Argentina – Mendoza

Sistema de Verificación de Cortes con Realidad Aumentada aplicado en un Sistema de corte Robotizado con plasma

Año: 2022
Proyecto Final de Catedra

Autor: IGLESIAS Alfredo Andres
Legajo:10299
alfredoandresiglesias@gmail.com

Catedra: Realidad Virtual
Profesor: Lic. Javier J. Rosenstein

RESUMEN

Se presenta el siguiente informe del Proyecto Final para la catedra Realidad Virtual, con el fin de integrar y evaluar los conocimientos adquiridos a lo largo del dictado de catedra.

El proyecto tiene como objetivo lograr crear un Sistema para Verificar los cortes realizados en un marco de aplicación. Se utiliza como marco de aplicación de referencia, un sistema de corte por plasma mediante robot industrial. Este sistema debe lograr mantener al operario lejos de la zona de peligro y evitar que el mismo, toque el material cortado, al intentar verificar lo que corto mediante un instrumento de medición clásico.

En el desarrollo de la problemática planteada, se describirán los detalles del sistema y luego se dividirá en tres grandes etapas la resolución del problema. Primeramente, se encuentra el desarrollo de una aplicación para celular en la plataforma Unity. Que hace uso de realidad aumentada (RA), mediante el SDK Vuforia. Luego, se desarrolla la implementación del Sistema de Verificación de Corte. Esta se explica mediante la instalación del mismo, con los inconvenientes encontrados y pasos necesarios para su puesta en marcha.

Por último, se presenta la ejecución y resultados del sistema trabajando. Donde se podrán constatar si los objetivos del proyecto fueron resueltos de manera correcta. Anexo a esto, se presenta un video explicativo del Sistema de Verificación de Corte en funcionamiento.

ÍNDICE

1. INTRODUCCIÓN

En la actualidad, si se analizan los sistemas de verificación de corte mediante realidad aumentada, para corte por plasma automatizado, se pueden encontrar algunas alternativas como gafas de realidad aumentada para controlar el material que se está creando, si no para el control de calidad, al finalizar el proceso de construcción. Por otro lado, se pueden encontrar aplicaciones para celulares, como herramientas de medición, que tienen un buen desempeño. Pero no se ha logrado encontrar un sistema que combine ambas estrategias y que se logren implementar en una automatización.

Considerando lo recién mencionado y gracias al tener disponible un Robot FANUC S-420I F (1400kg mod:1996) en la empresa Log Metal, se plantea en mi proyecto final de carrera, generar un sistema de corte por plasma para vigas perfil H de 12m con el robot mencionado. Obteniendo con este robot los veneficios de robustez, precisión, repetibilidad y capacidad de trabajar en los 6 grados de libertad, del gran espacio de trabajo que dispone el robot.

Teniendo en cuenta el marco aplicativo, se presenta concretamente este trabajo, que tiene como objetivo lograr crear un Sistema para Verificar los cortes realizados por el robot. Este sistema debe lograr mantener al operario lejos de la zona de peligro y evitar que el mismo, toque el material cortado, al intentar verificar lo que corto mediante un instrumento de medición clásico.

En el desarrollo de la problemática planteada, se describirán los detalles del sistema y luego se dividirá en tres grandes etapas la resolución del problema.

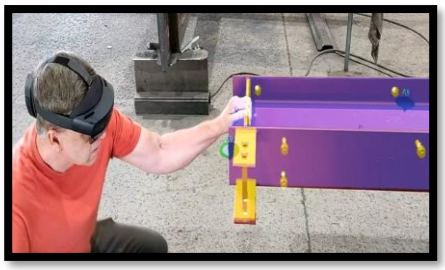
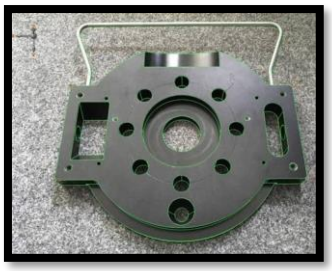
Primeramente, encontramos el desarrollo de una aplicación para celular en la plataforma Unity. Que hace uso de realidad aumentada (RA), mediante el SDK Vuforia. En esta sección se encontrarán todos los pasos necesarios para realizar la aplicación de celular entendiendo el flujo de la misma como el método de creación.

En una segunda etapa, se encontrará la implementación del Sistema de Verificación de Corte. Se abordará la instalación del mismo, con los inconvenientes encontrados y pasos necesarios para su puesta en marcha.

