УДК 681.325

У.Дзелендзяк[1], В.Самотий[2]

1. Національний університет “Львівська політехніка”,
кафедра комп’ютеризованих систем автоматики
2. Politechnika Krakowska im. Tadeusza Kościuszki,
katedra Automatyki i Technik Informacyjnych

СИСТЕМА АВТОМАТИЗОВАНОГО РОЗПІЗНАВАННЯ РУХІВ

Наведено систему автоматизованого розпізнавання рухів, яка дає можливість створення анімацій в режимі реального часу з використанням світлочутливих КМОН-матриць та кольорових відеокамер.

This system of automated motion recognition, which allows you to create animations in real time using photosensitive CMOS-matrix and color cameras.

Ключові слова – система розпізнавання рухів, КМОН-матриця, 3D-сканування, гіроскопічна система, XNA Framework, XNA Application, API-інтерфейси, DirectX Media Object.

Keywords - recognition system movements, CMOS sensor, 3D-scanning, gyro system, XNA Framework, XNA Application, API- interfaces, DirectX Media Object.

Вступ

Проблема розпізнавання рухів виникає в ігрових задачах, коли виникає необхідність віртуальним персонажам в точності відтворити рухи акторів. Це дає можливість спростити та пришвидшити процес створення анімацій, покращити їх якість, що обумовлює даний підхід економічно доцільним і конкурентоздатним на ринку. Досягається така швидкодія, яка стає відповідною до режиму реального часу. У порівнянні з традиційною технологією обсяги робіт значно менше залежать від складності чи тривалості анімації у порівнянні з традиційною технікою. Це дозволяє створювати багато тестових анімацій, щоб можна було оцінити різні стилі та постановки, таким чином якість кінцевого результату залежить тільки від майстерності актора.

Ми пропонуємо систему автоматизованого розпізнавання рухів, яка складається з апаратної та програмної частини. Апаратна частина є давачем глибини, що містить інфрачервоний проектор об'єднаний з монохромною світлочутливою КМОН-матрицею та кольорову відеокамеру. Програма розпізнавання рухів людини була написана мовою C# та C-подібною мовою високого рівня HLSL для програмування графіки. Ця мова була створена корпорацією Microsoft і є складовою пакету DirectX, який містить набір програмних пакетів для програмування комп’ютерних ігор. В апаратній частині використовується Point Cloud 3D-сканер. Ці пристрої в автоматичному режимі сканують велику кількість точок на поверхні об'єкту і генерують на виході файл даних. Таким чином, отриманий результат є множиною координат, отриманих 3D-скануванням об'єкту. Результати тривимірного сканування використовуються для створення тривимірних CAD-моделей виробничих деталей, для метрології та контролю якості, а також для безлічі інших цілей, пов'язаних з візуалізацією та комп'ютерною анімацією.

До переваг системи автоматизованого розпізнавання рухів слід віднести такі: один актор може грати багато ролей; живе відео на тривимірному тлі може виглядати дещо неприродньо; можливе редагування постфактум (зміна ракурсів, світла, незначне редагування рухів); більш широкі можливості костюму і гриму; можливість поєднання системи розпізнавання рухів з ручною мультиплікацією; сцену можна показати з такого ракурсу, який навіть для павільйонних зйомок

може бути утрудненим і недоступним; у сценах з великою кількістю комп'ютерних ефектів складно поєднати живих акторів з комп'ютерними персонажами.

Аналіз публікацій

Розглянемо основні види систем розпізнавання рухів. Вони розділяються на два типи, а саме, маркерні системи та безмаркерні системи. Маркерна система розпізнавання рухів – це система, де використовується спеціальне обладнання. На людину одягається костюм з датчиками, вона створює рухи, необхідні за сценарієм, встає в домовлені пози, імітує дії; дані з датчиків фіксуються камерами і надходять в комп'ютер, де зводяться в єдину тривимірну модель, яка точно відтворює рухи актора, на основі яких пізніше (або в режимі реального часу) створюється анімація персонажа. Також цим методом відтворюється міміка актора [4].

Безмаркерна технологія не потребує спеціальних датчиків або спеціального костюма. Безмаркерна технологія заснована на технологіях комп'ютерного зору й розпізнавання образів. Актор може зніматися в звичайному одязі, що сильно прискорює підготовку до зйомок і дає можливість знімати складні рухи (боротьба, падіння, стрибки, і т. п.) без ризику пошкодження датчиків або маркерів. Зйомка проводиться за допомогою звичайної камери і персонального комп'ютера [2].

