Trabajo de Graduación de la Licenciatura en Diseño Multimedial

Título:

Agrovision

Tema:

Visualización de cultivos subterráneos con detección de enfermedades y anomalías, y seguimiento de las distintas fases del cultivo mediante 3D y machine learning.

Nombre: Consuelo Oss
Legajo: 81720/3
Mail: consuoss1994@gmail.com
Teléfono: 2223-508934

2. Tema

Visualización de cultivos subterráneos con detección de enfermedades y anomalías, y seguimiento de las distintas fases del cultivo mediante 3D y machine learning.

3. Resumen

Este trabajo se enfoca en resolver las problemáticas que enfrenta un agricultor al cultivar vegetales, tubérculos y raíces subterráneas. Agrovision, es una aplicación/software cuyo objetivo es mostrarle al usuario cómo y de qué manera están creciendo sus cultivos subterráneos, ya que estos no pueden ser observados de manera directa.

Para resolver esta problemática, el usuario podrá gestionar sus cosechas, controlando los procesos, y podrá visualizar el cultivo gracias a que la aplicación está conectada a un dispositivo de geo radar que capta la señal y la transmite al dispositivo que el usuario elija. A su vez, este sistema de monitoreo tiene la particularidad de ser modular, es decir, se puede adaptar a las diferentes necesidades de un productor grande o pequeño.

Además, gracias a los datos que extraen los diferentes sensores, al usuario se le mostrarán estadísticas y recomendaciones para mejorar su producción, entre otras funcionalidades.

4. El Problema

La principal problemática de esta propuesta radica en la incapacidad de observar directamente los cultivos subterráneos una vez que han sido plantados, esta limitación conlleva consecuencias negativas tanto para los vegetales, raíces o tubérculos como para la pérdida económica del usuario ya que después no podrá ni vender ni consumir dicha materia prima.

Al cultivar de manera subterránea, existen diversas etapas de crecimiento sobre las cuales el usuario carece de información directa, no puede determinar si el cultivo está progresando, si se enfrenta a anomalías debido a plagas, o si presenta deficiencias de humedad, entre otros posibles problemas. Esto deja al usuario en la incertidumbre mientras espera el desarrollo del cultivo, el cual podría no estar creciendo adecuadamente o estar enfrentando dificultades significativas.

5. 1 Objetivo general

El objetivo principal de este proyecto es desarrollar una aplicación para dispositivos tecnológicos que permita al usuario visualizar, a través de la conexión con dispositivos de penetración terrestre, los cultivos subterráneos que no son visibles de manera directa. Esta

La solución utilizará técnicas de aprendizaje profundo (deep learning) para asegurar al usuario la capacidad de monitorear el estado y desarrollo de sus cultivos en tiempo real.

5.2 Objetivo específico

- Reducir los costos operativos del usuario al optimizar la gestión y supervisión de los cultivos subterráneos.
- Proporcionar herramientas intuitivas para la administración eficiente de los cultivos, incluyendo funciones de historial de cosechas y gestión de datos.
- Implementar una visualización en 3D de los cultivos subterráneos para una comprensión más completa de su estado y desarrollo.
- Desarrollar un sistema de asistencia virtual basado en los datos captados por el GPR (Ground Penetrating Radar) para ofrecer recomendaciones y alertas sobre condiciones adversas o anomalías en los cultivos.
- Diseñar interfaces interactivas que permitan a los usuarios explorar de manera intuitiva información detallada sobre cada cultivo y su evolución.
- Integrar diversas herramientas y tecnologías con el fin de crear una experiencia interactiva adaptada a las necesidades específicas de la agricultura subterránea.

