ДОДАТОК А

Слайди презентації

ДОСЛІДЖЕННЯ МЕТОДІВ ОБРОБКИ ТА ПРОЕЦІЮВАННЯ WEB-КОНТЕНТУ З ВИКОРИСТАННЯМ ЗАСОБІВ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

Виконалв:
ст.гр. ІПЗм-17-1 Брижниченко Андрій

Керівник:
к.т.н., доцент Лєсна Н.С.

1

Мета роботи

Дослідження засобів та методів реалізації доповненої реальності, виявлення можливих засобів та алгоритмів проеціювання динамічного веб-контенту.

Особлива увага в дослідженні приділяється використанню та оптимізації мобільними пристроями нейронних мереж для визначення маркерів для проеціювання, а також методи конвертації різних видів контенту до сприйнятливого засобами AR виду.

2

Постановка задачі

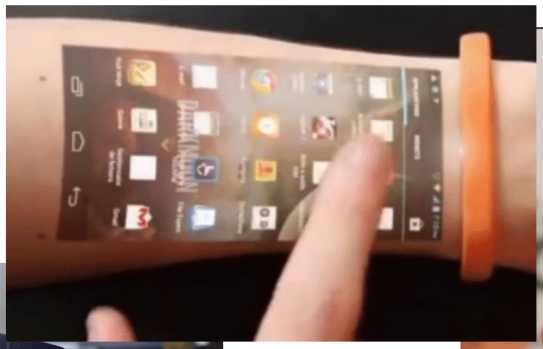
Дослідження засобів та методів реалізації доповненої реальності, виявлення можливих засобів та алгоритмів проеціювання динамічного веб-контенту, обрання оптимальних для вирішення задачі та проектування системи, що здатна до проеціювання веб-контенту на різних маркерах.

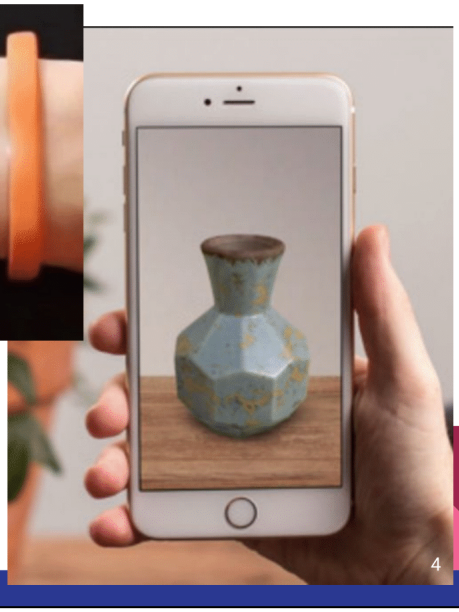
Досліджувані методи:

1. Доповнена реальність на основі маркера.
2. Markerless доповнена реальність.
3. Доповнена реальність на основі проекції.
4. Доповнена реальність на основі накладання