Por último, se presentará la ejecución y resultados del sistema trabajando. Donde se podrán constatar si los objetivos del proyecto fueron resueltos de manera correcta. Anexo a esto, se presentará un video explicativo del Sistema de Verificación de Corte en funcionamiento.

1.1. Antecedentes

Algunas de las aplicaciones existentes que se investigaron, se encontraban dentro del marco necesario. En la siguientes imágenes, se puede observar una aplicación para el control en tiempo de producción, así verificar que la viga sea construida de manera correcta. Luego se puede observar una aplicación para control de calidad de una pieza y finalmente una aplicación para obtener mediciones mediante realidad aumentada.


Analizando las aplicaciones que ya existen para verificar o generar un control de calidad mediante realidad aumentada. Se puede concluir que, si existe un mercado relacionado con el sistema que se pretende plantear, pero en su gran mayoría, estos sistemas mencionados, no están incorporados a las automatizaciones en las empresas. Si no que son una herramienta para el operario, pero no son parte de la automatización en sí misma, ya que esto contempla un aspecto mecatrónico más complejo.

1.2. Conceptos Teóricos necesarios para la Realidad Aumentada

Realidad Aumentada vs Realidad Virtual

De forma breve, la diferencia entre estas es que la realidad virtual (VR) genera mundos totalmente inexistentes mientras que la realidad aumentada (AR) combina elementos inexistentes con otros que sí están ahí. La primera busca sumergir al usuario por completo en un mundo virtual, permitiendo simular una experiencia sensorial completa dentro de un ambiente totalmente artificial. La segunda complementa el entorno real con objetos digitales. El usuario puede ver todo el ambiente que tiene alrededor, pero la pantalla utilizada le permitirá reproducir sobre este entorno objetos, animaciones y/o datos que no se encuentren en el lugar en dicho instante.

Unity un entorno 3D y 2D

Una parte primordial del proyecto era trabajar con los distintos sistemas de referencia en Unity y sus transformadas.

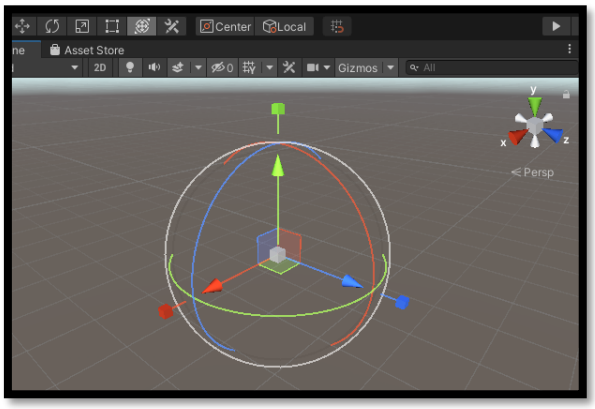


Figura 1: Transformaciones de la ubicación y orientación de objetos en Unity

Para la posición, se puede ubicar al objeto en el centro de los límites dictados por el objeto contenedor, o en uno de los pivotes. Para la rotación se puede tratar de transformaciones locales (relativas a sí mismas) o globales (espaciales en el mundo dado por un centro de coordenadas).

Componente Rect Transform: Es una herramienta muy utilizada y versátil. Permite posicionar el objeto en el espacio de manera tal, que pueda enlazarse con el árbol de prioridades. A demás provee de un sistema de anclaje, para referencia en algún extremo la posición 0 de donde parten las medidas, respecto al objeto en que este otro objeto este incluido.