6. Metodología

6.1 Tecnologías usadas

- **VisualSFM:** Software que utiliza una técnica llamada fotogrametría para crear modelos 3D a partir de un conjunto de fotografías de un objeto o escena. En pocas palabras, toma varias fotos desde diferentes ángulos y las combina para generar una representación tridimensional detallada. Sirve para c
- **MeshLab:** Software de código abierto especializado en el procesamiento y edición de mallas triangulares 3D. MeshLab toma como punto de partida modelos 3D y permite: eliminar agujeros, suavizar superficies, unificar vértices, modificar el modelo, agregar o eliminar detalles, etc.
- **Tensor Flow Lite:** Librería de código libre para Machine Learning (ML), desarrollado para satisfacer las necesidades a partir de redes neuronales artificiales. Te permite construir y entrenar redes neuronales para detectar patrones y razonamientos usados por los humanos. Facilita la creación e implementación de modelos de aprendizaje automático. Permite crear y entrenar modelos de Machine Learning con facilidad mediante APIs intuitivas.
- **Figma:** Plataforma de edición gráfica y diseño de interfaces online y colaborativa. Se puede diseñar páginas web e interfaces gráficas de aplicaciones, o crear publicaciones para redes sociales.

6.2 Proceso

Investigación sobre Cultivos Subterráneos

a. **Papas:** Es un tubérculo comestible que crece bajo la tierra. Su principal función es almacenar o acumular nutrientes. La planta requiere de un suelo drenado y rico en nutrientes para que estos crezcan en un suelo arenoso. Responde bien a las temperaturas templadas, sufre cuando es excesiva y sobre todo cuando hay sequía. La planta no debe pasar sed en ningún momento, sobre todo al principio, porque podría reducir el rendimiento a la cuarta parte. Esta tiene diferentes etapas que van desde la preparación del suelo y colocación de las semillas hasta la extracción de está pasando por la formación de tallos, ramas y hojas; inicio de floración donde las células de los tuberculos se expanden por la acumulacion del agua, nutrientes y carbohidratos.
a. **Zanahoria:** Es una de las hortalizas más cultivadas en el mundo y la parte consumida es su raíz, de la que existen múltiples formas y sabores. Destaca por su contenido en caroteno y vitaminas A, B y C.Crece en cualquier tipo de suelo sin requerir de muchos nutrientes y cuidados especiales. Asimismo, se desarrolla mucho mejor bajo condiciones templadas y tiene la capacidad de soportar heladas ligeras. Tiene diferentes etapas que empiezan cuando germinan las semillas aproximadamente de 1 a 3 semanas, luego crecerán un par de cotiledones largos y de ahí se pasa a la etapa del crecimiento y desarrollo que dura de 1,5 a 3 semanas y por último la extracción de la misma.
b. **Trufas:** La trufa es un género de hongos ascomicetos de la familia de las tuberías y crece cerca de las raíces de algunos árboles como robles o encinas. Existen más de 70 especies diferentes, 32 de ellas europeas, y solo hay 30 tipos de trufas comestibles, aunque no todas con el mismo valor culinario. El ciclo de crecimiento de la trufa comienza con la inoculación de esporas en el árbol huésped, donde germinan y forman una red de hifas llamada micelio que busca las raíces del árbol para establecer una relación simbiótica. Esta asociación mejora la absorción de nutrientes y agua para ambos. Luego, a partir del micelio, se forman los primordios de la trufa, que crecen lentamente bajo tierra. Finalmente, la trufa madura y se cosecha con la ayuda de perros entrenados para detectarlas.

Construcción de Dataset de Imágenes en 3D

Una vez que tenemos la información de cada cultivo el paso siguiente es poder detectar diferentes estadios del crecimiento, plagas y anomalías en rizomas, tubérculos, zanahorias y papas utilizando técnicas de visión artificial y deep learning. Para lograrlo, se entrenará un modelo de deep learning con datos recolectados incluyendo imágenes de estos en diversos estadios de crecimiento, que cuentan con diferentes condiciones de iluminación, ángulos y fondos, y que muestren plagas comunes, síntomas de enfermedades que están etiquetadas con nombres claves.