3

Існуючі засоби реалізації доповненої реальності





4

Інструменти реалізації доповненої реальності для мобільних пристроїв

-ARToolKit

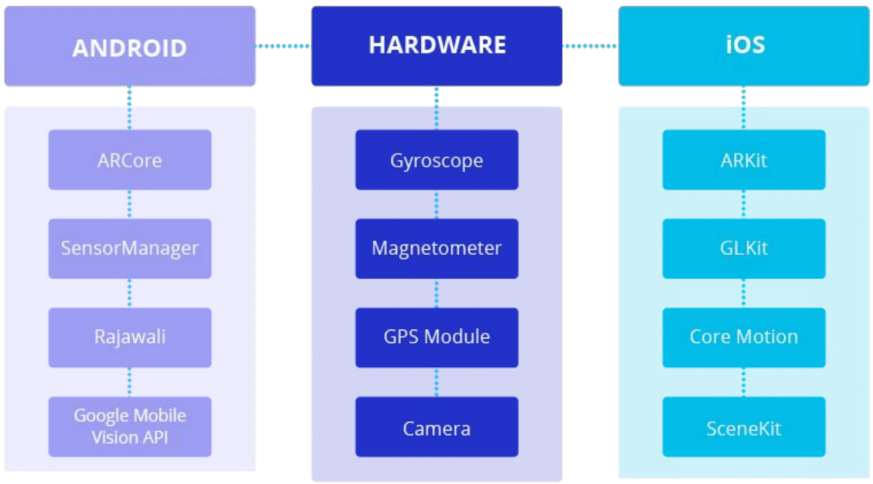
-Vuforia

-Wikitude

-ARCore

-ARKit

ARKit vs ARCore



Проблема гнучкості

Основні компоненти AR у мобільних застосунках:

1. Машинний зір (Vision)
2. Контент (SceneKit/SpriteKit)

Проблематика:

Ці компоненти вшиваються у пакет програми та для їх зміни необхідна повна перекомпіляція застосунку. Окрема проблема відображення відео та html контенту через непристосованість нативних засобів webView до ar компонентів

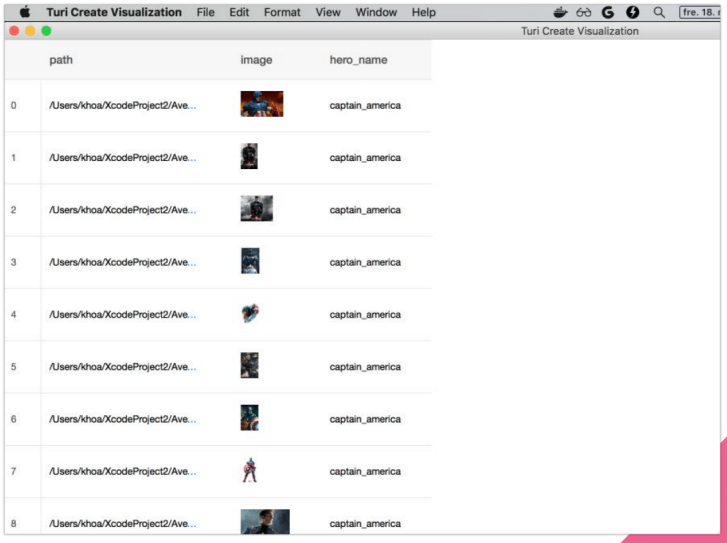
7

Вирішення проблем:

- Використання засобів комп'ютерного зору разом із нейронними мережами (Vision + CoreML)
- Шар конвертації контену:
  - 3D контенту зі сторонніх моделей шляхом декомпозиції сцени та отримання окремих компонентів
  - Шляхом перетворення 2D об'єктів у текстури 3D площин
  - Імплементація нащадка webView що буде здатен оновлюватися з певною періодичність та отримувати дії користувача через окремий об'єкт користувацької взаємодій

