Метод полуквадратичного разделения (HQS) в задаче предкомпенсации изображений

Абгарян А. А. [1, 3] Аль-Казир Н. Б. [2, 3]

[1]МФТИ [2]НИУ ВШЭ [3]ИППИ РАН

15 апреля 2025

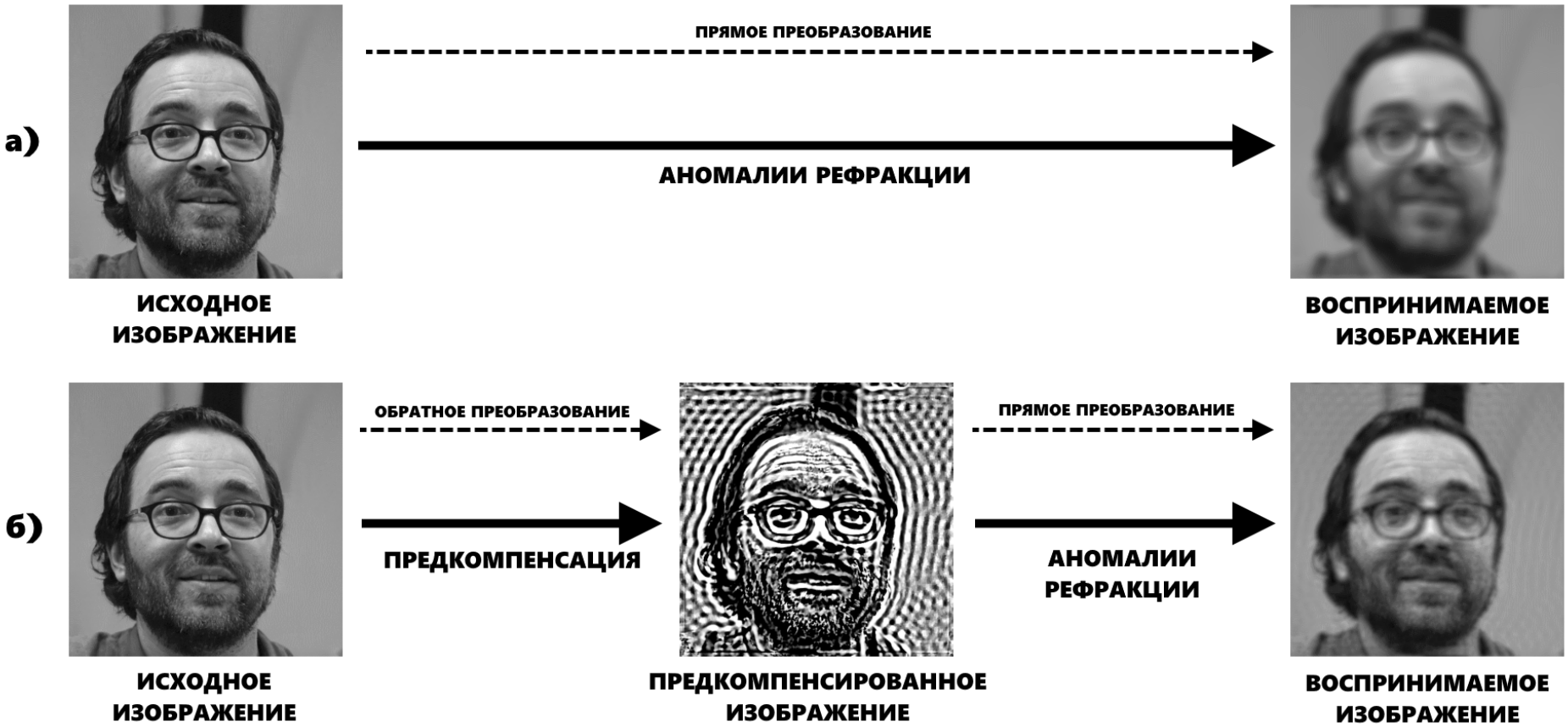
В предыдущем докладе...



