

*Государственное образовательное учреждение высшего профессионального
образования*

*«Московский государственный технический университет
имени Н. Э. Баумана»
(МГТУ им. Н.Э. Баумана)*

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Р А С Ч Ё Т Н О - П О Я С Н И Т Е Л Ь Н А Я З А П И С К А
к курсовой работе по компьютерной графике на тему:

Автоматизированный метод обработки графических новелл.

Студент _____ Якубаускайте М.А.
(Подпись, дата)

Руководитель курсового проекта _____ Оленев А.А.
(Подпись, дата)

Москва 2018

Содержание

Введение	3
1 Аналитический раздел	4
1.1 Описание структуры манги.	4
1.2 Методы, решающие задачу колоризации графических изображений.	6
1.2.1 Подготовка к закраске. Выделение областей колоризации.	8
1.2.2 Подготовка к закраске. Выделение областей колоризации. Нейросетевой метод.	9
1.3 Колоризация изображения с использованием цветовых подсказок.	10
1.4 Колоризация изображения с использованием цветного опорного.	13
1.5 Сравнительный анализ методов колоризации графических новелл.	15
2 Конструкторский раздел	19
2.1 Описание общего алгоритма работы.	20
3 Технологический раздел	23
3.1 Выбор инструментов для разработки.	23
3.2 Процесс обучения.	23
4 Исследовательский раздел	24
4.1 Примеры работы разработанного метода.	24
4.2 Исследование времени работы.	26
Заключение	28
Список использованных источников	29

Введение

Автоматизированная колоризация графических новелл - это проблема, стоящая на стыке компьютерной графики и машинного зрения.

Изучение существующих методов для колоризации рисунков, анализ их быстродействия, выявление недостатков и разработка собственного метода колоризации - цель данного курсового проекта.

Данная работа также посвящена колоризации графических изображений, называемых мангой. Манга - это японские комиксы, которая по графическому и литературному стилю заметно отличается от заладных, несмотря на то, что развивалась под их влиянием. Сценарий и расположение кадров строятся по-другому, в изобразительной части акцент делается на линиях рисунка, а не на его форме. Рисунок практически всегда чёрно-белый; он может варьироваться от фотorealистичного до гротескного, для передачи света, текстуры используются различные виды штриховок.

Вот почему колоризация манги требует детального разбора каждого кадра. Способ автоматизации процесса анализа изображения и дальнейшая закраска будут полезны при создании цветной манги, выступающей в роли сценария к аниме (японские мультифильмы).

Разработка автоматизированного метода колоризации графической новеллы, в нашем случае - манги, это создание более красочного представления уже имеющейся истории в считанные минуты при минимальных затратах человеческих ресурсов.

1 Аналитический раздел

1.1 Описание структуры манги.

Страницы манги - это графическое представление сценария (Рисунок 1.1). Сценарий отражает события, характеры персонажей, описывает места действий, отражает разбиение на страницы и кадры. Разбиение на кадры принято называть раскадровка. Процесс раскадровки зависит от типа сюжета: комедийный, мелодраматический или трагический требует своего способа постановки сцен. Раскадровка - это основа будущего кадра (фрейма) (Рисунок 1.1 а.). Внутри каждого фрейма располагаются баллоны, содержащие диалоги (Рисунок 1.1 б.). При отрисовке фреймов существует правило "динамичной сцены" (Рисунок 1.1 в.).

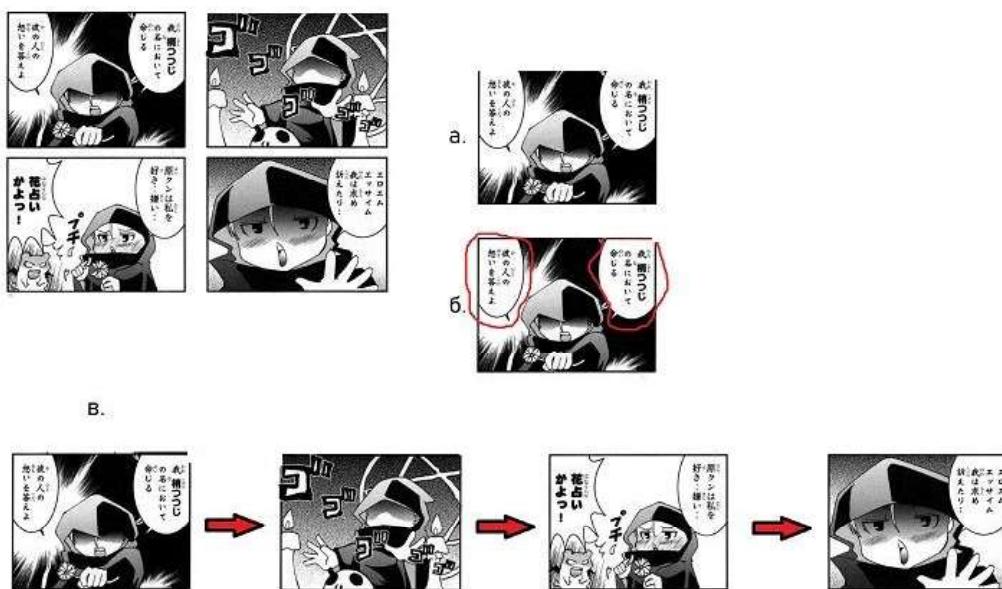


Рисунок 1.1 — Структура манги.

На одной странице движение персонажа обуславливается сохранением направления перемещения. При изображении движения внутри кадра используются вихри, описывающие направление движения и следа, отображающего положение на предыдущем шаге бледнее, чем на текущем, в рамках одного фрейма (Рисунок 1.2).

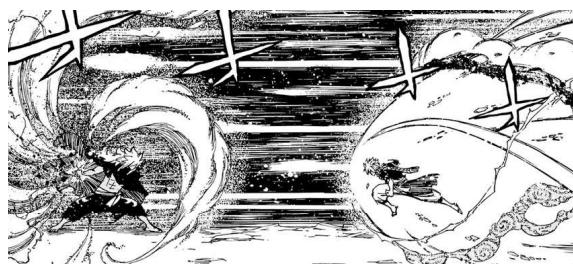


Рисунок 1.2 — Пример движения внутри фрейма.

Для отображения мимики и ее изменения используются крупные планы, для композиции общие, для второстепенных событий средние (Рисунок 1.3).



Рисунок 1.3 — Пример отображения мимики и второстепенного плана.

Панорамные (горизонтальные или вертикальные) кадры дают возможность выразительно показать движение, поскольку в движении главное - большая амплитуда (Рисунок 1.2). Если действие должно изображаться в разных местах, с разными персонажами в одно и то же время, персонаж придается воспоминаниям либо мечтает, изменяется объект или оформление страницы, которые подчеркивают факт, что события все таки отделены (например фоновый цвет страницы может быть не белым, а черным). Предметы заднего плана всегда имеют меньшую детализацию и изображаются либо более тонким контуром, либо более плотной массой, в отличие от деталей переднего плана. Тени на переднем плане более плотные и могут быть черными, в то время как на фоне они будут сделаны скринтоном (особый вид штриховки). Линии - попадающие под свет более тонкие, чем те, что находятся в тени. Тени бывают двух видов: рисующие объем поверхности и падающие (которые отбрасывают одни предметы на плоскости других). (Рисунок 1.4)

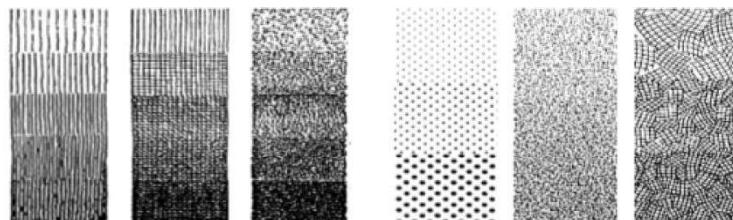


Рисунок 1.4 — Примеры используемых штриховок и скринтона.