8

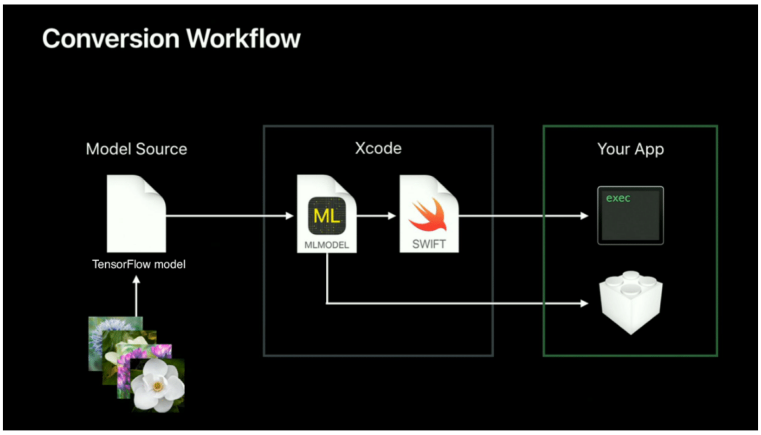
Turi Create



9

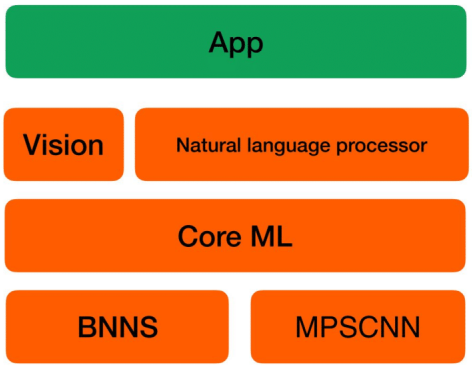
CoreML

– TensorFlow
– TuriCreate
– Keras
– Kaffe
– scikit-learn
– XGBoost
– libsvm



10

Vision + CoreML



Задача полягає у використанні отриманої моделі разом з CoreML.
Для цього є вибір між
−BNNS (Basic Neural Network Subroutines)
– MPSCNN (Metal Performance Shaders).

```
neural network with Metal... Elapsed time: 0.09986683333409019 sec
neural network with BNNS... Elapsed time: 0.03813095833356783 sec
inference with Metal... Elapsed time: 0.5680807083335822 sec
inference with BNNS... Elapsed time: 0.5057908750004572 sec
```

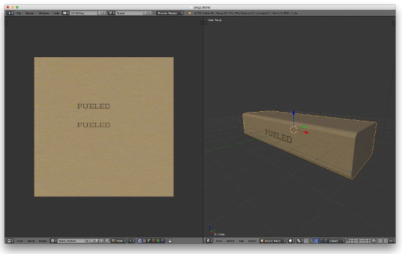
API	BNNS	MPSCNN
Компонент обробки	На основі ЦП	На основі GPU
Мова розробки	API на основі C	Адаптований для Swift
Ключова властивість	Швидка робота	Дозволяє створити власні функції активації

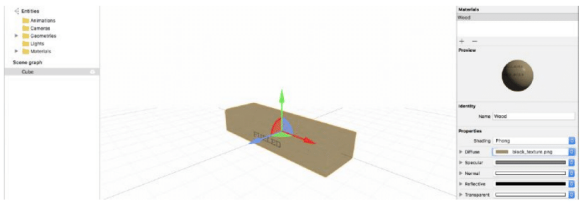
11

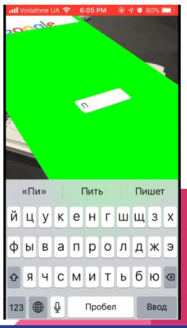
Vision + CoreML (програмний ланцюг)

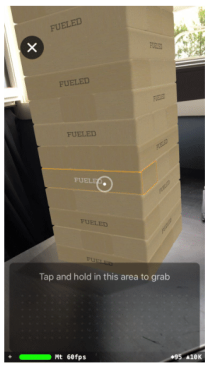
12

SceneKit Transformations









3D scene -> SCNSceneSource -> SceneKit Scene -> AR

2D scene -> UIView -> SceneKit Scene -> AR

html -> ARWebView -> SceneKit Scene -> AR

13

Висновки

Існують численні методи проеціювання контенту методами доповненої реальності, проте більшість з них або застарілі, або недоступні для широкого користування, до того ж більшість з них або зовсім не дають змоги постачання динамічного веб контенту, або лише в обмеженому вигляді.

В процесі роботи було розроблено алгоритм та модель системи, що здатна змінювати маркери проеціювання на основі тренованих мереж, що підключаються до застосунку, а також реалізовані методи конвертації різних типів веб-контенту у потрібний AR компоненту вигляд.