Задача предкомпенсации изображений (англ. image precompensation) — это процесс программной обработки изображения, который позволяет человеку с рефракционными аномалиями, неспособному воспринимать изображение во всей его полноте, увидеть его наиболее приближенным к тому, как его воспринимает наблюдатель с идеальным зрением, при этом без изменения физических характеристик дисплея.

$$\mathbf{t}, \mathbf{K} \longrightarrow \mathbf{p} \in [0, 1] : \ \mathbf{Kp} \approx \mathbf{t}$$

В предыдущем докладе...



В предыдущем докладе...

Существующие на данный момент методы предкомпенсации:

- Методы, основанные на фильтрации Виннера (Huang et al. 2014, Alonso et al. 2003, Ji et al. 2014)
- Оптимизационные методы (Montalto et al. 2015, Jumbo et al. 2021)
  На данный момент показывают лучшие результаты среди существующих алгоритмов предкомпенсации (Alkzir et al. 2023)
- Нейросетевые методы (Tanaka et al. 2021, Güzel et al. 2023)

В предыдущем докладе...

Метод полуквадратичного разделения (HQS): пусть x — исходный сигнал, y — наблюдаемое значение. Рассмотрим задачу оптимизации:

$$\min_x f(\theta, x, y) + g(x)$$

Алгоритм HQS путем разделения переменных сводит эту задачу к итерационному процессу (Cheng et al. 2020):

$$z^{k+1} = \arg\min_z g(z) + \frac{\beta}{2}\|z - x^k\|_2^2$$
$$x^{k+1} = \arg\min_x f(\theta, x, y) + \frac{\beta}{2}\|z^{k+1} - x\|_2^2$$

Алгоритм широко применяется в задаче не-слепого деблюринга: как для решения задачи оптимизации (Zhao et al. 2024, Li et al. 2020), так и для нейросетевых методов (Zhang et al. 2020, Gnanasambandam et al. 2024)

Применяем HQS

Подход Монтальто заключается в формулировке задачи предкомпенсации как следующей задачи оптимизации:

$$\mathrm{Montalto}(\mathbf{p}, \mu) := \frac{\mu}{2}\|\mathbf{Kp} - \mathbf{t}\|_2^2 + \|\nabla \mathbf{p}\| \longrightarrow \min_{\mathbf{p}\in[0,1]}$$

здесь $\|\nabla \mathbf{p}\| = \|\mathbf{D}_x\mathbf{p}\| + \|\mathbf{D}_y\mathbf{p}\|$ — полная вариация изображения $\mathbf{p}$. Для сравнения использовалось три реализации:

- Барьерная функция: $f(\mathbf{p}) = \tau \sum_\alpha \left[e^{\lambda \mathbf{p}_\alpha} + e^{\lambda(1-\mathbf{p}_\alpha)}\right]$
- Алгоритм **FISTA** (Beck & Teboulle, 2009)
  Использовался авторами оригинальной статьи
- Шаг **ADAM** + метод проекции градиента

Применяем HQS

Рассмотрим более общий случай, когда имеется произвольное количество N линейных фильтров $\mathcal{D} = \{\mathbf{D}_i\}_{i=1,\dots N}$:

$$\min_{\mathbf{p}\in[0,1]} \frac{\mu}{2}\|\mathbf{K}\mathbf{p} - \mathbf{t}\|_2^2 + \sum_i \|\mathbf{D}_i\mathbf{p}\|$$

Применение алгоритма HQS для задачи предкомпенсации:

$$\min_{\mathbf{z}\in[0,1];\,\mathbf{p};\,\mathbf{w}} \frac{\mu}{2}\|\mathbf{K}\mathbf{p} - \mathbf{t}\|_2^2 + \sum_i \|\mathbf{w}_i\| \quad \text{s.t.} \quad \mathbf{w}_i = \mathbf{D}_i\mathbf{p} \quad \mathbf{z} = \mathbf{p}$$

