Буценко Максим Сергеевич
студент
Дозоренко Михаил Сергеевич
студент
Никитин Давид Дмитриевич
студент
Трандасир Артем Сергеевич
студент
Научный руководитель
Куликова Наталья Николаевна
канд. биол. наук, доцент
ФГБОУ ВО «Кубанский государственный университет»
г. Краснодар, Краснодарский край

УЧЕБНЫЙ ТРЕНАЖЁР НА ПЛАТФОРМЕ UNITY С CV-МАСКОЙ ЗАХВАТА ЦЕЛИ И АВТОМАТИЧЕСКИМ РЕАГИРОВАНИЕМ

Аннотация: *описываются архитектура и программная реализация учебного тренажёра на платформе Unity, в котором модуль компьютерного зрения формирует маску захвата динамичной цели, а подсистема реагирования обрабатывает сформированные события без участия пользователя. Решение построено на связке встроенного движка Unity и компонента инференса нейросетевых моделей: сегментация объекта выполняется в реальном времени, а сценарии реакций запускаются автоматически. Приводятся схемы архитектуры тренажёра, пайплайн генерации маски и фрагменты кода ключевых компонентов. Тренажёр предназначен для магистерских курсов прикладной разработки и компьютерного зрения.*

Ключевые слова: *учебный тренажёр, Unity, компьютерное зрение, сегментация изображений, маска захвата цели, динамичная цель, автоматическое реагирование, инференс нейросетевых моделей.*

Введение.

Content is licensed under the Creative Commons Attribution 4.0 license (CC-BY 4.0)

Учебные тренажёры – распространённый формат цифровых образовательных ресурсов в прикладной подготовке инженеров. Средства компьютерного зрения и инструменты инференса нейросетевых моделей, встраиваемые непосредственно в среду разработки, существенно расширили круг задач, реализуемых в этой форме. Появилась возможность строить тренажёры, в которых обратная связь обучающемуся формируется по результатам реальной сегментации изображения, а не по заранее размеченной сцене; учебная задача при этом оказывается ближе к производственной [1; 2].

Ниже описывается реализация учебного тренажёра на платформе Unity, в котором модуль компьютерного зрения формирует маску захвата динамичной цели, а подсистема реагирования самостоятельно обрабатывает события, связанные с появлением цели в кадре. Архитектура ориентирована на магистерские курсы прикладной разработки.

Архитектура тренажёра.

Тренажёр построен по событийной схеме. Его компоненты сгруппированы в три блока – захват кадра, компьютерное зрение и реагирование, – а координацию между ними осуществляет отдельный управляющий объект, синхронизирующий тактирование кадра. Общая архитектура показана на рис. 1.

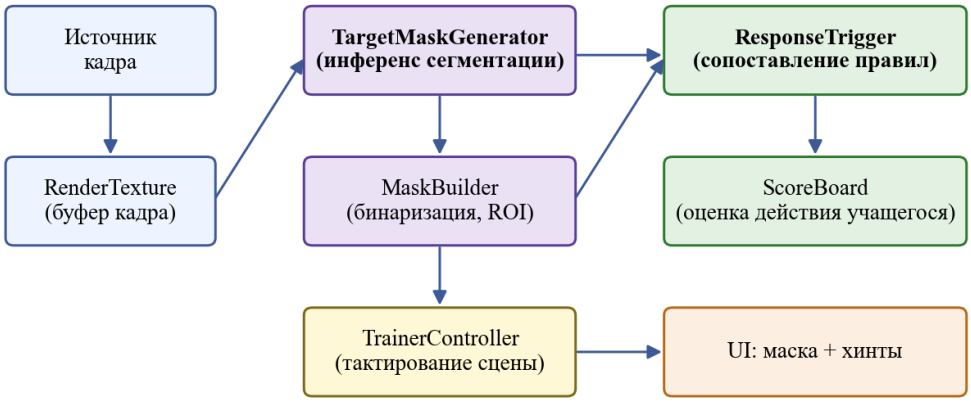

Рис. 1. Архитектура учебного тренажёра

Содержимое доступно по лицензии Creative Commons Attribution 4.0 license (CC-BY 4.0)

Источник кадра предоставляет видеоданные в RenderTexture – стандартный для Unity буфер, доступный как графическому конвейеру, так и компонентам компьютерного зрения. Использование RenderTexture в качестве промежуточного буфера позволяет работать как с реальной камерой, так и с заранее подготовленными видеофрагментами, не изменяя остальную часть архитектуры [3].

