УДК 006.6

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АНИМИРОВАНИЯ ЛИЦА НА ОСНОВЕ КОМПЬЮТЕРНОГО ЗРЕНИЯ

Орозобекова А.К., Имангазиева Б.Б., Кубанычбекова А. К.
КГТУ им. И.Раззакова

В данной статье рассматривается модель движения на основе тонкой пластинки является мощным инструментом для задач трансформации изображений. Использование таких библиотек, как PyTorch и OpenCV, позволяет выполнять сложные трансформации с относительной легкостью. Предоставленные фрагменты кода дают представление о том, как реализована и обучена модель.

Ключевые слова: анимирование лица, компьютерное зрение, нейроные сети, PyTorch, нейронные сети

КОМПЬЮТЕРДИН КӨРҮНҮШҮНҮН НЕГИЗИНДЕ БЕТТЕРДИН АНИМАЦИЯСЫ ҮЧҮН ТИРКЕМЕЛЕРДИ ИШТӨӨ

Орозобекова А.К., Имангазиева Б.Б., Кубанычбекова А.К.
И.Раззакова атындагы КМТУ

Бул макалада сүрөттү трансформациялоо көйгөйлөрү үчүн күчтүү курал болгон жука пластинкага негизделген кыймыл модели талкууланат. PyTorch жана OpenCV сыяктуу китепканаларды колдонуу татаал трансформацияларды салыштырмалуу оңой аткарууга мүмкүндүк берет. Берилген код үзүндүлөрү моделдин кантип ишке ашырылып, үйрөтүлгөнү жөнүндө түшүнүк берет.

***Баштапкы сөздөр*:** беттин анимациясы, компьютердик көрүнүш, нейрон тармактары, PyTorch, нейрон тармактары

DEVELOPMENT OF AN APPLICATION FOR FACE ANIMATION BASED ON COMPUTER VISION

Orozobekova A.K., Imangazieva B.B., Kubanychbekova A.K.
KSTU named after. I.Razzakova

This paper discusses a thin plate-based motion model that is a powerful tool for image transformation problems. Using libraries such as PyTorch and OpenCV allows you to perform complex transformations with relative ease. The provided code snippets provide insight into how the model is implemented and trained.

Keywords: facial animation, computer vision, neural networks, PyTorch, neural networks

В последние годы технология компьютерного зрения и анимирования лиц привлекает все больше внимания как в научной, так и в коммерческой сфере. Разработка приложения для анимирования лица на основе компьютерного зрения является актуальной и перспективной задачей. Это направление открывает широкие возможности для применения в различных областях, улучшая качество взаимодействия между людьми и технологиями. Постоянное развитие технологий компьютерного зрения и искусственного интеллекта способствует созданию новых, более совершенных решений, которые найдут свое применение в повседневной жизни. Это обусловлено широкими возможностями применения таких технологий в различных областях, начиная от развлечений и заканчивая здравоохранением. Интерактивные и развлекательные приложения: С развитием социальных сетей и мессенджеров возрос спрос на интерактивные и персонализированные формы общения. Анимация лиц в реальном времени позволяет пользователям создавать уникальные эмодзи, стикеры и аватары, которые точно передают их эмоции и настроение.

Последним передовым шагом в анимации антропоморфных существ является технология захвата движений. Её суть заключается в переносе мимики, жестов и движений тела актёра на анимируемую модель. Ниже представлены шаги и направления исследования аналогов:

- анализ существующих приложений для анимации лица:
- исследование методов определения ключевых точек лица:
- обзор моделей деформаций лица:

- анализ инструментов и библиотек:
- изучение инструментов и библиотек, предназначенных для разработки приложений компьютерного зрения и анимации.

Для создания анимированного лица необходимо создать 3D-модель и подготовить её для анимации – создать риг и(или) блэнд шейпы. Анимирование лица вручную происходит следующим образом:

3D-модель – это совокупность вершин, рёбер и полигонов, которые определяют форму объекта. В случае создания анимированного лица моделируется голова с нейтральным выражением лица (рисунок 1.), затем для каждого выражения расположение вершин деформируется – деформированные объекты называются «блэнд шейпами» (рисунок 1.).