14

ДОДАТОК Б

ЛІСТИНГ КОДУ ПРОГРАМИ

Classifier_resnet.py

```
    import turicreate as turi

    datasetId = 0
    def __init__(self, id):
            self.datasetId = id
    url = FileManager.urlFor(self.datasetId)
    data = turi.image_analysis.load_images(url)
    data["type"] = data["path"].apply(lambda path: "Type1" if "type2" in path
else "Types")
    data.save("type1orType2.sframe")
    data.explore()
    dataBuffer = turi.SFrame("type1orType2.sframe")
    trainingBuffers, testingBuffers = dataBuffer.random_split(0.9)
    model = turi.image_classifier.create(trainingBuffers, target="type",
model="resnet-50")
    evaluations = model.evaluate(testingBuffers)

    model.save("TypeClassifier.model")
    model.export_coreml("TypeClassifier.mlmodel")
```

VisionClassifier+CoreML.swift

```
    import Foundation
    import CoreML
    import Vision

    var classificationRequest: VNCoreMLRequest

    func loadModel() -> VNCoreMLRequest {
    let model = try VNCoreMLModel(for: MobileNet().model)

    let request = VNCoreMLRequest(model: model, completionHandler: { [weak
self] request, error in
        self?.processClassifications(for: request, error: error)
    })
    request.imageCropAndScaleOption = .centerCrop
    self.classificationRequest = request
    return request
    }

    func initiateHandler(){
    DispatchQueue.global(qos: .userInitiated).async {
        let handler = VNImageRequestHandler(ciImage: ciImage, orientation:
orientation)
        do {
            try handler.perform([self.classificationRequest])
```

```
        } catch {
            print("Failed to perform
classification.\n\(error.localizedDescription)")
        }
    }
    }

    func processClassifications(for request: VNRequest, error: Error?) {
        DispatchQueue.main.async {
            guard let results = request.results else {
                self.classificationLabel.text = "Unable to classify
image.\n\(error!.localizedDescription)"
                return
            }
                let classifications = results as!
[VNClassificationObservation]
    }
    }
```

ContentConverter3D.swift

```
    class func loadObjectNamed(_ objectName: String, fromSceneNamed sceneName:
String) -> CAAnimation? {
            let url = FileManager(urlForResource: sceneName, ofType: "DAE")
            #if os(OSX)
            let options: [SCNSceneSource.LoadingOption: Any] =
[SCNSceneSource.LoadingOption.convertToYUp: true]
            #else
            let options: [SCNSceneSource.LoadingOption: Any] = [:]
            #endif
            let sceneSource = SCNSceneSource(url: url, options: options)
            let animation = sceneSource?.entryWithIdentifier(animationName,
withClass: CAAnimation.self)
         animation?.fadeInDuration = 0.3
            animation?.fadeOutDuration = 0.3
            return animation
    }
```

ARViewController.swift

```
    class ARViewController: UIViewController, ARSCNViewDelegate {

       @IBOutlet var sceneView: ARSCNView!

       var nodeModel:SCNNode!
       let nodeName = ""

       override func viewDidLoad() {
            super.viewDidLoad()
            sceneView.delegate = self
```

```
            sceneView.showsStatistics = true
            sceneView.antialiasingMode = .multisampling4X

             let scene = SCNScene()
            sceneView.scene = scene

            let modelScene = SCNScene(named:
            "art.scnassets/cherub/cherub.dae")!

            nodeModel =  modelScene.rootNode.childNode(withName: nodeName,
recursively: true)
        }
    }
```

ДОДАТОК В

НАУКОВІ ПУБЛІКАЦІЇ

В.1 Тези доповіді на міжнародному молодіжному форумі «Perspectives of Science and Education».

ДОСЛІДЖЕННЯ МЕТОДІВ ОБРОБКИ ТА ПРОЕЦІЮВАННЯ WEB-КОНТЕНТУ З ВИКОРИСТАННЯМ ЗАСОБІВ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

БРИЖНИЧЕНКО А.В.

andrii.bryzhnychenko@nure.ua

студент кафедри Програмної інженерії

Харківський національний університет радіоелектроніки

м.Харків, Україна

ЛЄСНА Н.С.

natalya.lesna@nure.ua

Професор кафедри програмної інженерії, кандидат технічних наук, професор

Харківський національний університет радіоелектроніки

м.Харків, Україна

Останнім часом завдяки активному розвитку технологій людство все частіше реалізує концепти, що були закладені ще наприкінці минулого сторіччя, зокрема такі як реалізація віртуальної та доповненої реальності.