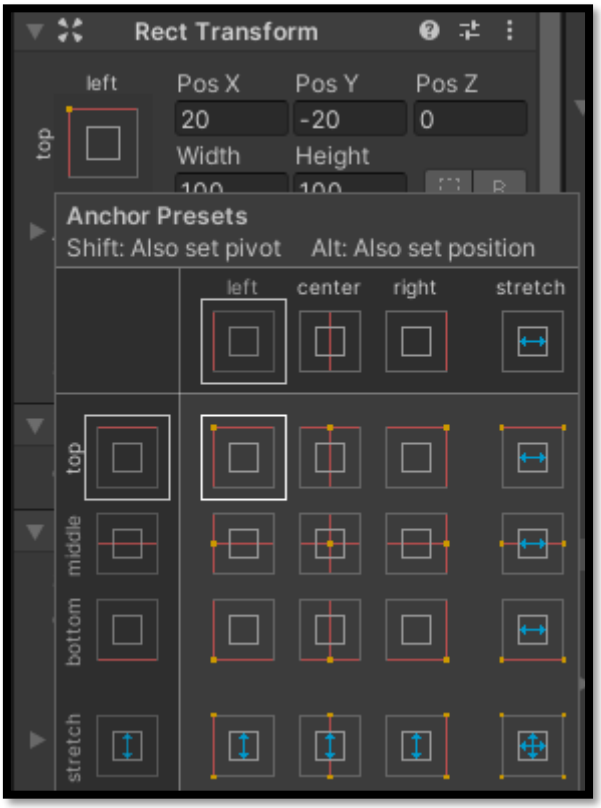


Figura 2: Componente Rect Transform de Unity

MATERIALES, SHADERS Y TEXTURAS

El renderizado en Unity usa Materiales, Shaders y Texturas que se definen de la siguiente manera:

- Materiales define cómo debe renderizarse una superficie, incluyendo referencias a las Texturas que utiliza, información de tiling, tintes de color y más. Las opciones disponibles para un Material dependen del Shader que el Material esté usando.
- Shaders son pequeños scripts que contienen los cálculos matemáticos y algoritmos para calcular el Color de cada píxel procesado, en función de la entrada de iluminación y la configuración del Material.
- Texturas son imágenes bitmap. Un Material puede contener referencias a las texturas, de modo que el Shader del Material pueda usar las texturas al calcular el color de la superficie de un GameObject. Además del color básico (Albedo) de la superficie de GameObject, las texturas pueden representar muchos otros aspectos de la superficie de un material, como su reflectividad o rugosidad

1.3. Descripción del Sistema

1.3.1. Objetivos

- Proveer un sistema de verificación para el corte que ya ha realizado por el operador, de esta manera, asegurar que el trabajo fue realizado correctamente.
- Trabajar con tolerancias por debajo o próximo a los (+- 3mm) de error al verificar los cortes.
- Mantener al operador en la zona segura de trabajo. Evitar que el operador se acerque al robot y que toque el material luego del corte con plasma.

1.3.2. Marco Aplicativo

Como ya se mencionó, el marco aplicativo del sistema Verificador de Cortes, se encuentra en un automatismo de corte por plasma mediante un robot industrial. Este consta de: Fuente de plasma, Robot Fanuc con su respectivo controlador RJ2, THC con su parte electromecánica de actuación acoplado en el extremo del robot y su gabinete de hardware y software en el Tablero de Control Plasma como se observa en la siguiente imagen.

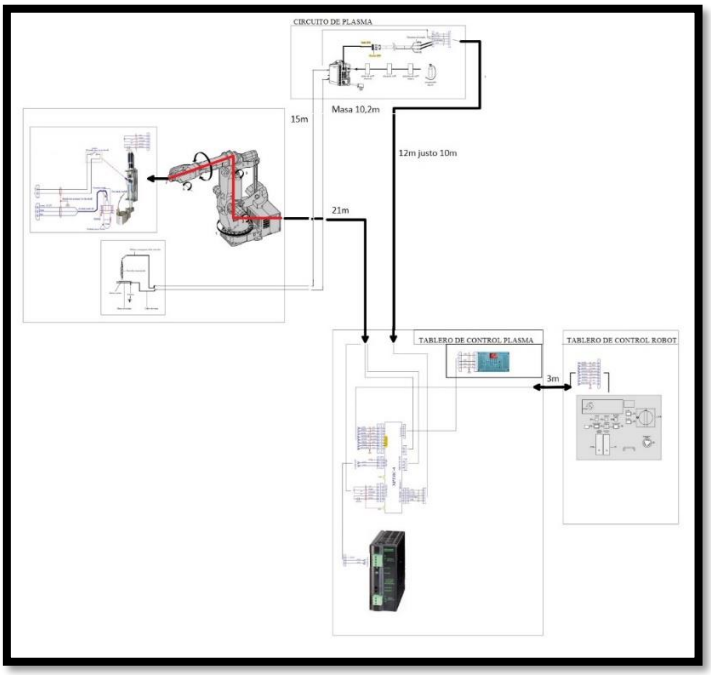


Figura 3: Croquis del automatismo, para el corte por plasma mediante robot industrial y sistema THC incorporado

1.3.3. Sistema de Verificación de Cortes

Para lograr la correcta implementación del sistema, se hizo uso de la Realidad Aumentada (RA). Esto implica la utilización de tecnologías y herramientas necesarias, para lograr su desarrollo y utilización.