Гіроскопічні системи для збору інформації про рух використовують мініатюрні гіроскопи і інерційні сенсори, розташовані на тілі актора - також як і маркери або магніти в інших системах розпізнавання рухів. Мінусами гіроскопічних систем є відсутність можливості захоплення рухів і міміки обличчя; для визначення положення актора в просторі потрібна додаткова міні-система [10].

Як наукова дисципліна, комп'ютерний зір належить до теорії та технології створення штучних систем, які отримують інформацію у вигляді зображень. Відеодані можуть бути представлені у вигляді багатьох форм, таких як відеопослідовність, зображення з різних камер або тривимірними даними з медичного сканера. Однак, існують функції, типові для багатьох систем комп'ютерного зору.[11]

Отримання зображень: цифрові зображення отримуються від одного чи декількох датчиків зображення, які окрім різноманітних типів світлочутливих камер включають датчики відстані, радари, ультразвукові камери тощо. Значення пікселів зазвичай відповідають інтенсивності світла в одній чи декількох спектральних смугах, але можуть бути пов'язані з різноманітними фізичними вимірюваннями, такими як глибина, поглинання чи відображення звукових або електромагнітних хвиль [3].

Сегментація – це процес розділення цифрового зображення на декілька сегментів. Більш точно, сегментація зображень – це процес присвоєння таких міток кожному пікселю зображення, що пікселі з однаковими мітками мають спільні візуальні характеристики [7].

Метод, заснований на кластеризації, а саме метод k-середніх – це ітераційний метод, який використовується для розділення зображення на кластери. Базовий алгоритм методу описаний в [16].

Методи з використанням гістограми дуже ефективні, в порівнянні з іншими методами сегментації зображень, тому що вони вимагають тільки один прохід по пікселям. Колір або яскравість можуть бути використані при порівнянні [12].

Підходи, засновані на використанні гістограм можна також швидко адаптувати для кількох кадрів, зберігаючи їх перевагу в швидкості за рахунок одного проходу. Цей підхід використовує сегментацію, засновану на рухомих об'єктах і нерухомому оточенні, що дає інший вид сегментації, корисний у відео трекінгу [6].

Методи розрізу графа можуть бути ефективно застосовані для сегментації зображень. У цих методах зображення представляється як зважений неорієнтований граф. Кожна частина вершин (пікселів), одержувана цими алгоритмами, вважається об'єктом на зображенні. Деякі популярні алгоритми цієї категорії — це нормалізовані розрізи графів, випадкове блукання, мінімальний розріз, ізоперіметричний поділ і сегментація з допомогою мінімального зваженого дерева [3].

Метод водоподілу — це заснований на областях метод математичної морфології. Основною проблемою даного алгоритму є надмірна сегментація, оскільки всі межі і шуми відображаються в градієнті, що робить необхідним процес видалення [1].

Постановка задачі та програмна реалізація

Система автоматизованого розпізнавання рухів розроблена для операційної системи Windows. На сьогоднішній день більшість 3D анімацій в ігровій індустрії та мультиплікації створюється вручну, що являє собою довгий та трудомісткий процес, в якому приймає участь велика кількість спеціалістів. А у медицині та спорті дані технології майже не використовуються через їх собівартість. Складні рухи та реальну фізичну взаємодію, таку як вторинні рухи, вплив ваги та фізичної сили при взаємодії об'єктів не можна легко відтворити з необхідною точністю при створенні вручну анімації для персонажів.

Для вирішення цих проблем необхідно розробити і впровадити систему, яка допоможе програмно розпізнавати та ідентифікувати рухи об'єктів, пришвидшувати процес створення анімацій, з можливістю отримання їх в режимі реального часу. Створена програма повинна швидко реагувати на зміну положення частин об’єкту, тобто його рух, відображати ці зміни на екрані та надавати можливість зберігати окремі рухи. Інтерфейс користувача повинен забезпечувати вільний доступ до всіх функціональних можливостей системи, коректно виводити інформацію на екран і при необхідності записувати анімації.

Система розпізнавання рухів допоможе покращити якість кінцевого результату, кінцевих анімацій, оскільки обсяг роботи набагато менше залежатиме від складності чи тривалості анімації в порівнянні з традиційною технікою. Це дає можливість створювати багато тестових анімацій, щоб можна було оцінити різні стилі та постановки. Як елементну базу для написання автоматизованої системи було використано:

−графічний Framework XNA;

−набір засобів розробки Kinect SDK;

−стандартні бібліотеки мови C#;

−стандартна бібліотека мови HLSL.