1.2 Методы, решающие задачу колоризации графических изображений.

Существует несколько методов, решающих проблему колоризации графической новеллы. Однако, среди них всех можно выделить два основных:

- а) колоризация изображения с использованием цветовых подсказок, полученных от пользователя;[5]
- б) колоризация изображения, с использованием цветовой функции, полученной из опорного изображения.[2]

Программное решение проблемы колоризации с использованием цветовых подсказок делится на основные этапы, представленные на диаграмме IDEF0 (Рисунок 1.5 и Рисунок 1.6).

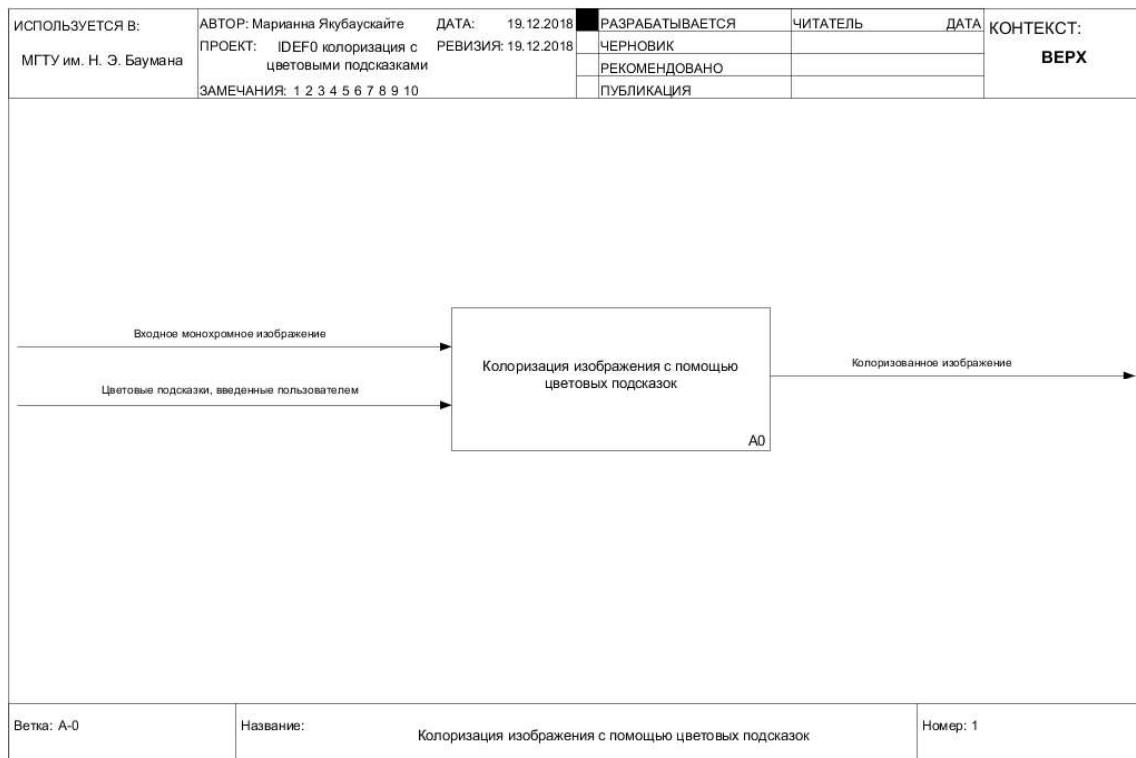


Рисунок 1.5 — IDEF0 - метод колоризации с использованием цветовых подсказок.

Программное решение проблемы колоризации изображения с использованием опорного цветного изображения, графика соответствия и квадратичного программирования представлено на диаграмме IDF0 (Рисунок 1.7 и Рисунок 1.8).

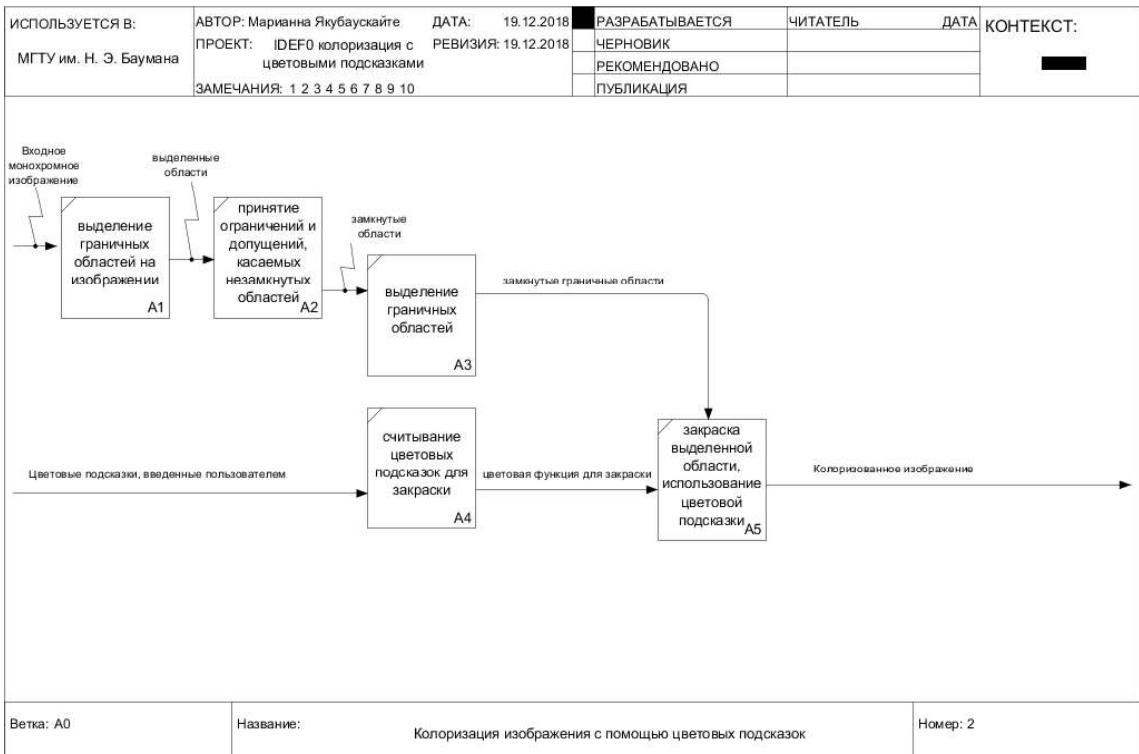


Рисунок 1.6 — IDEF0 - метод колоризации с использованием цветовых подсказок.