A partir de los objetivos planteados, se propusieron las siguientes herramientas y dispositivos de Interfaz Humano Maquina (IHM). Los cuales se resumen en la siguiente imagen.

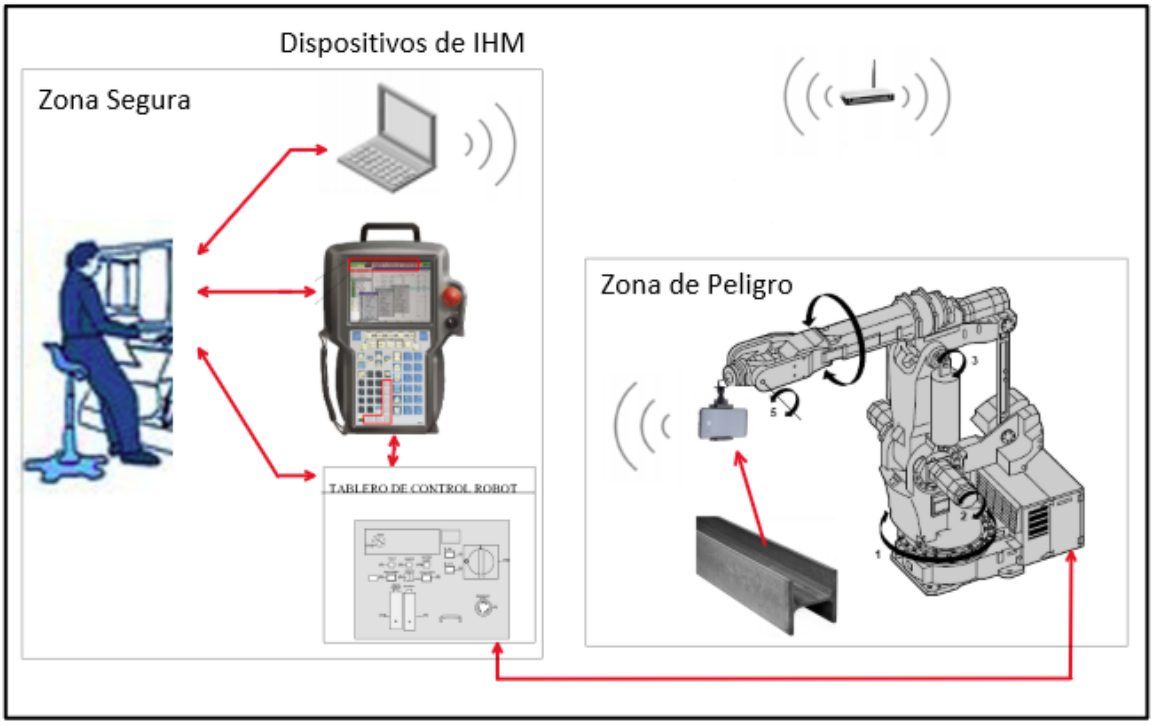


Figura 4: Sistema de Verificación de corte

Como se observa en la figura 3, el operario puede interactuar con los dispositivos IHM principales, que son el tablero de control del robot, para poder prender, apagar el mismo. También la terminal de enseñanza (Theach Pended), con la cual se generan los programas de movimiento del robot, y se ejecutan. Este tiene una interfaz con un teclado y una pantalla para observar los programas.

Por último, la notebook, la cual permite interactuar con el operador de manera visual con la pantalla y con el mouse. Esta se encarga de controlar y retrasmitir la pantalla del celular mediante la red WiFi que genera el ruter-modem. El cual este último se encuentra colocado en el extremo del robot. Encargado de cargar y procesar la aplicación android (APK).

Así de esta forma, el operario logra encontrarse siempre distante al robot en una zona segura. Como se observa, la interacción de los movimientos del celular, serán accedidos mediante la programación del robot, para lograr observar el material a cortar, mediante la cámara del celular. Así poder trabajar con las diferentes herramientas que la APK nos provee.

Dentro de la APK, se crearon tres diferentes escenarios de realidad aumentada (RA). El primero consta de RA con reconocimiento de QR, para proyectar sobre el material, un modelo del material virgen. Esto da la posibilidad de corroborar antes de efectuar el corte, de corroborar que sea el material virgen adecuado. El segundo escenario de RA, consta de las mismas herramientas, pero el modelo que se carga ahora es el del Diseño de Corte, es decir el diseño final del corte se quiere obtener. Para ello luego de generar el corte, se puede ingresar en este escenario, para corroborar el diseño obtenido. Cabe aclarar que estos dos primeros escenarios, son de verificación visual, no precisa.