XNA Framework ґрунтується на реалізації .NET Compact Framework 2.0 для Xbox 360 і .NET Framework 2.0 під Windows. Він включає великий набір бібліотек класів, специфічних для розроблення графічних додатків, що підтримують максимальне повторне використання коду на всіх цільових платформах. Framework виконується на модифікації Common Language Runtime, оптимізованої для ігор, щоб створити кероване середовище виконання. Середовище часу виконання (CLR) доступний для Windows XP, Windows 7, Windows 8 і Xbox 360. Так як додатки XNA пишуться для CLR, вони можуть бути запущені на будь-якій платформі, яка підтримує XNA Framework з мінімальними змінами або взагалі без таких. Додатки, які запускаються на Framework, технічно можуть бути написані будь-якою .NET-сумісному мовою, але офіційно підтримується тільки мова програмування C# і середовища швидкого розроблення XNA Game Studio Express і всі версії Visual Studio 2010.

XNA Framework приховує низькорівневі технологічні деталі, пов'язані з розробленням графічних додатків. Таким чином, XNA Framework піклується про відмінність між платформами, даючи можливість розробникам приділяти більше уваги смисловому вмісту додатку. XNA Framework інтеґрується з кількома інструментами, такими як XACT, для допомоги у створенні контенту. XNA Framework надає підтримку при створенні двомірних і тривимірних додатків і дає можливість використовувати можливості контролерів Xbox 360. XNA Framework був обраний через доступність у ліцензії, тому додатки створені з використанням XNA Framework, на даний момент, можна поширювати через Xbox Live Community. Програмне забезпечення також може бути використане для створення комерційних додатків, призначених для Windows.

Kinect SDK дає можливість розробникам писати Kinect додатки мовами C++/CLI, C#, або Visual Basic.NET. Також цей SDK надає драйвера сумісні з Windows 7 та Windows 8 для синхронізації ПК з Kinect пристроєм.

Основні елементи роботи з Kinect SDK та XNA

Kinect SDK є складною програмною бібліотекою та інструментами, що допомагають розробникам використовувати можливості Kinect, який слідкує за реальними подіями. Kinect і бібліотека програмного забезпечення взаємодіють з додатком, як показано на рис. 1, 2.

Компоненти SDK включають в себе наступне:

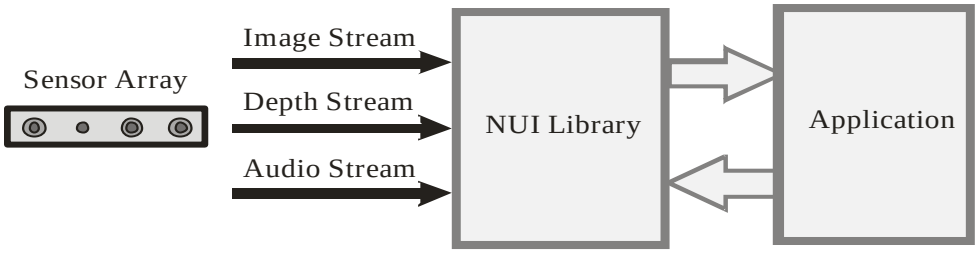


Рис.1. Апаратне і програмне забезпечення. Взаємодія з додатком

1. Апаратну частину Kinect – апаратні компоненти, в тому числі датчик Kinect і USB-вихід, через який датчик Kinect підключений до комп'ютера.

2. Драйвери Kinect – Windows драйвера для Kinect, які встановлюються в рамках процесу установки SDK. Драйвери Kinect підтримують:

- мікрофонну решітку Kinect в режимі ядра аудіопристрою, до якої ви можете отримати доступ через стандартні аудіо API в Windows;
- контроль потокового аудіо і відео для потокового аудіо і відео (колір, глибину і скелет);
- функція нумерації пристроїв, що дає можливість додатково використовувати більше одного Kinect;
- аудіо та відео компоненти;
- інтерфейс користувача для відстеження скелета, звуку, кольору і глибини зображення.