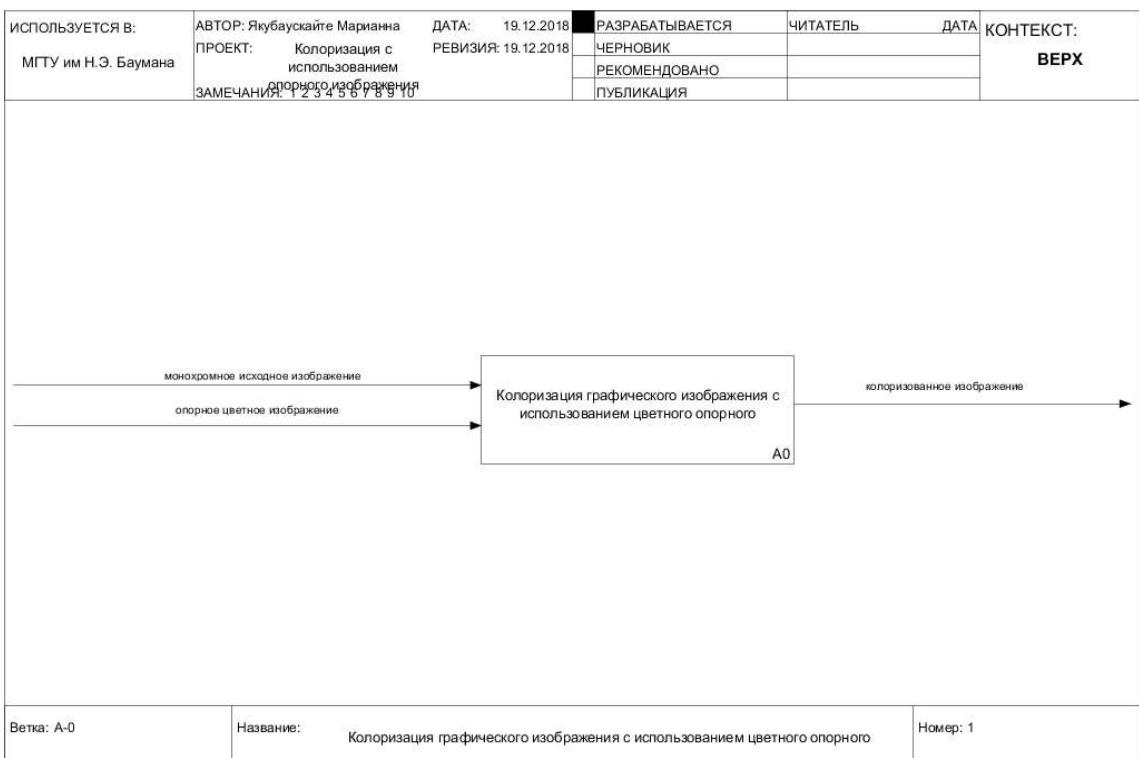


Рисунок 1.7 — IDEF0 - метод колоризации с использованием опорного изображения.

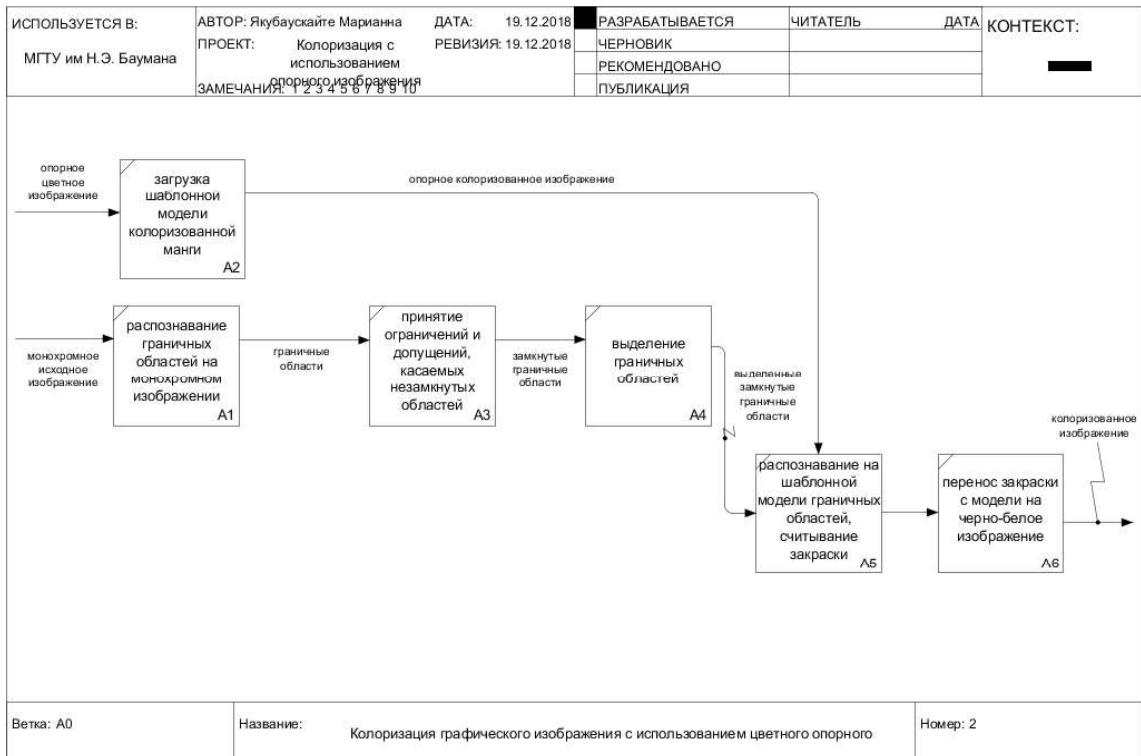


Рисунок 1.8 — IDEF0 - метод колоризации с использованием опорного изображения.

1.2.1 Подготовка к закраске. Выделение областей колоризации.

Для колоризации манги на начальном этапе необходимо определить области закраски, то есть выделить контур рисунка. Выделение контура, как и выделение границ изображения относится к поиску объектов и их выделению, основанному на алгоритмах, которые определяют точки цифрового изображения, резко изменяющие яркость и имеющие неднородности. Границы и края областей сильно связаны, так как часто существует сильный перепад яркости на границах областей. Обнаруженные края бывают разорванными. Но чтобы выделить объект на изображении, нужны замкнутые границы области. Для незамкнутых областей используют дополнение: соединяются концы линий, вычисляется площадь полученной фигуры (объемлющая оболочка), выбирается область равная максимальному размеру объемлющей оболочки. Так как объемлющая оболочка – это абстрактно замкнутая область, которая может не учитывать специфику границ, используется функция ошибки.

Область контура инициализируется функцией:

$$v(s) = [x(s), y(s)], s \in [0, 1]; \quad (1.1)$$

где $v()$ - функция, описывающая контур изображения, s - контур, $x(s)$, $y(s)$ - координаты x и y , описывающие каждую точку контура соответственно. Данная функция "перемещается" по пространственной области изображения с целью минимизации энергии перепадающей интенсивности, до тех пор, пока не будет подобрана идеаль-

ная область изображения. Энергия описывается следующим образом:

$$E = \int_0^1 \frac{1}{2} |\alpha(v'(s))^2 + \beta(v''(s))^2 + E_{ext}(v(s))| ds; \quad (1.2)$$

где E_{ext} - выходящая энергия (то есть энергия идущая из внутренней области во внешнюю), α, β - весовые коэффициенты, которые численно определяют замкнутость и стянутость относительно центральной точки:

$$E_{ext} = -[\nabla I(x, y)^2]; \quad (1.3)$$

где $\nabla I(x, y)$ - дивергенция интенсивности цвета границы. Функция E также еще обозначается как E_{int} - входящая энергия. Минимальное значение входящей энергии в контур, будет однозначно задавать контур, аппроксимирующий искомую область.

1.2.2 Подготовка к закраске. Выделение областей колоризации. Нейросетевой метод.

Для сегментации изображения - выделения границ так же используют нейронную сеть U-Net. U-Net — это архитектура свёрточной нейронной сети, предназначенная для сегментации изображений. Архитектура сети представляет собой последовательность слоёв свёртка и пулинг, которые сначала уменьшают пространственное разрешение картинки, а потом увеличивают его, предварительно объединив с данными картинки и пропустив через другие слои свёртки. Таким образом, сеть выполняет роль своеобразного фильтра. Структура сети представлена на Рисунок 1.9.

Сеть содержит сжимающий путь (слева) и расширяющий путь (справа), поэтому архитектура похожа на букву U, что и отражено в названии. На каждом шаге мы удваиваем количество каналов признаков.

Сжимающий путь похож на типичную свёрточную сеть, он содержит два подряд свёрточных слоя 3×3 , после которых идет слой ReLU и пулинг с функцией максимума 2×2 с шагом 2.

Каждый шаг расширяющего пути содержит слой, обратный пулингу, который расширяет карту признаков, после которого следует свертка 2×2 , которая уменьшает количество каналов признаков. После идет конкатенация с соответствующим образом обрезанной картой признаков из сжимающего пути и две свертки 3×3 , после каждой из которых идет ReLU. Обрезка нужна из-за того, что мы теряем пограничные пиксели в каждой свертке. На последнем слое свертка 1×1 используется для приведения каждого 64-компонентного вектора признаков до требуемого количества классов.