Модуль захвата маски.

Модуль захвата маски реализован как компонент TargetMaskGenerator, выполняющий инференс сегментационной модели на текущем кадре. В качестве бэкенда инференса используется Unity Sentis – компонент, входящий в состав платформы и обеспечивающий выполнение нейросетевых моделей на GPU [4]. Пайплайн обработки кадра приведён на рис. 2.

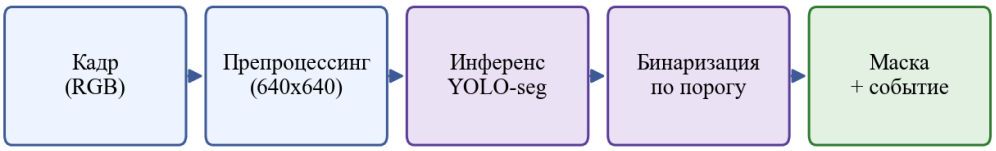

Рис. 2. Пайплайн генерации маски целевого объекта

Кадр приводится к фиксированному размеру входа модели (640×640) и передаётся на инференс, после чего тензор сегментационных масок бинаризуется по заданному порогу уверенности. Результирующая бинарная маска оборачивается в служебную структуру MaskFrame, содержащую массив маски, оценку уверенности и временную метку кадра.

Реализация компонента TargetMaskGenerator приведена на рис. 3.

Content is licensed under the Creative Commons Attribution 4.0 license (CC-BY 4.0)

```
using System;
using Unity.Sentis;
using UnityEngine;

namespace Trainer.Vision
{
    /// <summary>
    /// Запускает инференс сегментационной модели на текстуре
    /// видеокадра и формирует бинарную маску целевого объекта.
    /// </summary>
    public sealed class TargetMaskGenerator : MonoBehaviour
    {
        [SerializeField] private ModelAsset segmentationModel;
        [SerializeField] private RenderTexture cameraFeed;
        [SerializeField] private int inputSize = 640;
        [SerializeField, Range(0f, 1f)] private float confidence = 0.55f;

        private IWorker worker;
        private TensorFloat inputTensor;

        public event Action<MaskFrame> OnMaskReady;

        private void Awake()
        {
            var model = ModelLoader.Load(segmentationModel);
            worker = WorkerFactory.CreateWorker(BackendType.GPUCompute, model);
        }

        public void ProcessFrame()
        {
            inputTensor = TextureConverter.ToTensor(
                cameraFeed, inputSize, inputSize, 3);
            worker.Execute(inputTensor);

            using var output = worker.PeekOutput("masks") as TensorFloat;
            var mask = MaskBuilder.FromTensor(output, confidence);
            OnMaskReady?.Invoke(mask);
        }

        private void OnDestroy()
        {
            worker?.Dispose();
            inputTensor?.Dispose();
        }
    }
}
```

Рис. 3. Класс TargetMaskGenerator

Ключевыми элементами реализации являются хранение собственного экземпляра IWorker (исполнителя инференса), освобождение ресурсов в методе OnDestroy и публикация маски через событие OnMaskReady, на которое подписывается подсистема реагирования. Благодаря этому модуль компьютерного зрения изолирован от прикладного слоя и допускает повторное использование.

Подсистема автоматического реагирования.

Подсистема реагирования реализована как компонент ResponseTrigger, подписывающийся на события маски и сопоставляющий каждую новую маску с набором правил реагирования. Каждое правило (ResponseRule) описывается отдельной конфигурацией и содержит признаки маски, при которых правило срабатывает, а также действие, выполняемое при срабатывании [5]. Реализация компонента приведена на рис. 4.