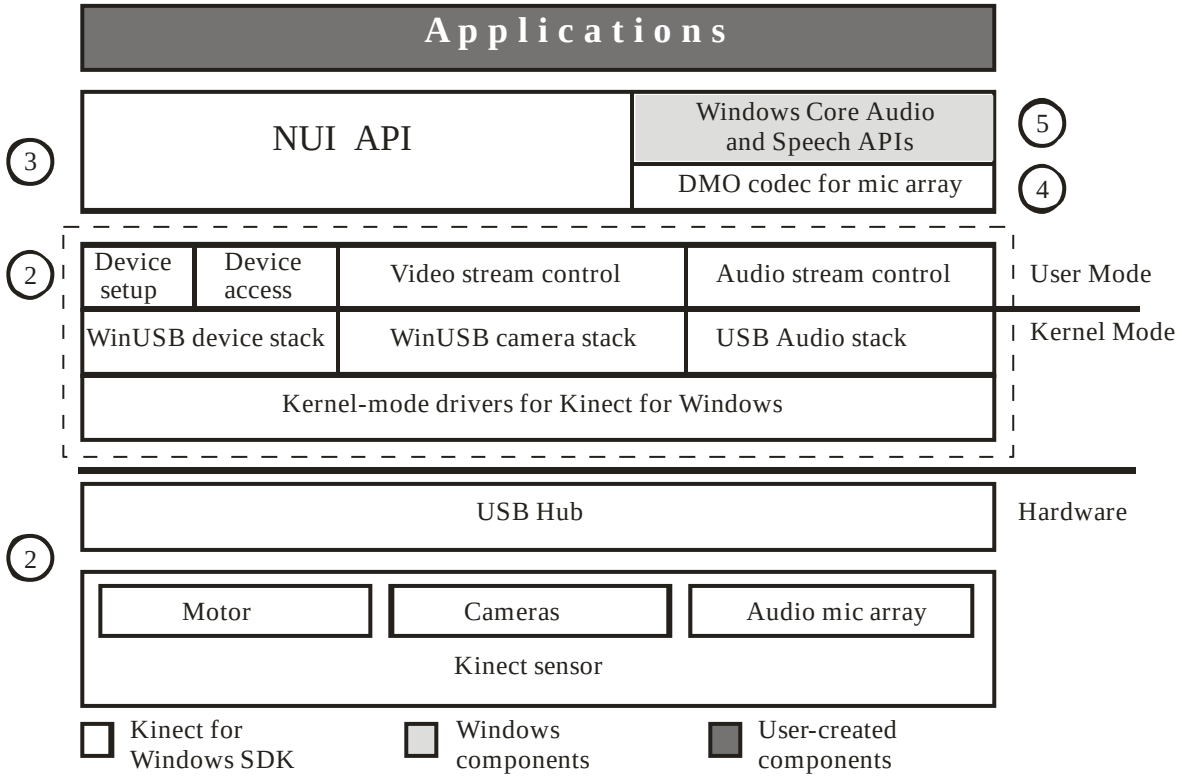


Рис. 2. SDK Архітектура

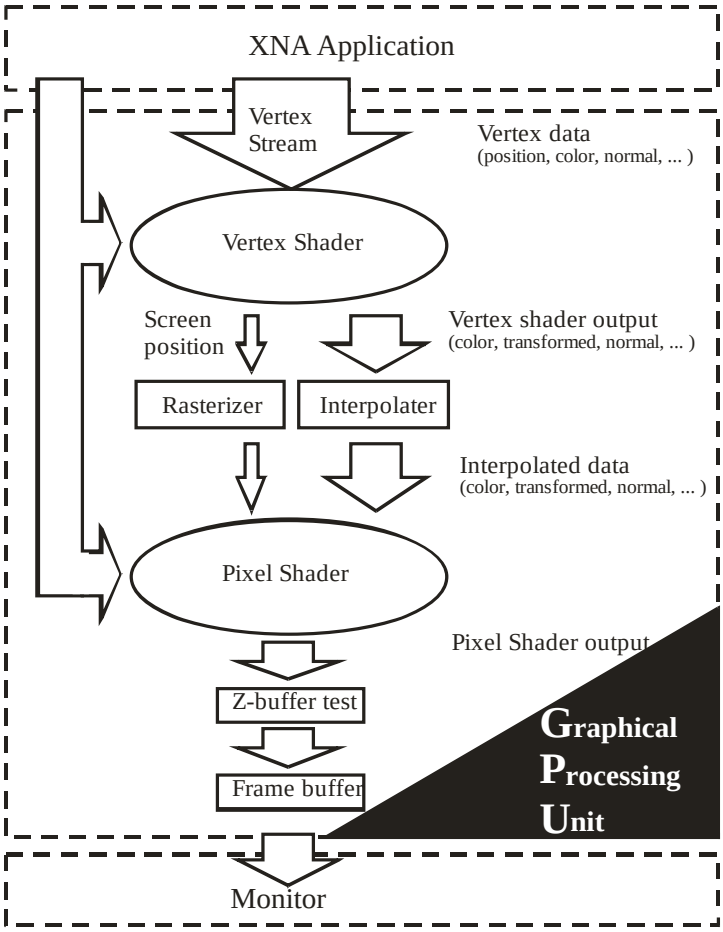


Рис. 3. Блок-схема відображення графічних об'єктів

3. DirectX Media Object (DMO) для мікрофонної решітки формування діаграми спрямованості та локалізації джерела звуку.

4. Windows 7, стандартні API-інтерфейси – звуки, мови, і API-інтерфейси медіа в Windows 7, як описано в Windows 7 SDK і Microsoft Speech SDK. Ці API також доступні для додатків в Windows 8.