Всего сеть имеет 23 свёрточных слоя.

Для обучения сети, считается мера Жаккара, которая показывает меру сходства — применимо к решению задачи выделения границ, показывает меру площади

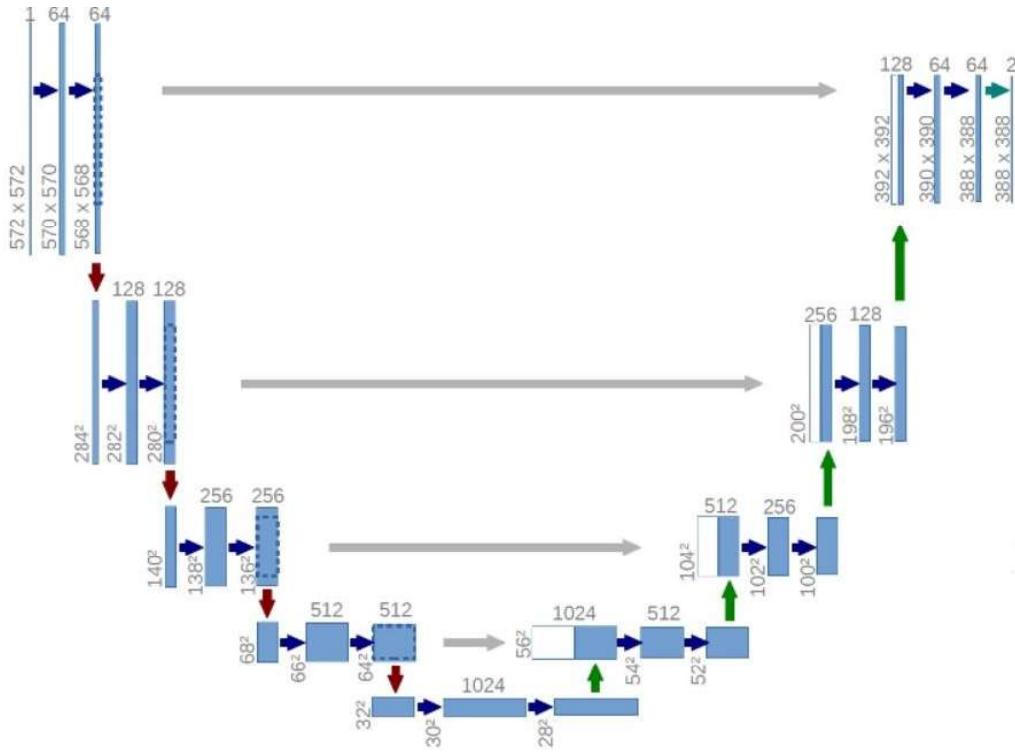


Рисунок 1.9 — Архитектура U-Net.

правильно отмеченных сегментов (отношение площади пересечения к площади объединения).

Мера Жаккара — бинарная мера сходства, безразмерный показатель сходства сравниваемых объектов.

$$K_j = \frac{a}{a + b - c}; \quad (1.4)$$

где а - количество видов на первой пробной площадке, b - количество видов на второй пробной площадке, с - количество общих видов на первой и второй площадках.

Пусть А - это множество, описывающее количество видов с первой пробной площадки, В - множество, описывающее количество видов со второй пробной площадки, $n(A)$, $n(B)$ - мера множества А и В соответственно.

Тогда, используемая в U-Net мера различия выражается так:

$$F_{1,-1} = 1 - \frac{n(A \cap B)}{n(A) + n(B) - n(A \cap B)} - \frac{n(A \cup B) - n(A \cap B)}{n(A \cup B)} \quad (1.5)$$

1.3 Колоризация изображения с использованием цветовых подсказок.

Автоматизированный метод раскраски манги с использованием цветовых подсказок решает задачу колоризации графических монохромных изображений.

Данный метод использует U-Net (для выделения границ) и архитектуру GAN (генеративная нейронная сеть). GAN представляет собой алгоритм машинного обучения без учителя, построен на комбинации двух нейронных сетей: генеративной, которая занимается генерацией образов, и, дискриминативной, которая отличает правильные ("подлинные") образцы от неверных.

На Рисунок 1.10 представлена общая структура метода.



Рисунок 1.10 — Структура метода колоризации с использованием цветовых подсказок.

GAN включает в себя 2 нейронные сети: генеративную $G(z; \theta_g)$ и дискриминативную $D(x; \theta_d)$. $G(z; \theta_g)$ - сеть, которая оценивает вероятность равномерного распределения $p_g(x)$ по x входным данным, z - это входные выделенные границы изображения. Сеть $D(x; \theta_d)$ выступает в роли классификатора - выносит суждение о том, является ли изображение сгенерированным или оно из реального набора. $G(z; \theta_g)$ - используется для прямой генерации из эскиза и цветовой подсказки цветного изображения внутри аналогично построенной U-Net сети. Дискриминатор В U-Net - $D(x; \theta_d)$ реализован, как двухклассовый классификатор. Для обучения двух нейронных сетей определена потеря, в зависимости от пройденных m итераций при классификации как :

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m (\log(D(x^i; \theta_d)) + \log(1 - D(G(z^i; \theta_d); \theta_d))) \quad (1.6)$$

$D(x; \theta_d)$ быстро обучается отличать реальную картинку от мусора, вызываемого $G(z; \theta_g)$. Процесс генерации более подходящих изображений, корреляции ошибки генератора, и постепенного инкрементирования значения ошибки представлено на графике Рисунок 1.11.

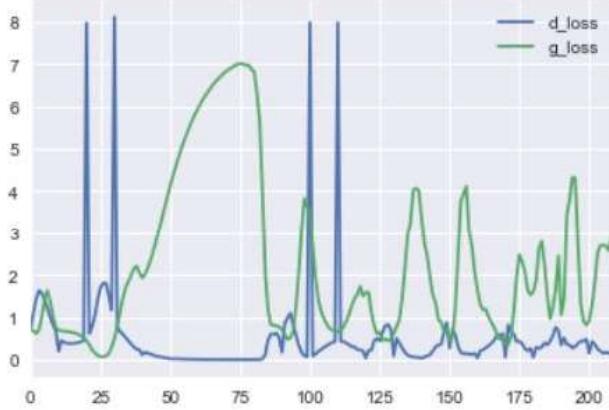


Рисунок 1.11 — Кривые потерь для генеративной и дискриминативной сетей.

[1]

Для получения более гладкого изображения, реализации плавных цветовых переходов, перед уровнем линейного предсказания, до использования дискриминатора, применяется формула тотального сглаживания, выступа, для каждого пикселя внутри строки (меняется координата y).

$$V(y) = \sum_{i,j} \sqrt{|y_{i+1,j} - y_{i,j}|^2 + |y_{i,j+1} - y_{i,j}|^2} \quad (1.7)$$

Ввиду графика (Рисунок 1.9) потеря, можно также сделать вывод, о том, что на практике трудно балансировать тренировку генератора и дискриминатора. Если дискриминатор слишком силен, его потеря скоро станет нулевой, а потеря генератора продолжит инкрементироваться. Для решения этой проблемы используется балансировка методом Вассерштейна. Тогда итоговая функция потери дискриминатора и генератора будет определена по формуле 1.8. W-расстояние (расстояние Вассерштейна) может быть аппроксимировано отрицательным значением, которое интерпритируется как декремент потери дискриминатора. Изображениям более высокого качества будет соответствовать меньшее значение W-расстояния. W-расстояние является случайной величиной, распределенной по показательному закону.

$$L(D) = E_{x \sim p_g}[D(x)] + \lambda E_{x \sim p_\xi}[\|\nabla D(x)\|_{p_g} - 1]^2 \quad (1.8)$$

Генератор $G(z; \theta_g)$ на каждой итерации обучения борется за инкремент значения потери, в свою очередь, для поддержки балансирования дискриминатор $D(x; \theta_d)$ пытается декрементировать потерю, для генерации изображений неотличимых от

оригинальных. Состязательный процесс между генератором и дискриминатором поддерживается особой структурой последнего, которая после каждого обновления значений, переданных генератором, имитирует подъем по градиентному спуску. В тоже время, при обучении генератора используется лишь метка, несущая значение потери, полученное от дискриминатора.