В полученный функционал добавляются ограничивающие квадратичные слагаемые:

$$\min_{\mathbf{z}\in[0,1];\,\mathbf{p};\,\mathbf{w}} \frac{\mu}{2}\|\mathbf{K}\mathbf{p} - \mathbf{t}\|_2^2 + \frac{\beta_2}{2}\|\mathbf{z} - \mathbf{p}\|_2^2 + \sum_i \left[\frac{\beta_1}{2}\|\mathbf{w}_i - \mathbf{D}_i\mathbf{p}\|_2^2 + \|\mathbf{w}_i\|\right]$$

Применяем HQS

Теперь можно разделить переменные и проводить оптимизацию итеративно:

$$\mathbf{w}_i^{k+1} = \arg\min_{\mathbf{w}_i} \frac{\beta_1}{2}\|\mathbf{w}_i - \mathbf{D}_i\mathbf{p}^k\|_2^2 + \|\mathbf{w}_i\|$$

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}\in[0,1]} \frac{\beta_2}{2}\|\mathbf{z} - \mathbf{p}^k\|_2^2$$

$$\mathbf{p}^{k+1} = \arg\min_{\mathbf{p}} \frac{\mu}{2}\|\mathbf{K}\mathbf{p} - \mathbf{t}\|_2^2 + \frac{\beta_2}{2}\|\mathbf{z}^{k+1} - \mathbf{p}\|_2^2 + \sum_i \frac{\beta_1}{2}\|\mathbf{w}_i^{k+1} - \mathbf{D}_i\mathbf{p}\|_2^2$$

Применяем HQS

Получившиеся подзадачи решаются аналитически:

$$\mathbf{w}_i^{k+1} = \max\left(\left|\mathbf{D}_i\mathbf{p}^k\right| - \frac{1}{\beta_1}, 0\right) \cdot \operatorname{sign}(\mathbf{D}_i\mathbf{p}^k) \tag{1}$$

$$\mathbf{z}^{k+1} = \operatorname{clip} \mathbf{p}^k = \min(\max(\mathbf{p}^k, 0), 1) \tag{2}$$

$$\mathbf{p}^{k+1} = F^{-1}\left[\frac{\beta_2 F[\mathbf{z}^{k+1}] + \mu F[\mathbf{K}]^* F[\mathbf{t}] + \sum_i \beta_1 F[\mathbf{D}_i]^* F\left[\mathbf{w}_i^{k+1}\right]}{\beta_2 + \mu F[\mathbf{K}]^* F[\mathbf{K}] + \sum_i \beta_1 F[\mathbf{D}_i]^* F[\mathbf{D}_i]}\right] \tag{3}$$

здесь $F[\cdot]$ — двумерное быстрое преобразование Фурье. Результат применения формул (1, 2, 3) n раз:

$$\mathbf{p}^* = \text{HQS-PREC}\left(\mathbf{t}, \mathbf{K}, \{\mathbf{D}_i\}_{i=1,\dots N}, \mu, \beta_1, \beta_2\right)$$

Применяем HQS

Algorithm 1 Image precompensation using HQS

Input: source image $\mathbf{t}$; PSF $\mathbf{K}$; set of linear filters $\mathbf{D}_i$, $i = 1, \dots N$; parameters μ, β_1, β_2; threshold ε;
$\mathbf{p}^0 \leftarrow \mathbf{t}$
for $k = 0, 1, 2, \dots$ **do**
    For all i update $\mathbf{w}_i^{k+1}$ according to (1)
    Update $\mathbf{z}^{k+1}$ according to (2)
    Update $\mathbf{p}^{k+1}$ according to (3)
    if $\|\mathbf{p}^{k+1} - \mathbf{p}^k\|_2 < \varepsilon$ **then**
        return $\mathbf{p}^{k+1}$

Проблема: для достижения сходимости на уровне $\varepsilon = 10^{-4}$ необходимо ~ 150 итераций, что занимает много времени (~ 10 секунд на CPU, ~ 1 секунда на CUDA)

Обновление гиперпараметров

Значения гиперпараметров, визуально дающие хорошее качество предкомпенсации:

$$\mu \sim 10^{-2}; \quad \beta_1 \sim 10^{-6}; \quad \beta_2 \sim 10^{-3}$$

Нейросети USRNet (Zhang et al. 2020) и FIONet (Gnanasambandam et al. 2024), предназначенные для решения задачи не-слепого деблюринга, используют в своей архитектуре алгоритм HQS. В обоих статьях рекомендуется обновлять гиперпараметры HQS на каждом шаге итерации.