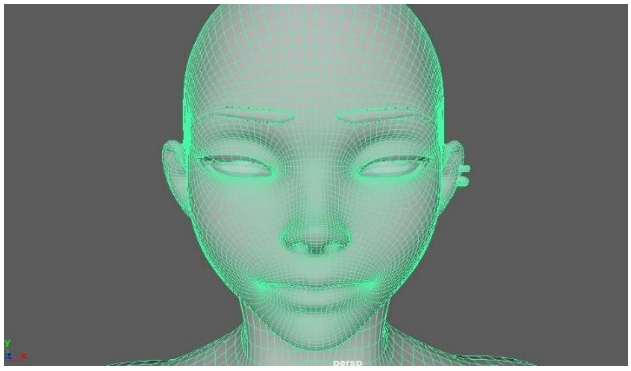

Рис. 1. Нейтральное выражение лица 3D-модели с показанной полигональной сеткой

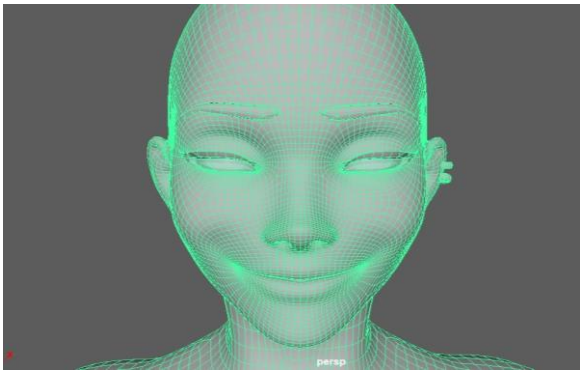

Рис. 2. – улыбающееся выражение лица с показанной полигональной сеткой

Обычно для создания анимации используется сочетание нескольких блэнд шейпов (рисунок 1.) с разными значениями, где значение – процент, определяющий на сколько процентов исходная модель соответствует блэнд шейпу. Это позволяет достичь выражения сложных эмоций лица.

Компьютерное зрение (Computer Vision, CV) — это область искусственного интеллекта, связанная с анализом изображений и видео. Она включает в себя набор методов, которые наделяют компьютер

способностью «видеть» и извлекать информацию из увиденного.

Системы состоят из фото- или видеокамеры и специализированного программного обеспечения, которое идентифицирует и классифицирует объекты. Они способны анализировать образы (фотографии, картинки, видео, штрих-коды), а также лица и эмоции. Рассмотрена модель движения на основе тонкой пластинки (Thin-Plate Spline Motion Model, TPS), которая является моделью машинного обучения, используемой для неригидной трансформации изображений [3]. Эта модель особенно полезна для приложений, таких как анимация изображений, где одно изображение может быть реалистично преобразовано в другое.

Модель TPS использует сетку контрольных точек, которые определяют трансформацию. Эти контрольные точки могут перемещаться для плавного и непрерывного деформирования изображения. Трансформация управляется набором математических уравнений, которые обеспечивают плавность деформации. Основная идея состоит в использовании функции сплайна, чтобы минимизировать изгиб изображения, сохраняя при этом точное соответствие контрольных точек. Использованы следующие инструменты и библиотеки:

1.**Python**:Основной язык программирования

2. **PyTorch**: Фреймворк для глубокого обучения, используемый для реализации компонентов нейронной сети модели.

3. **NumPy**: Библиотека для численных вычислений.

4. **OpenCV**: Библиотека для задач обработки изображений.

5. **Matplotlib**: Библиотека для визуализации результатов.

В библиотеках глубокого обучения есть механизмы вычисления градиента ошибки и обратного распространения ошибки через вычислительный граф. Этот механизм, называемый автоградиентом в PyTorch, легко доступен и интуитивно понятен. Переменный класс —

главный компонент автоградиентной системы в PyTorch. Переменный класс обертывает тензор и позволяет автоматически вычислять градиент на тензоре при вызове функции backward().

Объект содержит данные из тензора, градиент тензора (единожды посчитанный по отношению к некоторому другому значению, потеря) и содержит также ссылку на любую функцию, созданную переменной (если это функция, созданная пользователем, ссылка будет пустой).

Создадим переменную из простого тензора:

x = Variable (torch.ones(2, 2) * 2, requires_grad=True)

В объявлении переменной используется двойной тензор размера 2x2 и дополнительно указывается, что переменной необходим градиент. При использовании этой переменной в нейронных сетях, она становится способна к обучению. Если последний параметр будет равен False, то переменная не может использоваться для обучения.

Обзор кода

1. Установка и настройка

```python
# Клонирование репозитория
!git clone https://github.com/yoyo-nb/Thin-Plate-Spline-Motion-Model.git
# Переход в каталог
%cd Thin-Plate-Spline-Motion-Model
# Установка необходимых зависимостей
!pip install -r requirements.txt
```