Для работы с цветовыми подсказками синтезируется эскиз для генератора в виде $G(z; z_{hint}; \theta_g)$, изменяется предварительный вывод дискриминатора на $D(x; z_{hint}; \theta_d)$. Случайно поданное в сеть изображение z инициализирует карту изображения, которая представляет собой матрицу объектов входного канала сети. Матрица, заполненная случайно поданным изображением, инициализирует связь между входным каналом и дискриминатором.

1.4 Метод колоризации с использованием цветного опорного изображения.

Данное решение закрашивает монохромное изображение используя опорное цветное изображение, график соответствия и метод квадратичного программирования. Работа алгоритма, разделенная по основным этапам, представлена на Рисунок 1.12.

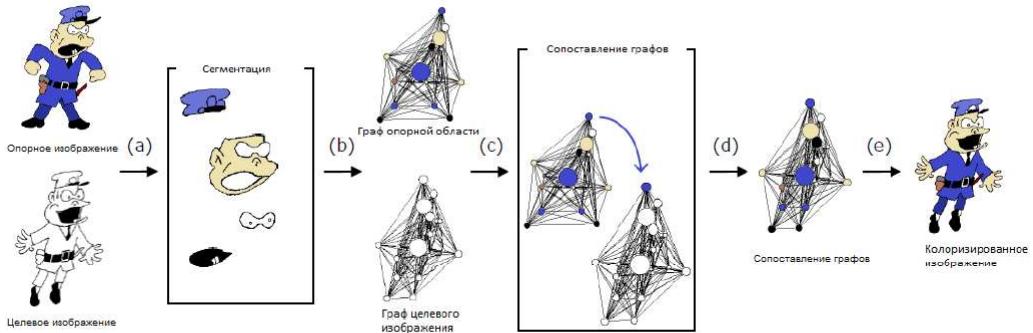


Рисунок 1.12 — Пример работы алгоритма

[2]

Граф опорного изображения обозначим, как $G_r = (V_r, E_r)$, а граф целевого $G_t = (V_t, E_t)$, где V - обозначает набор узлов, а E - набор ребер. Решение определено, как матрица перестановок $X = (x_{i,j})_{N_r \times N_t} \in \{0,1\}^{N_r \times N_t}$. N_t и N_r обозначают число вершин целевого и опорного графов соответственно.

Когда i -ая область опорного изображения сопоставляется с j -ой областью целевого изображения для каждого столбца и строки матрицы X должно выполняться:

$$\sum_j x_{i,j} = 1 \quad (1.9)$$

$$\sum_i x_{i,j} = 1 \quad (1.10)$$

Для реализации сопоставления графов G_r и G_t была введена функция стоимости $Q(i_1, j_1, i_2, j_2)$:

$$Q(i_1, j_1, i_2, j_2) = D_{length} * D_{angle} * D_{area} \quad (1.11)$$

$$D_{length} = \frac{||V_{i_1, i_2}| - |V_{j_1, j_2}||}{\max(V_{i_1, i_2}, V_{j_1, j_2})} \quad (1.12)$$

$$D_{angle} = \frac{1}{2}(1 - \frac{V_{i_1, i_2} * V_{j_1, j_2}}{|V_{i_1, i_2}| |V_{j_1, j_2}|}) \quad (1.13)$$

$$D_{area} = \frac{|a_{i_1}a_{j_2} - a_{i_2}a_{j_1}|}{\max(a_{i_1}, a_{j_1}, a_{i_2}, a_{j_2})} \quad (1.14)$$

Три меры $D_{length}, D_{angle}, D_{area}$ оценивают разницу между $(i_1, i_2); (j_1, j_2); (V_1, V_2)$. Если две пары имеют одинаковые относительные позиции, то стоимость переноса примет минимальное значение ($Q(i_1, j_1, i_2, j_2) = 0$). Но если хотя бы одна из пар будет содержать фиктивный узел, то стоимость примет максимальное значение ($Q(i_1, j_1, i_2, j_2) = 1$).

Пример отношения между вершинами графов показан на Рисунок 1.13.

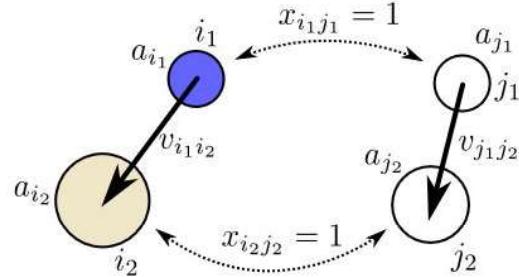


Рисунок 1.13 — Отношения между вершинами графов.

Базируясь на этой идеи, получаем что матрица X может быть получена путем минимизации суммы 1.15 с ограничением. λ_{min} - наименьшее собственное значение стоимости Q , I - единичная матрица.

$$\sum_{i_1, j_1, i_2, j_2} Q(i_1, j_1, i_2, j_2) x_{i_1, j_1} x_{i_2, j_2} = X^t Q_x \quad (1.15)$$

$$\min(X^t Q_x) \Rightarrow \min(X^t (Q_x - \lambda_{min} I)) x + \lambda_{min} \sum_{i,j=1}^N x_{i,j} : x \in \{0,1\}^{N^2} \quad (1.16)$$

1.5 Сравнительный анализ методов колоризации графических новелл.

Вышеописанные методы имеют свои преимущества и недостатки. Они сведены в таблицу ниже, где "Метод №1" это метод колоризации с использованием цветовых подсказок, а "Метод №2" это метод колоризации с использованием опорного изображения. Ниже приведены примеры работы "Метода №1" и "Метода №2". Так же представлен случай, описывающий недостатки колоризации вторым методом. (Рисунок 1.14)



Рисунок 1.14 — Пример работы метода колоризации с использованием цветовых подсказок.

[5]

Как можно видеть из Рисунок 1.14 данный метод, может колоризировать монохромное изображение также без ввода цветовых подсказок, дискриминатор лишь примет значение у генератора цветового распределения, которое уже было встречено в процессе обучения на наборе данных. Если же добавить цветовые подсказки, то результат изменится, но также увеличится и время работы метода, так как начнется ресурсозатратная гонка между генератором и дискриминатором, описанная выше и представленная на графике Рисунок 1.11.

Метод №1	Метод №2	Параметры сравнения
Загрузка целевого изображения.	Загрузка целевого изображения, опорного изображения.	Необходимость загрузки изображения
Ожидание ввода цветовых подсказок.	-	Ожидание пользовательского ввода
-	Требуется опорное изображение, содержащее колоризованные элементы целевого.	Загрузка опорного изображения
-	+ Хранение генерированных карт объектов, передача этих матриц между слоями сети Исправлены использование расстояния Вассерштейна.	Использование графовых структур Хранение матриц
-	Ресурсозатратный метод. Требует обучения моделей.	Не исправлены. (см Раздел 1.4.)
-	Медленный засчет счивания подсказок, построения цветового распределения и самого переноса соответствия с применением функции опибики.	Обработка графовых структур. Ресурсоемкий метод. Быстрее метода №1. Так как выставляется цветовое соответствие сразу, используя опорный граф и методы квадратичного программирования.
-	Построение цветового распределения и самого переноса соответствия с применением функции опибики.	Скорость работы предложенного метода

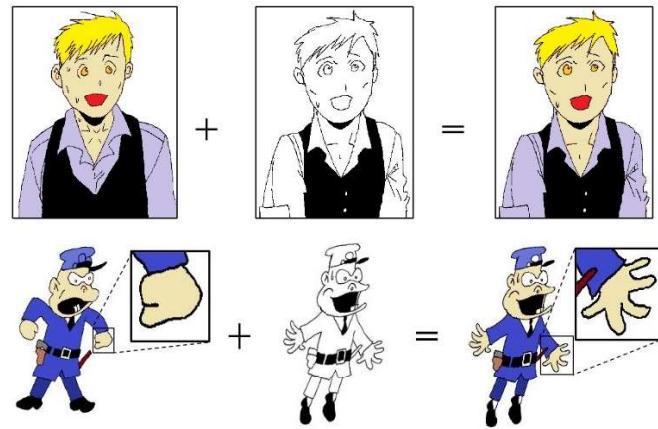


Рисунок 1.15 — Примеры работы метода колоризации с использованием опорного изображения.