Для створення системи використовувався графічний Framework XNA (рис. 3). Нижче наведені основні елементи роботи з даним Framework.

При створенні нового проекту за допомогою XNA, створюється клас Game1, який є похідним від класу Microsoft.Xna.Framework.Game, що забезпечує базову ініціалізацію графічного пристрою, логіку графічного додатку і код промальовування для додатків XNA. Клас Game1 досить простий, так як основні дії виконуються в класах GamePiece і GamePieceCollection.

До закритого члену класу відноситься об'єкт GamePieceCollection, в якому зберігаються елементи гри, об'єкт GraphicsDeviceManager і об'єкт SpriteBatch, які використовуваються для промальовування елементів додатку.

Initialize() – при виконанні даної функції виконується ініціалізація всіх об'єктів додатку.

Під час виконнаня додатку платформа XNA Framework також багаторазово викликає метод Draw. Метод Draw виконує промальовування елементів додатку, викликаючи метод Draw об'єкта GamePieceCollection.

Клас GamePiece інкапсулює всі функціональні можливості, необхідні для завантаження зображення елемента додатку Microsoft XNA, відстеження стану мишки щодо елемента гри, захоплення мишки, забезпечення оброблення маніпуляції та інерції, а також забезпечення можливості дезактивації елемента додатку, коли він досягає меж точки перегляду.

Метод UpdateFromMouse відповідає за виявлення натискання кнопки мишки, коли мишка перебуває в межах елемента додатку, і за виявлення відпускання кнопки мишки.

Коли натиснута ліва кнопка мишки (мишка знаходиться в межах кордонів елемента), цей метод встановлює прапорець, який вказує, що цей елемент гри захопила мишка, і починає оброблення маніпуляції.

Оброблення маніпуляції починається зі створення масиву об'єктів Manipulator2D і передачі їх об'єкту ManipulationProcessor2D. Це змушує процесор маніпуляції оцінити маніпулятори (в даному випадку один маніпулятор) та ініціювати події маніпуляції.

Особливості реалізації системи автоматизованого розпізнавання рухів

Система автоматизованого розпізнавання рухів працює за наступним алгоритмом:

1. З сенсору Kinect отримуємо набір байтів кольорового зображення.
2. З даного набору формуємо кольорове зображення.

3. Паралельно з сенсору отримується множина точок трьохвимірного простору.
4. На основі даних точок формується карта глибини.
5. Проводиться процес сегментації кольорового зображення.
6. Зображення передаються на NUI API.
7. Отримуємо масив точок з ідентифікаторами суглобів.

Для сегментації використовуються алгоритми розмивання Гауса, k-середніх.

Розмивання Гауса - це тип фільтру розмивання зображення, що використовує функцію Гауса для розрахунку трансформації кожного пікселя у зображенні.

Рівняння функції Гауса в одному вимірі:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}, \quad (1)$$

у двох вимірах, воно є результатом двох таких функцій, по одній на напрямок:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \quad (2)$$

де x – це відстань від початку координат по осі абсцис; y – це відстань від початку координат по осі ординат, σ – стандартне відхилення розподілу Гауса.

Коли метод застосовується у двох вимірах, отримується поверхня контури якої - це концентричні кола розподілу Гауса з центральної точки. Значення з цього розподілу використовуються для створення матриці згортки. Для кожного нового значення пікселя визначається середнє зважене в околі пікселя. Значення поточного оригінального пікселя має більшу вагу (найвище значення розподілу Гауса), а сусідні пікселі отримують все меншу вагу в залежності від того наскільки далеко вони знаходяться від поточного оригінального пікселя. Це надає ефект розмитості, яка зберігає кордони та краї краще, ніж інші аналогічні фільтри розмиття.