Содержимое доступно по лицензии Creative Commons Attribution 4.0 license (CC-BY 4.0)

```
using System.Collections.Generic;
using UnityEngine;

namespace Trainer.Response
{
    /// <summary>
    /// Принимает события CV-маски и автоматически активирует
    /// настроенные реакции тренажёра (звук, эффект, оценка действия).
    /// </summary>
    public sealed class ResponseTrigger : MonoBehaviour
    {
        [SerializeField] private TargetMaskGenerator maskSource;
        [SerializeField] private List<ResponseRule> rules;

        private readonly Queue<MaskFrame> buffer = new(capacity: 16);

        private void OnEnable()  => maskSource.OnMaskReady += HandleMask;
        private void OnDisable() => maskSource.OnMaskReady -= HandleMask;

        private void HandleMask(MaskFrame frame)
        {
            buffer.Enqueue(frame);
            if (buffer.Count > 16) buffer.Dequeue();

            foreach (var rule in rules)
            {
                if (!rule.Matches(frame, buffer)) continue;
                rule.Invoke(frame);
                ScoreBoard.Register(rule.Id, frame.Timestamp);
            }
        }
    }
}
```

Рис. 4. Класс ResponseTrigger

Буфер последних кадров используется для правил, требующих временного контекста (например, удержание цели в кадре на протяжении нескольких кадров). При срабатывании правила выполняется привязанное к нему действие, а результат регистрируется в подсистеме оценки действий обучающегося (ScoreBoard).

Координация компонентов.

Тактирование сцены обеспечивает компонент TrainerController. Он отвечает за вызов метода TargetMaskGenerator.ProcessFrame() с заданной частотой и обеспечивает независимость частоты обработки кадров от частоты кадров рендеринга. Это особенно важно при сравнительно ресурсоёмком инференсе сегментационной модели, когда обновление маски целесообразно выполнять реже, чем обновление сцены. Реализация компонента показана на рис. 5.

Content is licensed under the Creative Commons Attribution 4.0 license (CC-BY 4.0)

```
using UnityEngine;

namespace Trainer.Core
{
    /// <summary>
    /// Координатор учебной сцены: запускает захват кадра,
    /// инференс CV-модели и систему автоматического реагирования
    /// с фиксированной частотой.
    /// </summary>
    [DefaultExecutionOrder(-100)]
    public sealed class TrainerController : MonoBehaviour
    {
        [SerializeField] private TargetMaskGenerator maskGenerator;
        [SerializeField] private ResponseTrigger trigger;
        [SerializeField, Range(5, 60)] private int targetFps = 30;

        private float accumulator;

        private void Update()
        {
            accumulator += Time.unscaledDeltaTime;
            float frameStep = 1f / targetFps;
            if (accumulator < frameStep) return;

            accumulator = 0f;
            maskGenerator.ProcessFrame();
        }
    }
}
```

Рис. 5. Класс TrainerController

Внешний вид тренажёра.

Внешний вид сцены тренажёра в режиме исполнения показан на рис. 6. В кадре расположены статичные элементы окружения (конвейер, вспомогательные объекты) и целевой объект, поверх которого сформированы маска захвата, ограничивающий прямоугольник и значение уверенности. В верхней части экрана выводятся информационные блоки: STATUS – текущее состояние маски и сработавшее правило реагирования; SCORE – статистика сработавших и пропущенных правил за сессию.

Рис. 6. Сцена тренажёра в режиме исполнения

Содержимое доступно по лицензии Creative Commons Attribution 4.0 license (CC-BY 4.0)

Заключение.

Представленный тренажёр показывает, что событийно-ориентированный образовательный ресурс может опираться на компьютерное зрение и автоматические сценарии реагирования как на источник обратной связи. Модуль компьютерного зрения отделён от прикладного слоя, благодаря чему модель сегментации и набор правил реагирования могут заменяться независимо друг от друга.

Тренажёр применим в магистерских курсах прикладной разработки и компьютерного зрения, а также в составе сквозного учебного проекта, объединяющего практики DevOps, GameDev и компьютерного зрения.

Список литературы

1. Селевко Г.К. Современные образовательные технологии: учебное пособие / Г.К. Селевко. – М.: Народное образование, 2019. – 256 с.

2. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс; пер. с англ. – 4-е изд. – М.: Техносфера, 2020. – 1104 с.

3. RenderTexture / Unity Technologies. – URL: https://docs.unity3d.com/Manual/class-RenderTexture.html (дата обращения: 12.05.2026).

4. Sentis: neural network inference / Unity Technologies. – URL: https://docs.unity3d.com/Packages/com.unity.sentis@latest (дата обращения: 12.05.2026).

5. Гамма Э. Приёмы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес; пер. с англ. – СПб.: Питер, 2020. – 368 с.

Content is licensed under the Creative Commons Attribution 4.0 license (CC-BY 4.0)