[2]

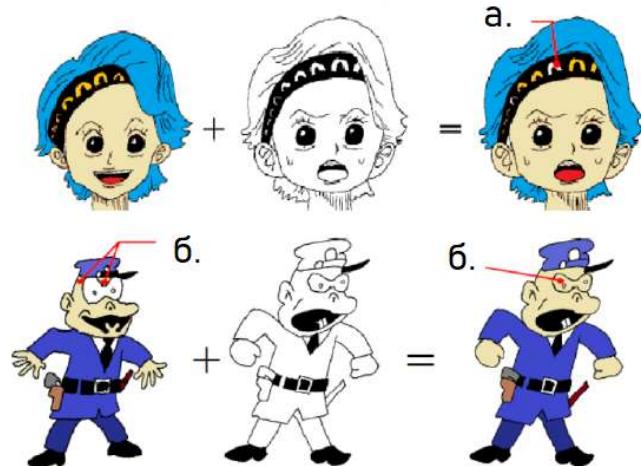


Рисунок 1.16 — Недостатки работы метода колоризации с использованием опорного изображения.

[2]

Как видно из примеров работы, представленных на Рисунок 1.15, данный способ колоризации справляется с решением поставленной задачи в случае, если число узлов графа опорного изображения совпадает с числом узлов графа, целевого.

На примерах из Рисунок 1.16 (а). видно, что в противном случае область, не проинициализированная узлом на графе опорного изображения, остается не закрашенной. Если же область была проинициализирована узлом на графе опорного изображения, а на графе целевого изображения отсутствует узел, отвечающий за соответствующую область, то происходит сильное искажение целевой области относительно используемой опорной, что приводит к неверному результату работы (Рисунок 1.16 (б.)).

2 Конструкторский раздел

Исходя из существующих методов, описанных в аналитическом разделе, можно сделать вывод о том, что оптимальное решение проблемы колоризации графических новелл будет содержать в себе комбинацию двух методов. Метод колоризации с использованием цветовых подсказок можно оптимизировать, используя вместо цветовых подсказок гистограмму распределения цветов, полученную на основе графа опорного изображения (граф получить, исходя из метода с использованием опорного изображения, графика и квадратичного программирования). То есть время работы модифицированного метода должно сократиться ввиду исключения или сведения к минимуму гонки генератора и дискриминатора. Гонка возникает на этапе работы с цветовыми подсказками.

Представим задачу разработки автоматизированного метода колоризации графических новелл на Рисунок 2.1 и Рисунок 2.2 в качестве "IDEF0" диаграммы.

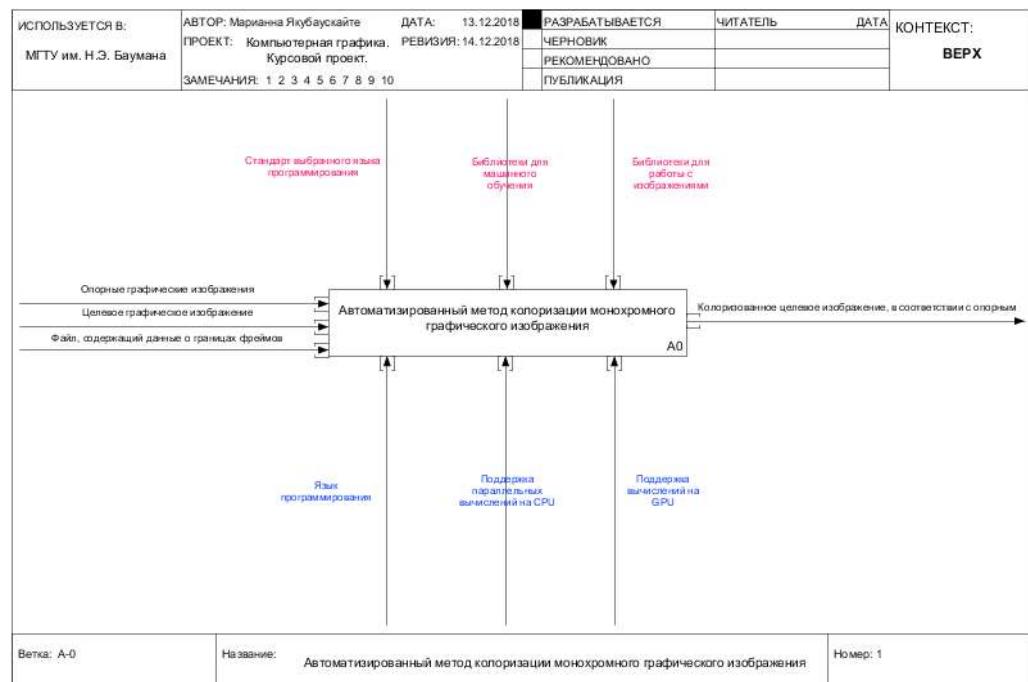


Рисунок 2.1 — IDEF0 диаграмма метода(Часть 1).