Antes de desarrollar el modelo de deep learning, debemos construir un dataset de imágenes en 3D de los cultivos en diferentes estados, incluyendo cultivos sanos, cultivos con enfermedades

y cultivos afectados, investigamos en Kaggle si había datasets de imágenes en 3D, pero no logramos encontrar ninguno adecuado, por lo que se tuvo que crear un dataset. Para la construcción de la imagen en 3D, utilizaremos dos programas de software: VisualSFM (Wu, 2015) y MeshLab (MeshLab, 2015), ambos fácilmente accesibles a través de internet.

El primer paso es obtener un conjunto de fotografías del cultivo, asegurándonos de que las imágenes se solapen entre sí, es necesario tomar las fotografías moviendo la cámara alrededor del objeto. El objetivo es que el software encuentre un conjunto de puntos comunes entre cada una de las imágenes para generar una nube de puntos tridimensional que represente el escenario donde se encuentra situado el cultivo. Luego, utilizando VisualSFM, se importarán las imágenes tomadas y se creará una "nube de puntos", que es una estructura creada a partir de los puntos comunes obtenidos por la comparación entre todas las fotografías y en base a esta nube de puntos, utilizando MeshLab para eliminar todos aquellos puntos que aunque formen parte de las fotografías no pertenecen al objeto que queremos reconstruir. Posteriormente, se cambiarán nuevos puntos utilizando un algoritmo matemático basado en la ecuación de Poisson, con el fin de construir una malla cerrada que dé como resultado el cuerpo del objeto. Finalmente, a partir de las fotografías originales, habrá que añadirle la textura y el color al cuerpo, dando como resultado el objeto 3D deseado.

Implementación del Modelo de Deep Learning

El resultado será una representación tridimensional del cultivo que puede ser observada desde todos sus ángulos y una vez que ya tenemos el dataset con las imágenes de los cultivos en 3D, volvemos a la plataforma Kaggle, para buscar datasets sobre imágenes de estados de cultivos, plagas en tierra y en cultivo, etc. Este contiene sus etiquetas, que son descripciones o categorías asignadas a cada imagen, como "estadios de crecimiento", "tipos de plagas" o "anomalías", estas son importantes para el entrenamiento del modelo porque le permiten aprender a partir de ejemplos con respuestas conocidas.

Hay que extraer, procesar y analizar los datos para entender la estructura del dataset. El preprocesamiento implica normalizar los valores de píxeles para que estén en un rango común (por ejemplo, entre 0 y 1) y aplicar técnicas de aumento de datos como rotaciones, traslaciones y cambios de brillo y contraste para multiplicar la variabilidad de los datos y mejorar la capacidad del modelo para generalizar patrones.

Para la implementación del modelo, se eligieron las redes neuronales convolucionales (CNN), que son adecuadas para tareas de visión artificial porque están diseñadas para procesar datos con estructura de cuadrícula, como imágenes. Las principales capas que componen una CNN incluyen capas convolucionales que extraen características específicas de las imágenes aplicando filtros, capas de pooling que reducen la dimensionalidad de las características extraídas manteniendo la información esencial, y capas totalmente conectadas que realizan la clasificación final basándose en las características extraídas y reducidas. ResNet es una red neuronal residual de CNN usada especialmente en problemas de visión artificial complejos. Para la implementación, usamos Tensor Flow, donde se definieron las capas convolucionales, las capas de pooling y las capas totalmente conectadas necesarias para la clasificación final.

El entrenamiento tiene varias etapas hasta alcanzar un rendimiento óptimo y se evalúa usando un conjunto de datos de validación para verificar su capacidad de generalización a datos no vistos. Se calculan métricas de rendimiento como precisión, que es la proporción de predicciones correctas sobre el total de predicciones; recall, que mide la capacidad del modelo para detectar todas las instancias relevantes; F1-score, que es la media armónica de la precisión y el recall, proporcionando una medida equilibrada; y la matriz de confusión, una tabla que muestra las predicciones correctas e incorrectas distribuidas en cada clase, ayudando a identificar patrones de error.