Обновление гиперпараметров

Идея: применить такой же подход к полученному алгоритму.

$$\mathbf{p}^*(\beta_2) = \text{HQS-PREC}\left(\mathbf{t}, \mathbf{K}, \{\mathbf{D}_i\}_{i=1,\dots N}, \mu, \beta_1, (\beta_2^0, \dots \beta_2^n)\right)$$

Проблема: как подобрать параметры $\beta_2^0, \dots \beta_2^n$? **Предложение**: обучить параметры $\beta_2^0, \dots \beta_2^n$ на некоторой выборке, используя методы машинного обучения.

Обновление гиперпараметров

Будем добиваться лучшего качества предкомпенсации используя в качестве функции потерь метрику $\mathrm{MS\text{-}SSIM}$:

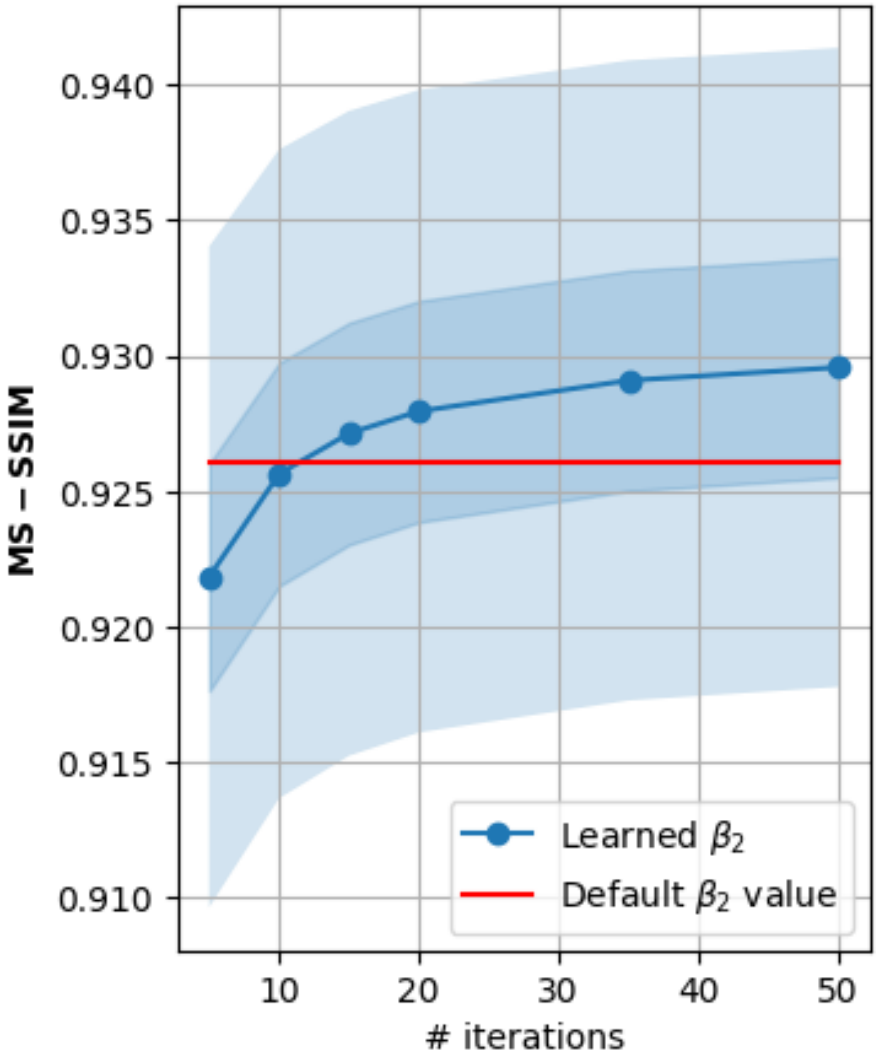
$$\mathcal{L}(\beta_2) = \mathrm{MS\text{-}SSIM}\left(\mathbf{Kp}^*(\beta_2), \mathbf{t}\right)$$