Теоретично, в кожній точці зображення буде відмінним від нуля результату функції Гауса. Це означає, що при розрахунку для кожного пікселя необхідно брати значення усіх пікселів у зображенні. На практиці, при обчисленні дискретного спрощення функції Гауса, вага пікселів на відстані більш ніж 3σ достатньо мала, щоб мати якийсь вплив на середнє зважене значення і вважається нулем. Значення пікселів, що розташовані за межами цього діапазону можуть бути проігноровані. Як правило, програмі оброблення зображень необхідно тільки розрахувати матриці з розмірами $\lceil 6\sigma \rceil \times \lceil 6\sigma \rceil$, щоб забезпечити результат, який є досить близьким до отриманого від усього розподілу Гауса.

Розмивання Гауса може застосовуватися для двовимірних зображень у вигляді двох незалежних одновимірних, так званих лінійно розділених обчислень. Тобто, ефект застосування двовимірної матриці може бути досягнуто за рахунок застосування ряду одновимірної матриці Гауса в горизонтальному напрямку, а потім повторно у вертикальному напрямку. З точки зору швидкості обчислення, це корисна властивість, оскільки розрахунки можуть бути виконані в $O(\omega_{\text{kernel}}\, \omega_{\text{image}}\, h_{\text{image}}) + O(h_{\text{kernel}}\, \omega_{\text{image}}\, h_{\text{image}})$ час (де h – це висота і ω – це ширина), на відміну від $O(\omega_{\text{kernel}}\, h_{\text{kernel}}\, \omega_{\text{image}}\, h_{\text{image}})$ для нерозділених обчислень.

Застосування розмивання Гауса призводить до розмивання на зображення і має ті ж наслідки, що застосування єдиного розмивання Гауса, радіус якого рівний квадратному кореню з суми квадратів радіусу розмиття, що застосовується. Наприклад, застосування послідовного розмивання Гауса з радіусом 6 і 8 дає ті ж результати, як застосування єдиного розмивання радіусом 10, оскільки $\sqrt{6^2 + 8^2} = 10$. Згідно з цим відношенням, час оброблення не може бути зменшений шляхом імітації розмивання Гауса з послідовними процесом.

В розробленні системи використовувався лінійно відокремлений алгоритм розмивання Гауса, який розділяє процес на два етапи. У першому проходженні, одновимірна матриця

використовується для розмиття зображення тільки в горизонтальному або вертикальному напрямку. На другому проході, матриця використовується для розмиття в іншому напрямку. Отриманий результат відповідає результату з використанням двовимірних матриць, але вимагає меншої кількості обчислень.

Метод *k*-середніх (*k*-means) – це метод кластеризації, мета якого розділити n спостережень на *k* кластерів, так щоб кожне спостереження належало до кластера з найближчим до нього середнім значенням. Метод базується на мінімізації суми квадратів відстаней між кожним спостереженням та центром його кластера, тобто функції

$$\sum_{i=1}^{N} d(x_i,\ m_j(x_i))^2, \tag{3}$$

де d – метрика, x_i – *i*-ий об’єкт даних, $a^{m_j(x_i)}$ – центр кластера, якому на *j*-ій ітерації приписаний елемент x_i.

Алгоритм методу:

1. Маємо масив спостережень (об’єктів), кожен з яких має певні значення по ряду ознак. Відповідно до цих значень об’єкт розташовується у багатовимірному просторі.
2. Дослідник визначає кількість кластерів, що необхідно утворити.
3. Випадковим чином обирається *k* спостережень, які на цьому кроці вважаються центрами кластерів.
4. Кожне спостереження «приписується» до одного з *n* кластерів – того, відстань до якого найкоротша.
5. Розраховується новий центр кожного кластера як елемент, ознаки якого розраховуються як середнє арифметичне ознак об’єктів, що входять у цей кластер.
6. Відбувається така кількість ітерацій (повторюються кроки 3-4), поки кластерні центри стануть стійкими (тобто при кожній ітерації в кожному кластері опинятимуться одні й ті самі об’єкти), дисперсія всередині кластера буде мінімізована, а між кластерами – максимізована.

Вибір кількості кластерів відбувається на основі дослідницької гіпотези.

Якщо її немає, то рекомендують створити 2 кластери, далі 3, 4, 5, порівнюючи отримані результати.

Головні переваги методу *k*-середніх – його простота та швидкість виконання. Метод *k*-середніх більш зручний для кластеризації великої кількості спостережень, ніж метод ієрархічного кластерного аналізу (у якому дендограми стають перевантаженими і втрачають наочність).

