительное ограничение, найденное изображение и зашумленное должны быть максимально похожи

$$J(x) = \|y - x\|_2^2 + \lambda \|TV(x)\|_1 \quad (17)$$

где

- $J(x)$ - функция, которую нужно минимизировать
- y - зашумленное изображение
- x - изображение после шумоподавления
- λ - регуляризационный параметр

Уравнение 17 является оптимизационной задачей.

3.6 Random Markov Field

Список литературы

- [1] Д. Калинкина, Ватолин Д. Проблема подавления шума на изображениях и видео и различные подходы к ее решению. 2005. <http://masters.donntu.org/2013/fknt/lashchenko/library/ni.pdf> дата обращения 02.05.2019.
- [2] Р. Гонсалес, Р. Вудс. Цифровая обработка изображений. М.: Техносфера, 2005. 1072 с.
- [3] К. He, J. Sun, X. Tang.
- [4] К. Dabov A. Foi V. Katkovnik, Egiazarian K. Image denoising by sparse 3D transform-domain collaborative filtering // IEEE Trans. Image Process. 2007. Т. 16. С. 2080–2095.
- [5] Hasan Mahmud, El-Sakka Mahmoud R. Improved BM3D image denoising using SSIM-optimized Wiener filter // EURASIP Journal on Image and Video Processing. 2018. Apr. Т. 2018, № 1. с. 25. URL: <https://doi.org/10.1186/s13640-018-0264-z>.
- [6] Lebrun Marc. An Analysis and Implementation of the BM3D Image Denoising Method // Image Processing On Line. 2012. Т. 2. С. 175–213.

**

A Name of Appendix A