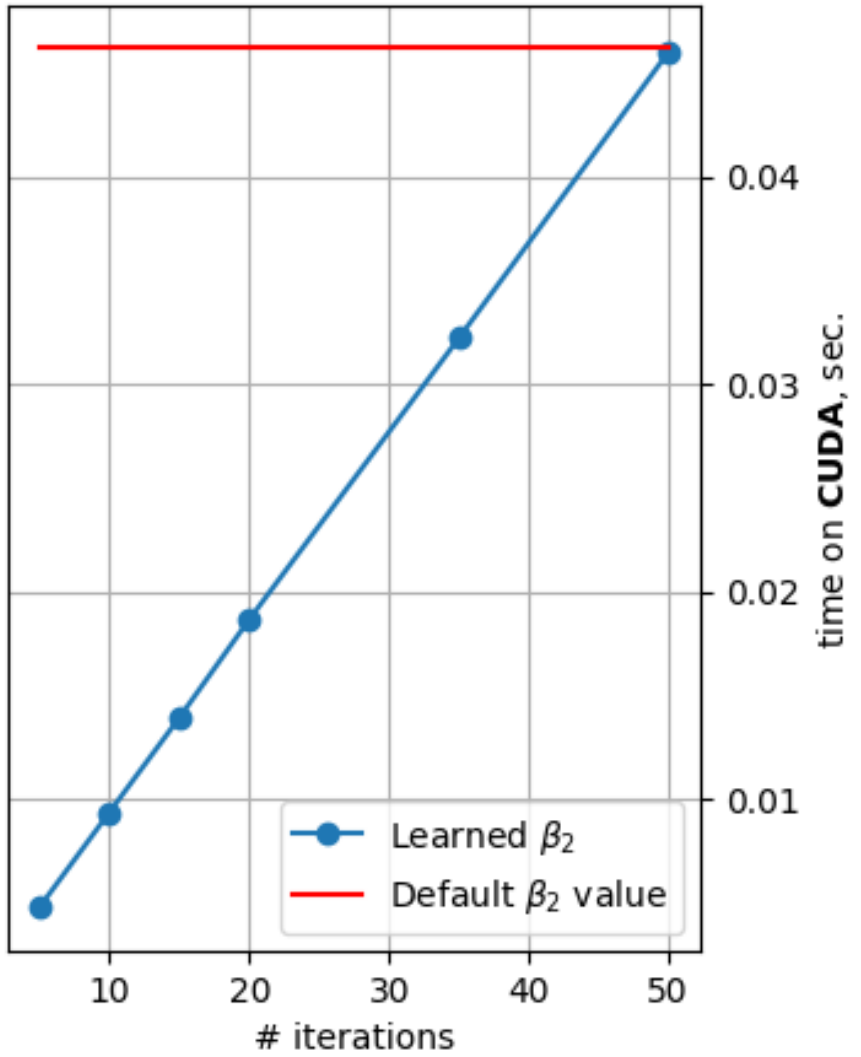
В качестве обучающей, валидационной и тестовой выборки были выбраны по 100 случайных пар изображение + ФРТ из датасета. Для обучения был выбран алгоритм оптимизации ADAM, параметр обучения $\mathtt{lr} = 10^{-6}$, обучение проводилось на 25 эпохах.