Como último paso debemos desarrollar la aplicación e integrar todo el modelo neuronal para que el usuario pueda hacer uso de dicha aplicación, esta se desarrollará por separado con Android Studio para dispositivos que tengan Android y con Swift para aquellos que utilicen iOS.

El flujo del proceso es el siguiente: primero, el modelo recibe como entrada una imagen, que debe ser escalada al tamaño utilizado durante el entrenamiento de la red neuronal para asegurar una correcta predicción. Esta imagen puede ser adquirida mediante la cámara del dispositivo y una vez obtenida, se escala adecuadamente y se convierte en un buffer compatible con Java.

Desarrollo de la aplicación

Luego del entrenamiento, se exporta el modelo en formato TensorFlow Lite (.tflite), se crea un nuevo proyecto en Android Studio, importando las librerías de TensorFlow Lite añadiendo las dependencias necesarias en el archivo `build.gradle` y se le dan los permisos correspondientes para acceder a la camara del dispositivo que se usará la aplicación.
Para el desarrollo en Java se deben importar las librerías de TensorFlow Lite en los archivos Java necesarios e implementar funciones para capturar imágenes desde la cámara, para concluir se carga el modelo TensorFlow y se pasa la imagen preprocesada al modelo utilizando las API de TensorFlow Lite.
Adjunto el codigo de cómo se integraría TensoFlow Lite una aplicación Android utilizando Java:

El ejemplo de código en Swift proporcionado anteriormente es específico para iOS y no puede ser utilizado directamente para Android. Para desarrollar una aplicación Android, se debe usar Java o Kotlin y seguir un enfoque similar pero utilizando las herramientas y API específicas de Android. A continuación, te muestro un ejemplo de cómo se integraría TensorFlow Lite en una aplicación Android utilizando Java:

Agregar Dependencias de TensorFlow Lite en `build.gradle`:

```gradle
dependencies {
    implementation 'org.tensorflow:tensorflow-lite:2.7.0'
}
```

Código para Capturar y Procesar Imágenes en Android:

```java
import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.provider.MediaStore;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import org.tensorflow.lite.Interpreter;
import java.io.FileInputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;

public class MainActivity extends AppCompatActivity {

    private static final int REQUEST_IMAGE_CAPTURE = 1;
    private static final int REQUEST_GALLERY_IMAGE = 2;
    private static final int REQUEST_PERMISSIONS = 3;

    private Interpreter tflite;
    private ImageView imageView;
    private TextView resultTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        imageView = findViewById(R.id.imageView);
        resultTextView = findViewById(R.id.resultTextView);
```

```
        // Solicitar permisos
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED ||
            ContextCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA,
Manifest.permission.READ_EXTERNAL_STORAGE}, REQUEST_PERMISSIONS);
        }

        // Cargar el modelo TensorFlow Lite
        try {
            tflite = new Interpreter(loadModelFile());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // Método para abrir la cámara
    public void openCamera() {
        Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
            startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
        }
    }

    // Método para abrir la galería
    public void openGallery() {
        Intent pickPhoto = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(pickPhoto, REQUEST_GALLERY_IMAGE);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == RESULT_OK) {
            Bitmap imageBitmap = null;
            if (requestCode == REQUEST_IMAGE_CAPTURE && data != null) {
                Bundle extras = data.getExtras();
                imageBitmap = (Bitmap) extras.get("data");
            } else if (requestCode == REQUEST_GALLERY_IMAGE && data != null) {
                try {
```

```
                imageBitmap = MediaStore.Images.Media.getBitmap(this.getContentResolver(),
data.getData());
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        if (imageBitmap != null) {
            imageView.setImageBitmap(imageBitmap);
            performInference(imageBitmap);
        }
    }
}

// Método para cargar el modelo TensorFlow Lite
private MappedByteBuffer loadModelFile() throws IOException {
    FileInputStream fileInputStream = new
FileInputStream(getAssets().openFd("model.tflite").getFileDescriptor());
    FileChannel fileChannel = fileInputStream.getChannel();
    long startOffset = getAssets().openFd("model.tflite").getStartOffset();
    long declaredLength = getAssets().openFd("model.tflite").getDeclaredLength();
    return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset, declaredLength);
}

// Método para realizar la inferencia
private void performInference(Bitmap bitmap) {
    Bitmap scaledBitmap = Bitmap.createScaledBitmap(bitmap, 224, 224, true); // Ajustar
tamaño según el modelo
    ByteBuffer inputBuffer = ByteBuffer.allocateDirect(4 * 224 * 224 * 3);
    inputBuffer.order(ByteOrder.nativeOrder());
    int[] intValues = new int[224 * 224];
    scaledBitmap.getPixels(intValues, 0, 224, 0, 0, 224, 224);
    for (int pixel : intValues) {
        inputBuffer.putFloat(((pixel >> 16) & 0xFF) / 255.0f);
        inputBuffer.putFloat(((pixel >> 8) & 0xFF) / 255.0f);
        inputBuffer.putFloat((pixel & 0xFF) / 255.0f);
    }
    float[][] output = new float[1][10]; // Ajustar según el número de categorías
    tflite.run(inputBuffer, output);
    // Mostrar el resultado
    resultTextView.setText("Predicción: " + getMaxIndex(output[0]));
}

// Método para obtener la categoría con mayor probabilidad
private int getMaxIndex(float[] probabilities) {
```

```
        int maxIndex = 0;
        float maxProb = probabilities[0];
        for (int i = 1; i < probabilities.length; i++) {
            if (probabilities[i] > maxProb) {
                maxProb = probabilities[i];
                maxIndex = i;
            }
        }
        return maxIndex;
    }
}
```