2. Загрузка данных

```python
import os
import cv2
import numpy as np
from torch.utils.data import DataLoader, Dataset
class ImageDataset(Dataset):
  def __init__(self, root_dir, transform=None):
    self.root_dir = root_dir
    self.transform = transform
    self.image_paths = [os.path.join(root_dir, f) for f in os.listdir(root_dir) if
os.path.isfile(os.path.join(root_dir, f))]
  def __len__(self):
```

```
        return len(self.image_paths)
    def __getitem__(self, idx):
        image_path = self.image_paths[idx]
        image = cv2.imread(image_path)
        if self.transform:
            image = self.transform(image)
        return image
# Пример использования
dataset = ImageDataset(root_dir='path/to/images')
dataloader = DataLoader(dataset, batch_size=32, shuffle=True)
```

3. Определение модели

```python
import torch
import torch.nn as nn
class ThinPlateSplineModel(nn.Module):
    def __init__(self, control_points):
        super(ThinPlateSplineModel, self).__init__()
        self.control_points = control_points
        self.fc = nn.Linear(len(control_points), 2 * len(control_points))
    def forward(self, x):
        transformed_points = self.fc(x)
        return transformed_points
# Пример использования
control_points = [(0, 0), (1, 0), (0, 1), (1, 1)]
model = ThinPlateSplineModel(control_points)
```

4. Обучение модели

```python
import torch.optim as optim
def train(model, dataloader, epochs=10):
    criterion = nn.MSELoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)
    for epoch in range(epochs):
        for images in dataloader:
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, images)
            loss.backward()
            optimizer.step()
        print(f'Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.4f}')
# Пример использования
train(model, dataloader)
```

Объяснение компонентов модели:

Модель TPS основывается на использовании сплайнов для деформации изображения. Сплайны тонкой пластинки минимизируют функционал изгиба, что позволяет создать гладкие и реалистичные деформации. Основные шаги включают:

1. **Определение контрольных точек**: Контрольные точки выбираются на исходном изображении и перемещаются для создания новой формы.
2. **Создание матрицы сплайна**: С помощью контрольных точек создается матрица сплайна, которая определяет, как будут перемещаться остальные точки изображения.
3. **Обучение модели**: Модель обучается с использованием этих матриц и целевых изображений, минимизируя ошибку между предсказанным и целевым изображением.

Модель движения на основе тонкой пластинки является мощным инструментом для задач трансформации изображений. Использование таких библиотек, как PyTorch и OpenCV, позволяет выполнять сложные трансформации с относительной легкостью. Предоставленные фрагменты кода дают представление о том, как реализована и обучена модель.

Рис 1. Фото для аннимирования

Рис 2. Анимирование лица

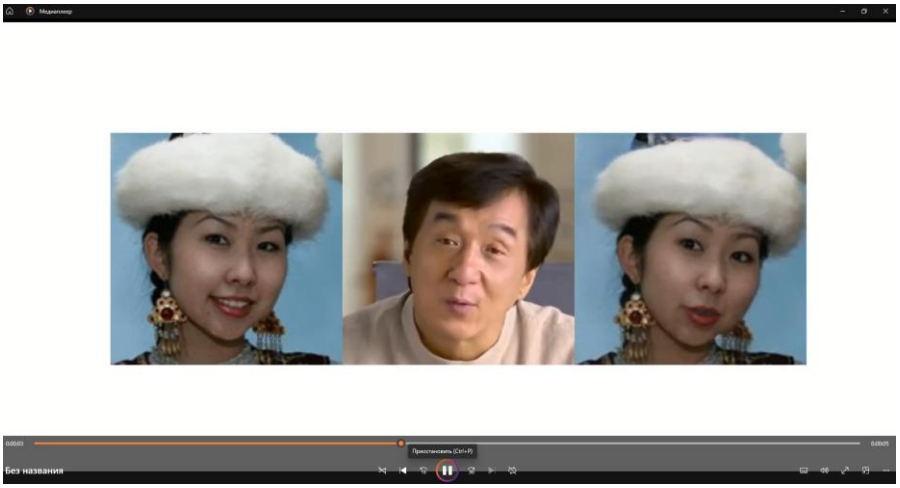

Рис. 3. Анимирование лица

Для анимирования лица с использованием компьютерного зрения на языке программирования Python, можно воспользоваться несколькими библиотеками и инструментами. Вот некоторые из них:

1. **OpenCV**: Это одна из самых популярных библиотек для компьютерного зрения. Она предоставляет множество инструментов для обработки изображений и видео, а также для распознавания лиц.
2. **Dlib**: Эта библиотека включает в себя алгоритмы машинного обучения для распознавания лиц и определения ключевых точек (landmarks) на лице.
3. **MediaPipe**: Эта библиотека от Google используется для отслеживания рук, лица и других объектов в реальном времени.
4. **DeepFace:** Это библиотека, которая предоставляет различные модели для распознавания лиц и анализа эмоций.
5. **Facial Animation** (FAN): Это специализированные модели для анимации лиц на основе ключевых точек и других данных.