ContentLoadException – виняток використовується для повідомлення про помилки методу ContentManager.Load. ContentManager – компонент, який завантажує керовані об'єкти з бінарних файлів. Він також управляє терміном життя завантажених об'єктів, вивільненням ContentManager, сприяє вивільненню всіх завантажених об’єктів. Load<T>() – завантажує актив, оброблений за конвеєром вмісту. Effect – використовується для встановлення та доступу до ефекту та вибору методу промальовування. Model – складається з декількох об'єктів ModelMesh, які можуть бути оброблені самостійно. SpriteFont – текстура шрифту. Для її завантаження потрібно додати XML-файл в проект, що описує, як побудувати карту текстури для вашого шрифту. Під час побудови проекту, XNA Game Studio створює текстуру з зображенням символів шрифту, який ви вкажете, з урахуванням зазначеного розміру точки шрифту. Texture – це ресурс текстури. Texture2D – це 2D сітка з текселів. Тексель є найменшою одиницею текстури, яка може бути порахована або записана в GPU. Тексель складається від 1 до 4 компонентів. Зокрема, тексель може бути поданий у будь-якому з доступних форматів текстур, наявних в SurfaceFormat. TextureCube – це набір з шести 2D текстур, по одному для кожної грані куба. SpriteBatch включає групу спрайтів для провалювання з деякими параметрами. SpriteBatch.Begin() починає операцію пакетного промальовування спрайтів з використанням відкладеного сортування та параметрами за замовчуванням. SpriteBatch.Draw додає спрайт до пакету спрайтів для промальовування, використовуючи зазначену текстуру, положення, вихідний прямокутник, колір, обертання, походження, масштаб. SpriteBatch.DrawString додає спрайт до пакету спрайтів для промальовування використовуючи зазначений шрифт, текст,

положення, колір, обертання, масштаб. SpriteBatch.End очищає пакет спрайтів і відновлює стан пристрою, як це було раніше перед викликом SpriteBatch.Begin(). BlendState містить стан накладення. Additive – при побудові проекту налаштовує об’єкти, як AdditiveBlend, що додає дані призначення до вихідних даних без використання альфа каналу. AlphaBlend – при побудові проекту налаштовує об’єкти для змішування об’єктів з використанням Alpha каналу. NonPremultiplied – при побудові проекту налаштовує об’єкти NonPremultipliedAlpha, для змішування даних об’єктів. StaticOpaque – при побудові проекту налаштовує об’єкти для змішування даних об’єктів. SamplerState – містить стан Sampler, який визначає зразок даних текстури. AnisotropicClamp – містить стан за замовчуванням для анізотропної фільтрації текстур з одиночним координуванням. AnisotropicWrap містить за замовчуванням стан для анізотропної фільтрації текстур з повторенням координат. LinearClamp містить стан за замовчуванням для лінійної фільтрації текстур з одиночним координуванням. LinearWrap містить за замовчуванням стан для лінійної фільтрації текстур з повторенням координат. PointClamp містить стан за замовчуванням для точкової фільтрації текстур з одиночним координуванням. PointWrap містить за замовчуванням стан для точкової фільтрації текстур з повторенням координат. DepthStencilState містить стан глибини трафарету. Default вбудований стан об'єкта з налаштуваннями за замовчуванням для використання буфера глибини трафарету. DepthRead – вбудований стан об'єкта з налаштуваннями, які дозволяють лише читати буфер глибини трафарету. None – вбудований стан об'єкта з налаштуваннями за замовчуванням без використання буфера глибини трафарету. RasterizerState містить стан растеризатор, який визначає, як перетворити векторні дані в растрові дані. CullClockwise – вбудований стан об'єкта з налаштуваннями для виклику примітивів за годинниковою стрілкою. CullCounterClockwise – вбудований стан об'єкта з налаштуваннями для виклику примітивів проти годинникової стрілки. CullNone – вбудований стан об'єкта з налаштуваннями, щоб не викликати жодний з примітивів.

Реалізація додаткових підпрограм системи автоматизованого розпізнавання рухів

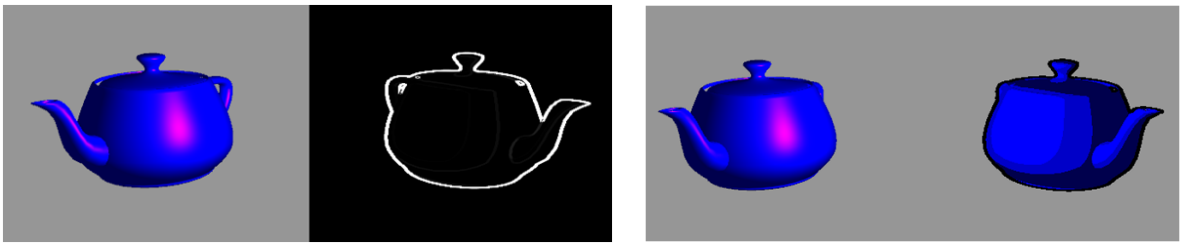
В системі автоматизованого розпізнавання рухів було створено три підпрограми, а саме:

1. Перетворення 3D анімації в 2D анімацію, з подальшим записом результатів перетворень.
2. Тестова гра наочно відображає взаємодію сенсорів системи з користувачем в режимі реального часу.
3. Постоброблення анімації для згладжування та видалення паразитних шумів.