Para desarrollar la aplicación en Swift se debe exportar el modelo entrenado a un archivo de Tensor Flow Lite (.tflite), crear un nuevo proyecto en Xcode, utilizaremos swift como lenguaje de programación y se agregarán las dependencias de Tensor Flow Lite usando CocoaPods. Luego se importará las librerías de TensorFlow Lite en los archivos Swift necesarios y se implementaran las funciones para capturar imágenes desde la cámara usando `UIImagePickerController`. Luego se cargará el modelo Tensor Flow Lite y se pasa la imagen preprocesada al modelo utilizando las API de Tensor Flow Lite en Swift.

Adjunto el codigo de como cómo se integraría Tensor Flow Lite en una aplicación iOS:

```swift
import UIKit
import TensorFlowLite

class ViewController: UIViewController, UIImagePickerControllerDelegate,
UINavigationControllerDelegate {

    // Cargar el modelo Tensor Flow Lite
    var interpreter: Interpreter?

    override func viewDidLoad() {
        super.viewDidLoad()
        guard let modelPath = Bundle.main.path(forResource: "model", ofType: "tflite") else {
            fatalError("Failed to load the model file.")
        }
        interpreter = try? Interpreter(modelPath: modelPath)
    }

    // Función para abrir la cámara
    func openCamera() {
        let imagePicker = UIImagePickerController()
```

```
        imagePicker.delegate = self
        imagePicker.sourceType = .camera
        present(imagePicker, animated: true, completion: nil)
    }

    // Función para abrir la galería
    func openGallery() {
        let imagePicker = UIImagePickerController()
        imagePicker.delegate = self
        imagePicker.sourceType = .photoLibrary
        present(imagePicker, animated: true, completion: nil)
    }

    // Procesar la imagen seleccionada
    func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo
info: [UIImagePickerController.InfoKey : Any]) {
        if let selectedImage = info[UIImagePickerController.InfoKey.originalImage] as? UIImage {
            // Preprocesar la imagen
            let scaledImage = preprocessImage(image: selectedImage)
            // Realizar la inferencia
            performInference(image: scaledImage)
        }
        picker.dismiss(animated: true, completion: nil)
    }

    // Función para preprocesar la imagen
    func preprocessImage(image: UIImage) -> Data {
        // Escalar la imagen al tamaño adecuado y convertirla a un buffer de bytes
        // Este es solo un ejemplo y debería ser ajustado según tu modelo
        return Data()
    }

    // Función para realizar la inferencia
    func performInference(image: Data) {
        guard let interpreter = interpreter else { return }
        do {
            try interpreter.allocateTensors()
            try interpreter.copy(image, toInputAt: 0)
            try interpreter.invoke()
            let outputTensor = try interpreter.output(at: 0)
            // Obtener y procesar la predicción
            let prediction = outputTensor.data.toArray(type: Float32.self)
            // Mostrar el resultado en la interfaz de usuario
            showPredictionResult(prediction: prediction)
```

```
        } catch {
            print("Failed to perform inference with error: \(error)")
        }
    }

    // Función para mostrar el resultado
    func showPredictionResult(prediction: [Float]) {
        // Actualizar la interfaz de usuario con el resultado de la predicción
    }
}
```