Описание датасета

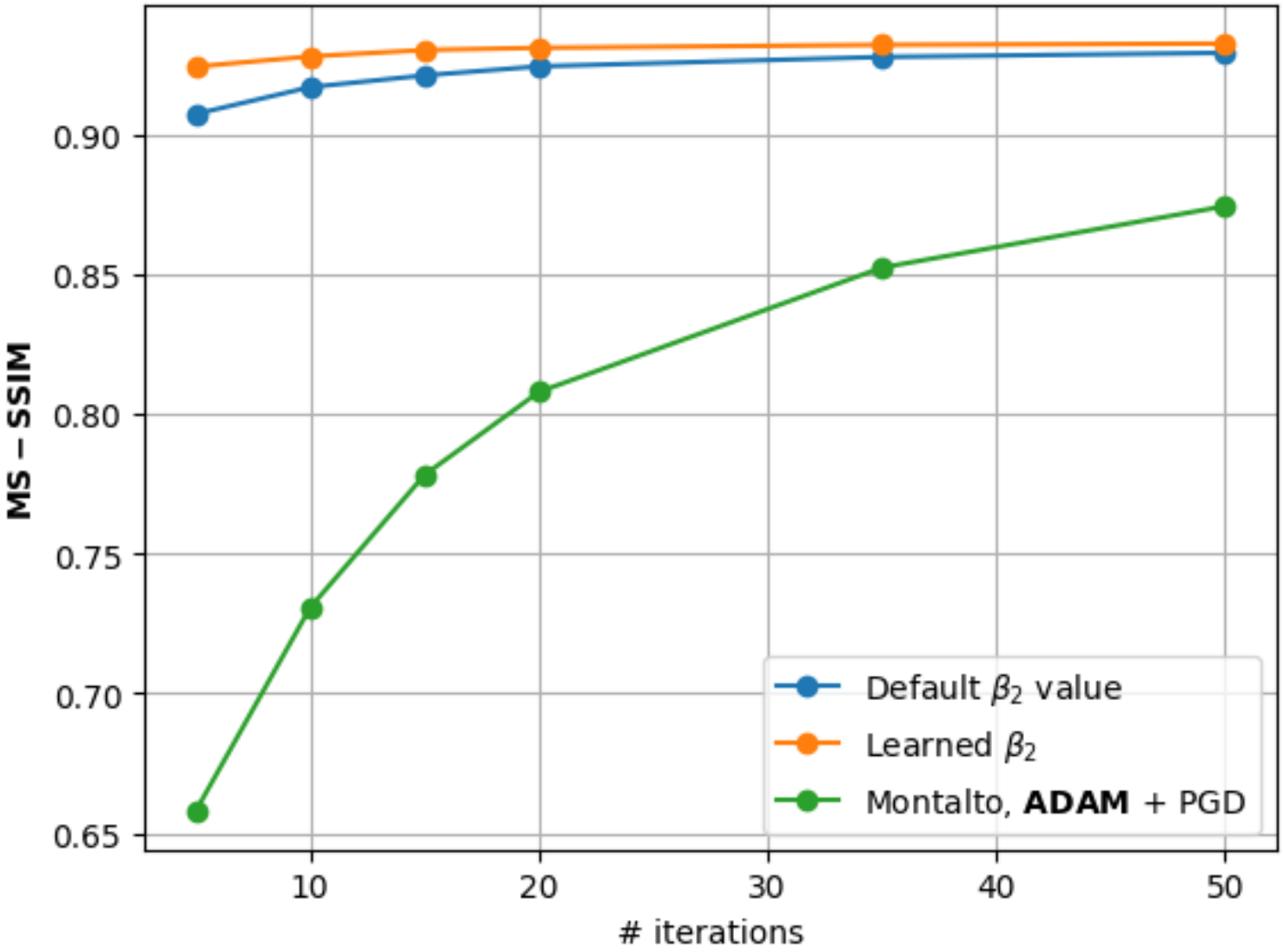
В качестве набора данных для проведения экспериментов использовалась часть датасета SCA-2023 (Alkzir et al. 2023).