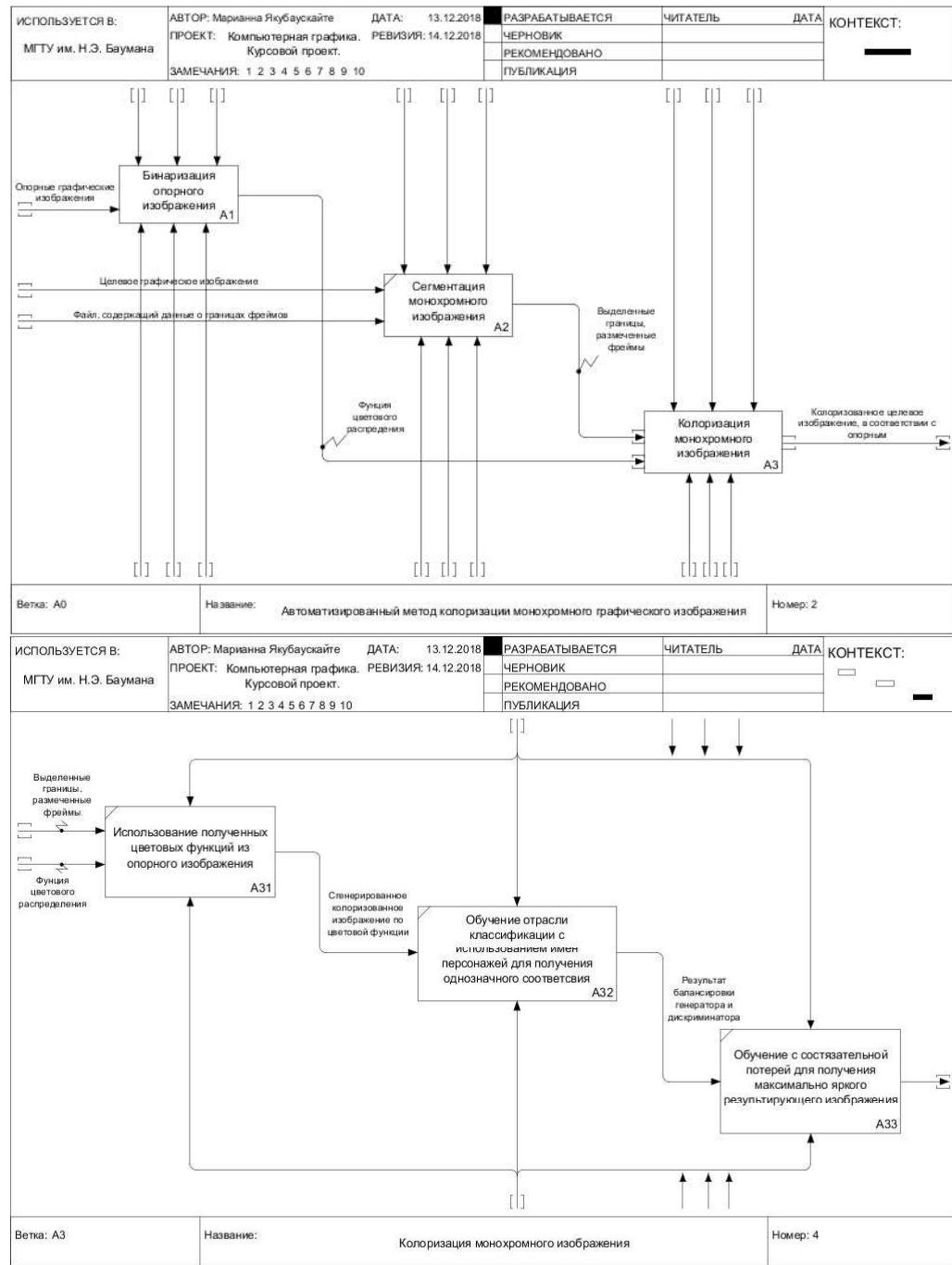


Рисунок 2.2 – IDEF0 диаграмма метода(Часть 2).

2.1 Описание общего алгоритма работы.

Общая модель работы представлена на Рисунок 2.3 и Рисунок 2.4.

На вход требуется подать монокромное изображение манги и опорные цветные изображения, причем количество опорных изображений должно удовлетворять условию-соответствию: один фрейм = одно опорное изображение. Далее проводится сегментация изображения по фреймам (координаты самих фреймов могут быть представлены на языке разметки, к примеру быть описаны в *.json файле). Каждому

фрейму проводится соответствие с его опорным изображением, из опорного изображения методом бинаризации извлекается цветовая гистограмма - называемая в частности, цветовой функцией и представляющая собой цветовую палитру, использованную для раскраски опорного изображения. Затем происходит колоризация монохромного сегмента в соответствии с цветовой гистограммой, наносятся поверх контуры и текст.

Архитектура модели нейронной сети позаимствована у метода колоризации с использованием цветовых подсказок - их модель поддерживает концепцию четырех сетей, представляющей слои одной единой сверточной нейронной сети:

- а) глобальная сеть функций;
- б) низкоуровневая сеть функций;
- в) сеть среднего уровня;
- г) сеть раскраски.

Основываясь на методе, описанном в вышеупомянутой работе, и используя метод переноса цветовой функции с опорного изображения на монохромное, в архитектуру будут добавлены следующие элементы:

- а) использование цветовой палитры от опорных изображений;
- б) обучение классификации с использованием имен персонажей (использование data-set "*manga 109*");
- в) обучение с учетом состязательной потери для получения визуально более ярких и насыщенных цветов.

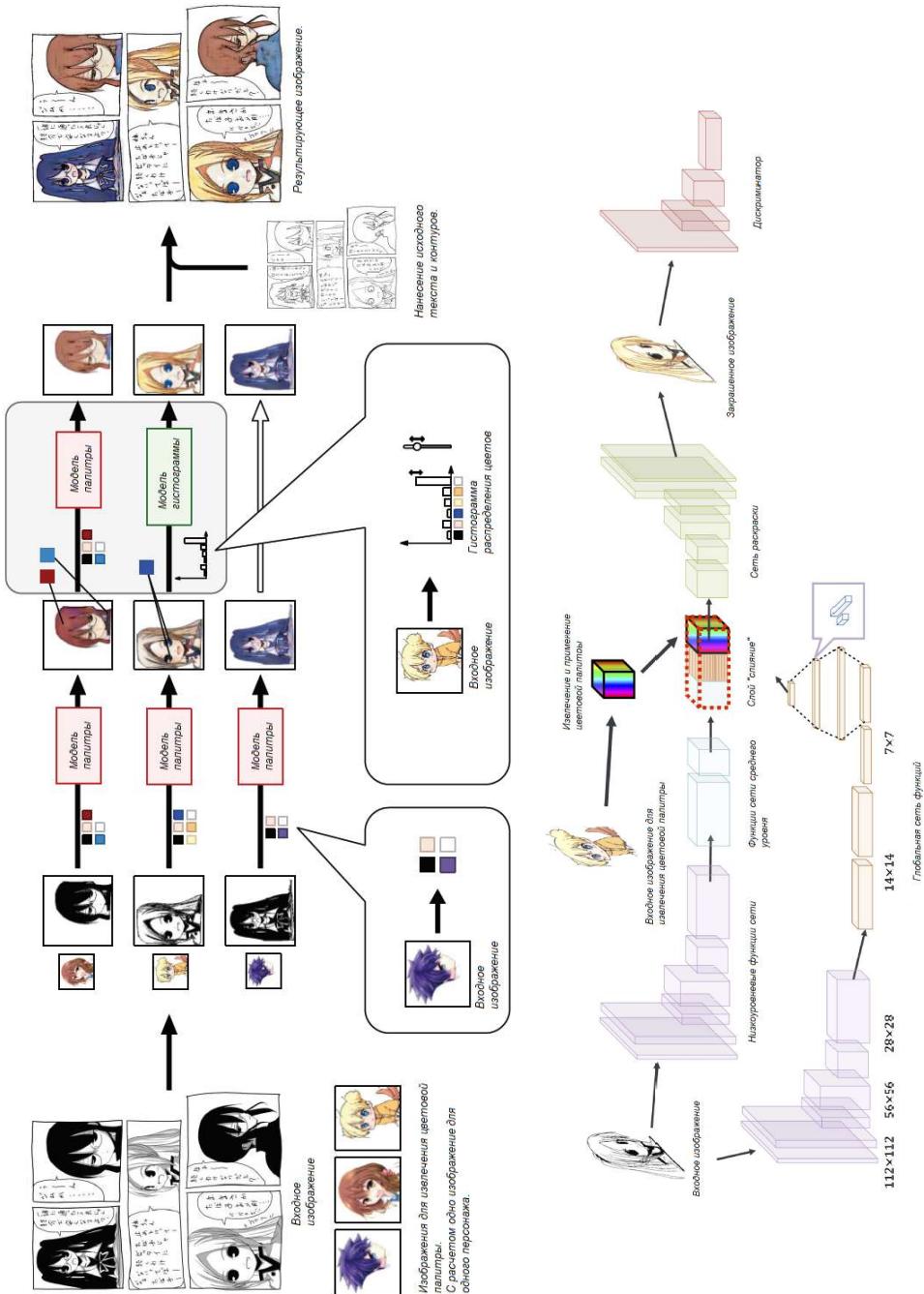


Рисунок 2.3 – Общая модель работы.

3 Технологический раздел

Разработка встраиваемой в графические редакторы библиотеки, с целью автоматизации методов закраски графических новелл, в частности, манги.

3.1 Выбор инструментов для разработки.

Так как создание библиотеки с целью автоматизации методов закраски графических новелл подразумевает применение методов статистики для анализа данных, работу с нейронными сетями, выбор инструмента разработки был остановлен на языке программирования Python 3.7. Python является высокоуровневым интерпритируемым и кросплатформенным языком программирования. Для глубокого машинного обучения использовалась библиотека Chainer, поддерживающая работу со сверточными нейросетями, а так же позволяющая использовать ускорение вычислений на GPU - с помощью набора инструментов от CUDA. Библиотека использует систему многоуровневых узлов, которая позволяет вам быстро настраивать, обучать и развертывать искусственные нейронные сети с большими наборами данных.