7. Fuentes de Datos

Información de los cultivos:

Papa: La información sobre las papas incluye detalles sobre su crecimiento y condiciones óptimas de cultivo. Las papas son tubérculos comestibles que crecen bajo la tierra y requieren suelos bien drenados y ricos en nutrientes. El ciclo de crecimiento de las papas abarca desde la preparación del suelo y la siembra, hasta la formación de tallos, ramas, hojas, y la fase de floración, donde los tubérculos acumulan agua, nutrientes y carbohidratos.

Zanahorias: Las zanahorias, una de las hortalizas más cultivadas en el mundo, crecen en cualquier tipo de suelo y son conocidas por su alto contenido en caroteno y vitaminas A, B y C. Su ciclo de crecimiento incluye la germinación de semillas, el desarrollo de cotiledones, y el crecimiento y desarrollo hasta la etapa de extracción.

Trufas: Las trufas son hongos que crecen en simbiosis con las raíces de ciertos árboles, como robles y encinas. Existen más de 70 especies de trufas, de las cuales 32 son europeas y sólo 30 son comestibles. El ciclo de crecimiento de las trufas comienza con la inoculación de esporas en el árbol huésped, seguido por la formación de una red de hifas (micelio) y finalmente la maduración de la trufa, que se cosecha con la ayuda de perros entrenados.

Creación de Dataset: Dado que no se encontraron datasets adecuados de imágenes en 3D de cultivos en Kaggle, se procedió a la creación de un dataset propio. Este dataset incluye imágenes de cultivos en diversos estados de crecimiento y condiciones, tomadas en diferentes iluminaciones, ángulos y fondos, y etiquetadas con nombres clave que indican estados de crecimiento, tipos de plagas y anomalías. Para la construcción del dataset en 3D, se utilizaron los programas de software VisualSFM y MeshLab. Se obtuvieron fotografías de los cultivos desde múltiples ángulos y se procesaron para generar nubes de puntos tridimensionales y mallas cerradas que representan los cultivos en 3D. Finalmente, se añadió textura y color a las mallas, produciendo representaciones tridimensionales detalladas.

Datasets Adicionales: Adicionalmente, se utilizaron datasets disponibles en la plataforma Kaggle para complementar la información del dataset propio. Estos datasets incluyen imágenes

etiquetadas de cultivos en diferentes estados de crecimiento, plagas en tierra y cultivos, y ejemplos de anomalías como deformidades, colores inusuales y daños físicos.