Пример использования OpenCV и Dlib для определения ключевых точек лица:

1. Установите необходимые библиотеки:

```
bash
pip install opencv-python dlib
```

2. Пример кода для определения ключевых точек лица:

```
import cv2
import dlib
# Загрузка предобученной модели для определения лиц
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
# Загрузка изображения
img = cv2.imread("path_to_image.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Обнаружение лиц на изображении
faces = detector(gray)
for face in faces:
    landmarks = predictor(gray, face)
    for n in range(0, 68):
        x = landmarks.part(n).x
        y = landmarks.part(n).y
        cv2.circle(img, (x, y), 4, (255, 0, 0), -1)
# Показ изображения с ключевыми точками
cv2.imshow("Face Landmarks", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

3. Для анимации лица можно использовать найденные ключевые точки. Например, можно перемещать точки или изменять их положение для создания эффектов, таких как моргание или улыбка.

Пример анимации:

```
python
import numpy as np
# Функция для изменения положения ключевых точек
def animate_landmarks(landmarks):
    animated_landmarks = []
    for n in range(0, 68):
        x = landmarks.part(n).x
        y = landmarks.part(n).y
        # Добавление случайного шума для анимации
```

```
        x += np.random.randint(-2, 2)
        y += np.random.randint(-2, 2)
        animated_landmarks.append((x, y))
    return animated_landmarks
# Обработка и анимация лиц
for face in faces:
    landmarks = predictor(gray, face)
    animated_landmarks = animate_landmarks(landmarks)
    for (x, y) in animated_landmarks:
        cv2.circle(img, (x, y), 4, (0, 255, 0), -1)
cv2.imshow("Animated Face Landmarks", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Для более сложной анимации можно использовать более продвинутые методы, такие как обучение нейронных сетей для генерации лицевых выражений или использование библиотек вроде MediaPipe для отслеживания движений в реальном времени.

NumPy является мощной библиотекой для численных вычислений в Python и может быть использована для различных математических операций, необходимых для анимации лица. Вместе с библиотеками OpenCV и Dlib, NumPy может быть использована для обработки изображений и выполнения операций с массивами, что часто требуется при работе с ключевыми точками лица и их анимацией.

Рассмотрим пример, в котором будем использовать NumPy для создания простейшей анимации лица, изменяя ключевые точки, чтобы создать эффект моргания.

В результате последовательного выполнения действий, описанных выше, получается приложение, которое:

1. Принимает на вход набор кадров с выражениями эмоций актёра, соответствующим блэнд шейпам у 3D-модели, нейтральным выражением лица и кадр, значения эмоций которого необходимо определить.
2. Распознает лицо на каждом изображении и его ключевые точки.
3. Рассчитывает их смещение относительно нейтрального выражения

лица и на основе этих расчетов делается вывод, какой блэнд шейп на кадре задействован и на сколько процентов . разработан алгоритм определения значения эмоции на найденном изображении лица, а также разработано приложение для анимирования на его основе.

4. Проведено исследование эффективности приложения, базирующегося на разработанном выше алгоритме – разработанное приложение позволяет сократить время, отведенное на разработку анимации лица на 30% без видимой потери качества разработанной анимации.

В результате проделанной работы создан алгоритм анимирования лица и приложение на его основе. Исследование эффективности работы полученного приложения показало, что работая с ним аниматор тратит на 30% меньше времени на разработку лицевой анимации без потери её качества. Наличие публикаций и выступлений на конференциях по теме выпускной работы.

ЛИТЕРАТУРА

1. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python: руководство для специалистов по работе с данными / Мюллер Андреас, Гвидо Сара. – Санкт-Петербург, 2018
2. One millisecond face alignment with an ensemble of regression trees [Электронный ресурс] URL: https://pdfs.semanticscholar.org/d78b/6a5b0dcaa81b1faea5fb0000045a62513567. pdf (дата обращения: 17.03.2019)
3. Social media statistics [Электронный ресурс] URL: https://chrissniderdesign.com/blog/resources/social-media-statistics/ (дата обращения: 12.05.2019)
4. Helen dataset [Электронный ресурс] URL: http://www.ifp.illinois.edu/~vuongle2/helen/ (дата обращения:

03.04.2019)
5. Facial point annotations [Электронный ресурс] URL: https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/ (дата обращения: 03.03.2019)
6. Николенко С., Кадурин А., Архангельская Е. ГЛУБОКОЕ ОБУЧЕНИЕ погружение в мир нейронных сетей / Николенко С., Кадурин А., Архангельская Е. – Санкт-Петербург, 2018
7. https://dzen.ru/a/ZPrftS3_5l2Y5cmj
8. https://trends.rbc.ru/trends/industry/5f1f007e9a794756fafbfa83