Для перетворення 3D анімації в 2D анімацію, був використаний алгоритм оператора Собеля.

Оператор Собеля (рис. 4 а)) використовується в області оброблення зображень. Часто його застосовують в алгоритмах виділення кордонів. По суті, це дискретний диференціальний оператор, який обчислює наближене значення градієнта яскравості зображення.

Строго кажучи, оператор використовує ядра 3×3, з якими згортають вихідне зображення для обчислення наближених значень похідних по горизонталі і по вертикалі. Нехай A вихідне зображення, а Gx і Gy - два зображення, де кожна точка містить наближені похідні по x і по y.



а) *б)*

Рис. 4. а) результат виконання оператора Собеля, б) результат кінцевого оброблення

Далі, зрівнюючись з деяким пороговим значенням, будуємо спеціальну текстуру з межами і складаємо зображення (рис. 4 б)). Тестова гра, яка створена у даній роботі, наглядно показує практичне застосування розробленої системи у сфері розважання.

Висновок

Розроблено і досліджено систему автоматизованого розпізнавання рухів для платформи Windows. Створена система має такі функції: розпізнавання рухів людини, причому розпізнавання відбувається в реальному часі; збереження створених анімацій та подальше їх оброблення. На основі порівняльної оцінки існуючих технологій та програм для створення анімацій, була розроблена система автоматизованого розпізнавання рухів, яка включає апаратну частину та програмне середовище. Апаратна частина представляє собою давач глибини, що складається з інфрачервоного проектора об'єднаного з монохромною КМОН-матрицею, та кольорову відеокамеру. Програмна частина – це програма написана мовою C# з використанням технології DirectX. Розроблена автоматизована система дає можливість створення як 3D, так і 2D анімації, в залежності від потреб користувача. Розроблена система дозволяє порівнювати рухи користувача в реальному часі з еталонними, що дозволяє користувачу вивчати дані рухи (наприклад танцювальні рухи, семафорна азбука). Застосування даної системи автоматизованого розпізнавання рухів для створення розважальних програм може зменшити витрати та обсяг роботи на створення ключових кадрів анімації. Вона дозволяє створювати багато тестових анімацій для оцінювання різних стилів та постановок, таким чином якість кінцевого результату залежить тільки від майстерності актора.

1. Hubert Nguyen GPU Gems 3, 2007. 2. Edwards, Cliff (2009-06-01). «Microsoft Moves onto Nintendo's Motion Turf». BusinessWeek. McGraw-Hill. с. 1–2. 3. Nutt, Christian (2008-01-17). «Q&A: 3DV's Barel On The Future Of Camera-Based Game Control». Gamasutra. Think Services. 4. Totilo, Stephen (2009-06-01). «Why Xbox 360 New Controller Is Called 'Natal'». Kotaku. Gawker Media. 5. Jason Sanders, Edward Kandrot CUDA BY EXAMPLE: AN INTRODUCTION TO GENERAL-PURPOSE GPU PROGRAMMING, 2010. 6. Wilson, Mark; Buchanan, Matt (2009-06-03). «Testing Project Natal: We Touched the Intangible». Gizmodo. Gawker Media. 7. Wingfield, Nick (2009-05-13), «Microsoft Swings at Wii With Videocam», The Wall Street Journal (Dow Jones & Company): B1, ISSN 0099-9660, OCLC 4299067, процитовано 2009-06-02. 8. «E3 2009: Microsoft Press Conference Live Blog». IGN. 2009-06-01. 9. «E3: Microsoft Xbox throws down gauntlet with "Natal" controller». The Seattle Times. 10. «„Project Natal“ 101». Microsoft. 2009-06-01. Archived from the original on 2009-06-01. 11. «"Project Natal" 101». Microsoft. 2009-06-01. 12. http://http.developer.nvidia.com/GPUGems3/gpugems3_ch26.html. 13. http://cvrc.ece.utexas.edu/Publications/HAU3D11_Xia.pdf. 14. http://blog.seattlepi.com/digitaljoystick/archives/169993.asp.