Влияние количества итераций на качество





Влияние количества итераций на качество

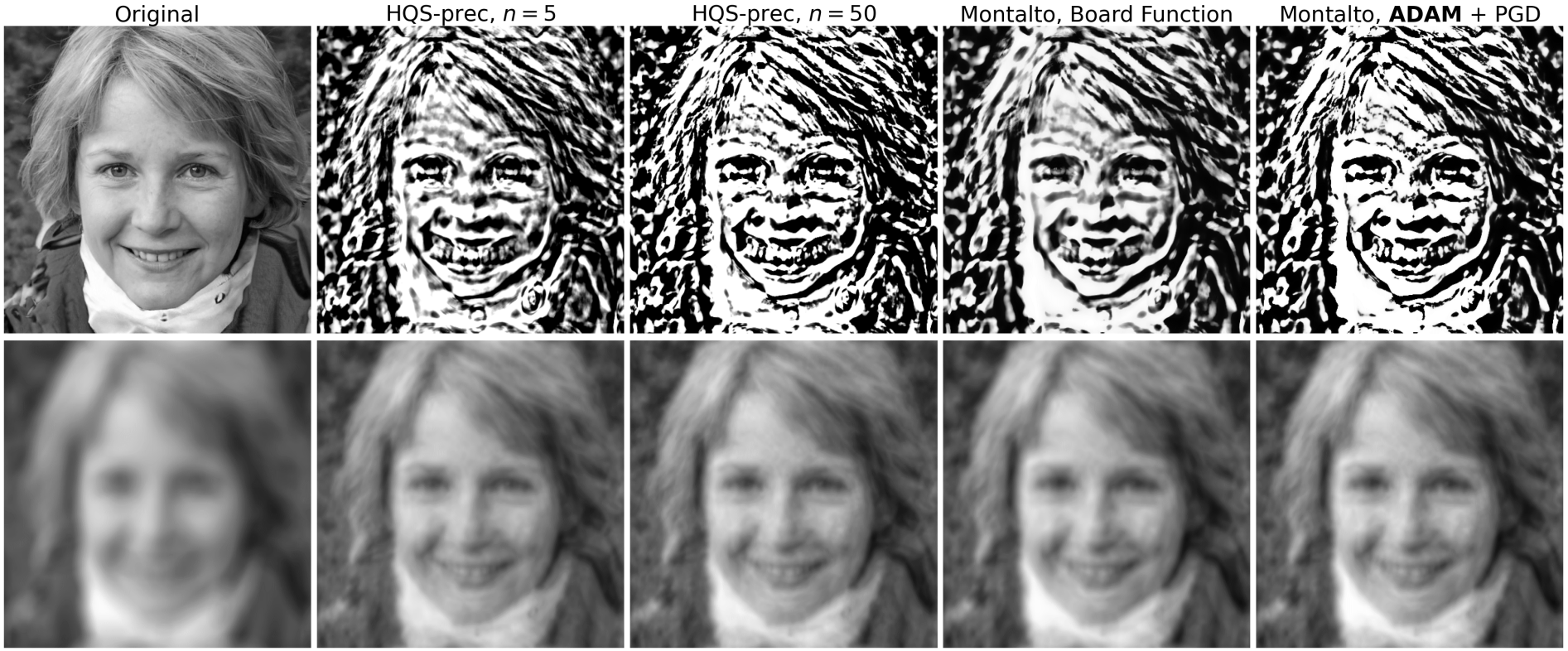


Сравнение качества и времени работы

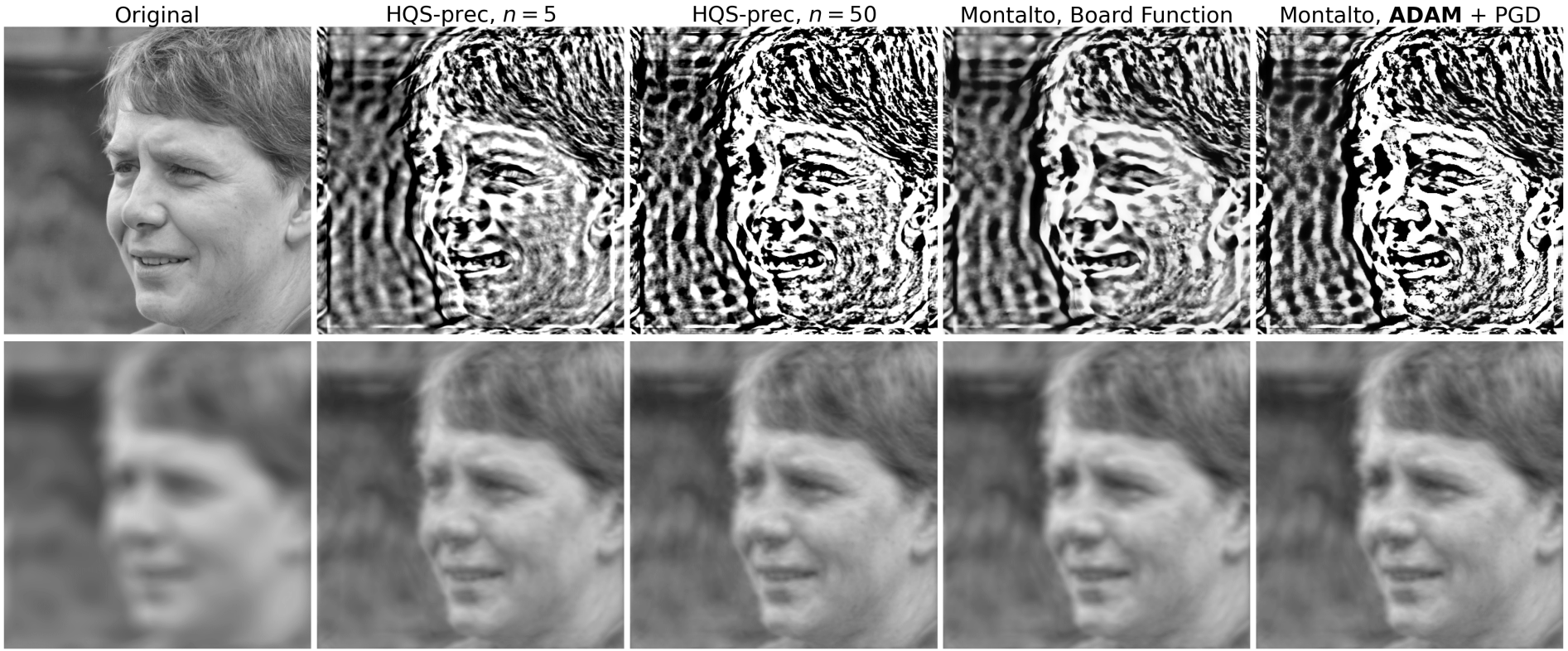
Согласно исследованию (Alkzir et al. 2024) одними из лучших метрик для оценки качества предкомпенсации являются MS-SSIM, SSIM, CORR, STRESS, NRMSE, PSNR.

Metric	HQS-PREC			MONTALTO		
	$n = 5$	$n = 10$	$n = 15$	Board Function	**FISTA**	**ADAM** + PGD
MS-SSIM	0.926	0.929	**0.931**	0.925	0.927	0.930
SSIM	0.870	0.877	**0.879**	0.870	0.873	**0.879**
CORR	0.979	0.980	0.980	0.980	0.980	**0.981**
STRESS	0.907	0.911	0.913	0.914	0.916	**0.917**
NRMSE	0.906	0.910	0.912	0.914	0.915	**0.916**
PSNR	27.359	27.861	28.070	28.381	28.553	**28.674**
Average time	0.006	0.011	0.015	0.299	1.006	3.927

Примеры работы алгоритма



Примеры работы алгоритма



Дальнейшие цели

- Осуществить проведение экспериментов по сравнению качества на более широкой выборке из датасета
- Провести обучение гиперпараметра β_1 по аналогии с β_2
- Встроить предложенный алгоритм в двухвходовую нейросеть для предкомпенсации
- Протестировать предложенный алгоритм на видео