Особливо бурхливого розвитку набули мобільні технології, що дозволяє використовувати засоби доповненої реальності навіть на смартфоні користувача, утилізуючи як програмні так і апаратні потужності пристрою. Великий внесок у розвиток AR було зроблено компанією Apple:

- випуск ARKit, що сумісний з пристроями, що були випущені 4 роки тому;
- випуск останніх X моделей що мають ще потужніші можливості та підтримують продвинуті технології AR, такі як розпізнавання 3D об’єктів, реалістичне світло, розпізнавання комплексний образів;
- випуск CoreML, що дозволяє взаємодіяти з ARKit та покращувати його роботу за допомогою моделей створених користувачем

Рисунок В.1 – Сторінка 1 статті

- випуск Create ML, утиліти що дозволяє швидко та легко створювати або приводити у належний для CoreML формат користувацькі моделі

Завдяки появі та розповсюдженні нових надійних та легких в освоєнні інструментів створення доповненої реальності, ці методи можуть бути використані у широкому спектрі застосувань, обмежених лише людською уявою, дозволяючи візуалізувати програмні об’єкти у проекції на реальний світ через камеру користувача, що може бути корисним у медицині, освіті, розважальній та навіть військових сферах. Вже зараз існують численні рішення з використанням AR: медицінські застосунки, що проеціюють внутрішні травми на тіло людини, 3D моделювання архітектурних споруд, освітні стосунки, що дозволяють проеціювати інтерактивні 3D-моделі, розважально-навчальні мультиплеєрні ігри, тощо.

Перевага сучасних AR застосунків в тому, що вони доступні та легкі в освоєнні. Однак при великому різноманітті AR-застосунків існує проблема гнучкості, а саме постачання динамічного контенту [1]. У переважній більшості випадків застосунки працюють з включеною на етапі компіляції нейронною мережею натренованою на певний датасет та з певним контентом, що зазвичай також додається до проекту на етапі створення, виключенням слугують застосунки що оперують примітивами та мітками.

Ця проблема гостро підіймається при:

- потребі змінити AR-anchors (якорі) застосунку;
- робить неможливим динамічний стрімінг та рендер контенту;
- звужує область використання застосунка;
- ускладнює використання застосунком іншого типу контенту.

Рисунок В.2 – Сторінка 2 статті

Було вирішено спроектувати систему, що може вирішити цей комплекс задач, а саме буде здатна тренувати на обраному датасеті та зберігати моделі доповненої реальності, постачати їх на мобільний застосунок, зберігати та передавати мобільному застосунку тривимірний, двовимірний або динамічний веб-контент у вигляді 3D моделей, зображень, відео, відеопотоку, анімацій, веб-сторінок, динамічно обробляти його для подальшого використання у доповненій реальності, та передавати отриманий результат на компонент доповненої реальності для подальшого проеціювання. Процес передачі, генерації та проеціювання контента відбувається в реальному часу у працюючому додатку

Для вирішення задачі створення і постачання нейронних мереж необхідно налаштувати алгоритм тренування моделі нейронної мережі на віддаленому сервері із запропонованим датасетом, переведення цієї мережі у формат, що буде здатний працювати з мобільними технологіями доповненої реальності.

Також варто враховувати те, що доповнена реальність оперує просторовими моделями, тому при отриманні веб-контенту і проекції його на реальний світ необхідно спочатку побудувати віртуальний об’єкт, на який цей контент буде проеціюватися, що враховує складну просторову геометрію розташування для побудови цього об’єкта.

Для класифікації зображень використовується згортуюча нейронна мережа, яку необхідно тренувати на запропонованому датасеті. Зазвичай це тривіальна задача, однак оскільки ця нейронная мережа буде використовуватися мобільним пристроєм, гостро стоїть проблема розміру цієї мережі. Для вирішення цієї проблеми необхідно врахувати математичну модель мереж для її оптимізації [2].