8. Instrumentos de recolección de datos

El instrumento es el dispositivo que se utilizará para la obtención de la información. Está íntimamente vinculado a la especialidad y a los estudios a realizar. En investigación en ciencias sociales los más usuales son la observación, la encuesta, el cuestionario, la entrevista, así como los dispositivos previstos para la realización de un experimento. Esto resulta pertinente ya que nuestro trabajo está orientado a la interacción con personas definidas dentro de una tipología con características específicas

A su vez, encontramos instrumentos específicos en diversas áreas del conocimiento, por ejemplo en nuestra disciplina: instrumental técnico, tecnológico, lenguajes y sistemas de programación, etc.

Eso significa que en primer término puede considerarse el instrumento que se prevé utilizar y secundariamente, a partir de las características de éste, estipular las fuentes de datos y los datos que se espera obtener.

9. Conclusiones

El presente trabajo se centra en resolver los desafíos del monitoreo de cultivos subterráneos como papas, zanahorias y trufas, utilizando tecnologías avanzadas como sensores, cámaras y reconstrucción 3D. Estas herramientas permiten a los agricultores acceder a información precisa sobre el estado de sus cultivos, detectar problemas a tiempo y tomar decisiones informadas para mejorar su producción.

Se pone especial énfasis en identificar y solucionar problemas como plagas, anomalías y deficiencias en los cultivos, lo que permite reducir pérdidas y optimizar recursos. Además, el sistema está diseñado para ser práctico y accesible, adaptándose a las necesidades específicas de cada agricultor y garantizando recomendaciones útiles y personalizadas.

En conclusión, este proyecto no solo busca mejorar la eficiencia en la producción agrícola, sino también fomentar prácticas sostenibles que beneficien tanto a los productores como a la sociedad. Su impacto potencial incluye mayor productividad, reducción de desperdicios y una agricultura más consciente y tecnológica.

10. Bibliografía

- Fan, W., Dong, J., Nie, Y., Chang, C., Yin, Q., Lv, M., Lu, Q., & Liu, Y. (2023). Alfalfa plant age (3 to 8 years) affects soil physicochemical properties and rhizosphere

microbial communities in saline–alkaline soil. Agronomy, 13(12), 2977. https://doi.org/10.3390/agronomy13122977

- Reddy, K. T. (2017). Técnicas de procesamiento de imágenes para la detección de formas de insectos en cultivos de campo. Conferencia internacional sobre computación e informática inventivas (ICICI) de 2017, 699-704.
- Larkin Alonso, J. (2022, junio 15). ¿Qué es Tensor Flow y para qué sirve? *Incentro*. Recuperado de https://www.incentro.com/es-ES/blog/que-es-tensorflow
- Martín Abadi, A. A. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Obtenido de Software available from tensorflow.org.: ww.tensorflow.org
- Grandón-Pastén, N., Aracena-Pizarro, D., & Tozzi, C. L. (2007). Reconstrucción de objeto 3D a partir de imágenes calibradas. *Ingeniare. Revista chilena de ingeniería*, *15*(2), 158-168. https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-33052007000200006
- Wood, Lucía (2015, revisión 2017). Guía para la elaboración y presentación del trabajo de investigación, Cátedra Metodología de la Investigación (FBA, UNLP). Buenos Aires.
- Samaja, J.; (1993); “Epistemología y Metodología”; cap. III y IV. Ed. Eudeba; Buenos Aires.
- Blandino, G. (2023, April 1). *Figma: Qué es y cómo funciona*. Pixartprinting. https://www.pixartprinting.es/blog/figma-que-es/
- Valencia, J. H. (2022). Construcción de una app nativa Android para la detección facial de emociones usando técnicas de inteligencia artificial. Pro Sciences: Revista de Producción, Ciencias e Investigación, 52-61.
- 3Dnatives. (2022, julio 7). Todo lo que necesitas saber sobre MeshLab. *3Dnatives*. Recuperado de https://www.3dnatives.com/es/meshlab-caracteristicas-070720222/
- Pereira, J. (s.f.). Introducción a la fotogrametría (Parte 1º). *JPereira.net*. Recuperado de https://www.jpereira.net/revisiones/software/introducicon-a-la-fotogrametria-parte-1o/