3.2 Процесс обучения.

Процесс обучения подразумевает собой подготовку data-set nico-opendata, для решения задачи колоризации), инициализацию функции потери, использующуюся на этапе классификации и подготовку data-set manga-109 для реализации соответствия между именем и персонажем.

4 Исследовательский раздел

Данный метод был протестирован на наборе данных manga 109. Данный набор данных содержит монохромные изображения манги, а также .json файлы, с описанием координат каждого фрейма. Опорные изображения были взяты из набора данных nico-opendata. Технические характеристики устройства таковы:

- a) процессор Intel Core i7-7700HQ с 16 ГБ оперативной памяти и 8 логическими ядрами;
- б) графический процессор NVIDIA GeForce GTX 1060Ti;
- в) дисковый SSD накопитель, имеющий среднюю скорость считывания - 520 МБ/с, а время доступа 5,61 мс;
- г) установленная операционная система Windows 10 Home 64-бит;
- д) пакет CUDA 9 поколения для ускорения вычислений на CPU-GPU.

4.1 Примеры работы разработанного метода.

При тестировании метода выявлены три типа случаев.

Лучший случай: целевое изображение не имеет мелких деталей, которые могут при сканировании U-Net сетью превратиться в незамкнутые области, каждому фрейму соответствует свое опорное изображение, в опорном изображении также четко отличаются границы и может быть получен однозначный результат для цветовой функции. (Рисунок 4.1)

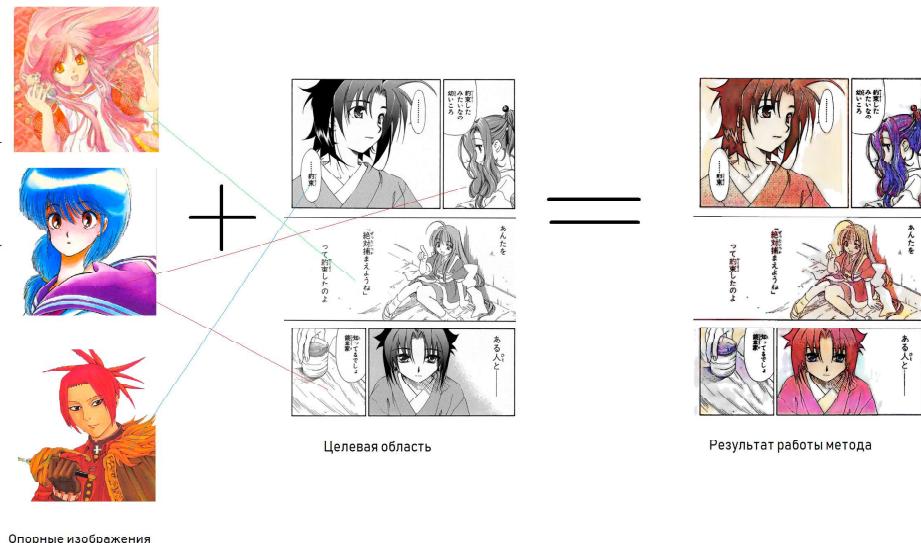


Рисунок 4.1 — Пример работы разработанного метода (лучший случай).

В стандартном случае: нет сильного изменения значения интенсивности (помимо черного и белого, используются различные оттенки серого) на целевом изображении.

жении, что приводит к "растеканию" цвета, в следствии нечеткого выделения границ U-Net сетью. (Рисунок 4.2)



Рисунок 4.2 — Пример работы разработанного метода (стандартный случай).

В худшем случае: монохромное целевое изображение содержит множество мелких деталей, которые после обработки U-Net сетью превращаются в незамкнутые области, из-за чего происходит "растекание" краски.

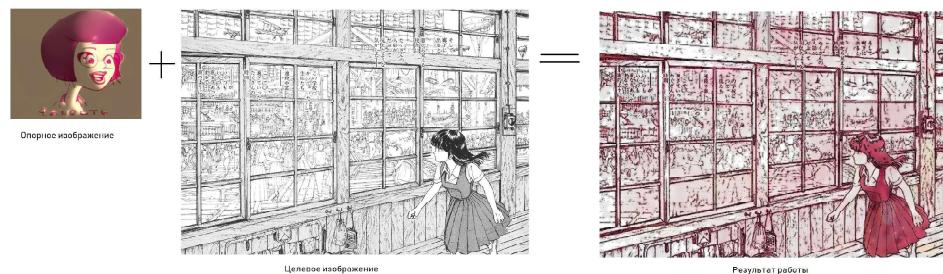


Рисунок 4.3 — Пример работы разработанного метода (худший случай).

4.2 Исследование времени работы.

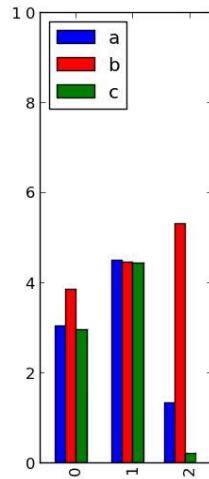


Рисунок 4.4 — Диаграмма сравнения времени работы методов.

На Рисунок 4.4 а - метод колоризации с использованием опорного изображения, б - метод колоризации с цветовыми подсказками, с - разработанный метод. Было проведено 3 тестовых случая на одном и том же изображении с разрешением 1280x720: 1 - соответствует опорному изображению из трех цветов и трем цветовым подсказкам, 2 - соответствует опорному изображению из шести цветов и шести цветовых подсказок, 3 - соответствует опорному изображению из двенадцати цветов и двенадцати цветовым подсказкам.

По диаграмме, представленной на Рисунок 4.4 видно, что поставленная гипотеза об ускорении метода колоризации, использующего цветовые подсказки методом, работающим с опорным изображением оправдалась. Полученный метод, действительно на том же наборе данных, при всех прочих равных условиях, на диаграмме зависимости времени от качества входного изображения (разрешение в пикселях) показал результаты, выше чем использованные за основу методы.

Данный метод так же поддерживает вычисления на GPU. Сравнение временных характеристик, полученных при колоризации изображений из набора данных *manga 109*, с использованием оптимизации на GPU и без. Результаты представлены на графике Рисунок 4.5. График показывает зависимость пройденных итераций (ось y) от времени в секундах.

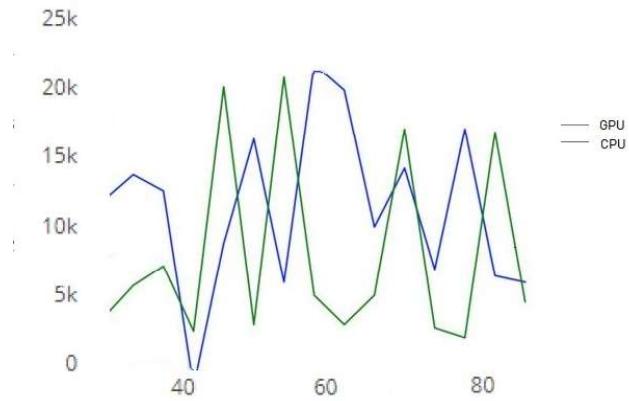


Рисунок 4.5 — График времени работы на GPU и CPU.

Заключение

В результате выполнения курсового проекта были получены следующие основные результаты:

- а) были изучены различные методы выделения границ изображения;
- б) были изучены методы колоризации графических изображений;
- в) выделены основные недостатки имеющихся методов;
- г) был разработан собственный метод колоризации графических новелл;
- д) было изучено глубинное машинное обучение, разобрана архитектура GAN и U-Net сетей, а также полученные знания были закреплены на практике;
- е) были изучены и применены на практике методы ускорения вычисления с помощью использования графических процессоров.