Обчислювальна складність для сучасних моделей як для навчання, так і для висновків є надзвичайно високою, що вимагає декількох графічних процесорів для навчання протягом сотень годин. Найбільш трудомістким будівельним блоком CNN, кон'юнктурним шаром, є згортання 3D вхідних даних з серією 3D ядер. Складність конволюційного шару є квадратичною з трьома гіпер-параметрами: розміром ядра, кількістю каналів і просторовими

Рисунок В.3 – Сторінка 3 статті

вимірами. З іншого боку, баланс повинен бути обережним контролем серед них, щоб забезпечити доступність обчислень. На практиці широко використовуються 1 × 1 і 3 × 3 ядра, а збільшення каналів часто супроводжується зменшенням просторових розмірів.

Якщо розглядати шари нейронної мережі як матрицю, що складається з ядер як параметрів, то можна досягти зменшення розміру нейронної мережі шляхом зменшення кількості параметрів матриці. Цього можна досягти шляхом факторизації згортань, а саме зменшення кількості з'єднань / параметрів без зменшення ефективності мережі [3].

Як було зазначено на початку текста, великий внесок у розвиток доповненої реальності було зроблено компанією Apple. Майже всі кроки цього алгоритму можна реалізувати інструментами екосистеми:

- ARKit стек дозволить приймати контент у вигляді 2D або 3D об’єктів та проеціювати його на точки, знайдені методами комп’ютерного зору;
- SpriteKit та SceneKit дозволяють створювати 2- та 3-вимірні моделі на основі динамічно отримного контенту та будування просторових моделей на основі якорів знайдених ARKit;
- CoreML фреймворк дозволить використовувати моделі, отримані ззовні для класифікації оточення, отриманого з Vision [4];
- Vision фреймворк дозволяє використовувати сенсори мобільного пристрою як інструменти машинного зору та введення оточення для класифікації нейронною мережею;
- Create ML утиліта дозволить створювати прості класифікатори для подальшого використання CoreML або конвертувати складні існуючі в необхідний формат.

Як було вказано в останньому пункті, CreateML при розробці CoreML не може створювати складні моделі, а лише базові класифікатори зображенн, звуку

Рисунок В.4 – Сторінка 4 статті

та тексту. Тому для складних комплексних ситуацій необхідно використати додатковий алгоритм генерації складних моделей:

- фреймворк для навчання моделі (TensorFlow, Keras, Turi Create і т.д.);
- платформа для запуску цієї основи для навчальних цілей (Google Colab, FloydHub, Conda і т.д.);
- програми для розмітки фотографій (RectLabel, LabelImg);
- скрипти Python, які будуть використовуватися для перетворення цих анотацій у формат, необхідний фреймворкам.

Таким чином, задача динамічного постачання моделей та контенту до кінцевого застосунку зводиться до алгоритму тренування та адаптації нейронної мережі у необхідний формат та перебудування контенту відповідно. Алгоритм роботи наведеної системи враховує складні випадки та є досить гнучким для переважної більшості випадків. Наведений алгоритм легше всього розробляється на інструментарії екосистеми Apple, проте цей підхід є універсальним для будь-якої AR платформи з відповідними інструментами.

Використана література:

[1] Ray Wanderlich, ARKit by Tutorials// Razeware LLC – p. 32-37, 2019

[2] Ian Goodfellow, Yoshua Bengio, Deep Learning (Adaptive Computation and Machine Learning series) // The MIT Press – p. 621-638, 2016.

[3] Frank Millstein, Deep Learning: 2 Manuscripts - Deep Learning With Keras And Convolutional Neural Networks In Python // Paperback – p.117-129, 2018

[4] Joshua Newman, Machine Learning with Core ML: An iOS developer's guide to implementing machine learning in mobile apps // Packt Publishing – p. 45-48, 2018

Рисунок В.5 – Сторінка 5 статті

ДОДАТОК Г

Електронні матеріали