Por último, un tercer escenario se planteó para efectuar medidas. En este se resalta la importancia del método utilizado para medir, que es el de planos paralelos. Por ende, el plano de la cámara, se debe encontrar paralelo al plano que uno quiere medir. Así trazando una línea de referencia con una medida conocida en el diseño, uno puede luego generar una línea a medir y obtener el resultado de una medición con mayor precisión

1.3.4. Hardware utilizado

- Robot Fanuc S-420I: de 6 grados de libertad
- Controlador RJ2 del robot: con su respectivo tablero de operador, terminal de enseñanza

(Teach Pended)
- Soporte de celular para motocicleta
- Viga perfil H previamente cortada
- Código QR impreso
- Computadora Notebook Lenovo ThinkPaD: Proyecta y controla el celular
- Mouse
- Rúter-Modem Wifi: Genera una red interna entre el celular y la computadora
- Celular Huawei P30 Lite: Donde procesa la aplicación y se retrasmite a la Notebook
  - Sistema operativo y procesador: Procesador: Kirin 710 Octa-core (4 x Cortex-A73 2.2 GHz + 4 x Cortex-A53 1.7 GHz)
  - Memoria: Memoria interna: 128 GB, Memoria RAM: 4 GB
  - Sistema operativo: Android 10
  - Cámara principal: 48 MP + 8 MP + 2 MP (f/1.8 + f/2.4 +f/2.4)
  - Pantalla: Tamaño de pantalla: 6,15", Resolución pantalla (pixels): FHD+ (1280 x 720) (415 PPI)

1.3.5. Software utilizado

- Unity: es un motor de videojuego multiplataforma disponible para Windows, OS X y Linux. La plataforma de desarrollo posee soporte de compilación con diferentes tipos de plataformas objetivo que van desde WebGL, computadoras con los sistemas operativos nombrados, dispositivos móviles android, iOS y Windows Phone entre otros, smart TVs, consolas de videojuegos y otros dispositivos de realidad virtual. Esto último, sumado a que es un software libre (versión estudiantil), la variada disponibilidad de bibliografía y foros que lo utilizan y principalmente la gran cantidad de paquetes adicionales disponibles como es el caso de Vuforia fueron los motivos de haberlo seleccionado.
- Visual Studio: es un editor de código fuente desarrollado por Microsoft. Es gratuito y de código abierto, con el cual se lograron armar los scripts necesarios para la aplicación en Android.
- Vuforia: es un SDK que permite construir aplicaciones basadas en la RA. Este SDK, ofrece la experiencia para reconocimiento de texto e imágenes con detección y rastreo simultáneo de Targets. La arquitectura de una aplicación genérica de Vuforia es la siguiente:
  - Cámara: debe asegurar que la imagen sea captada por el Tracker.
  - Target: son utilizados por el rastreador para reconocer un objeto en el mundo real. Entre los principales tipos encontramos imágenes que pueden ser fotos, hojas de revistas, logos, íconos, y elementos textuales.
  - Tracker: es el encargado de analizar la imagen de la cámara y detecta objetos del mundo real a través de los frame de la cámara con el fin de encontrar coincidencias en la base de datos.
- Windows 10: Sistema de proyección de pantalla mediante red wifi
- Android 10: Sistema de proyección de pantalla mediante red wifi

2. DESARROLLO

2.1. Desarrollo de la APK en Unity

En la presente sección se tratará todo lo concerniente a la creación de los distintos Paneles en Unity y la programación de los distintos scripts en C#. Para lograr una Interfaz de Usuario Grafica (GUI), amigable con el operador y fácil de trabajar.

2.1.1. Paneles y Flujo de la aplicación

El tronco principal de la APK consiste en cuatro Paneles:

- Panel Principal: Menú de entrada a la APK. Encargada de proveer un acceso correcto a los Paneles secundarios y otorgar una interfaz amigable

  - Panel secundario Verificar Material: aplicada mediante RA, con lectura del código QR, mostrando un diseño tridimensional del material virgen antes de cortar. Para lograr una corroboración visual del mismo.

  - Panel secundario Verificar Diseño de Corte: aplicada mediante RA, con lectura del código QR, mostrando un diseño tridimensional del material con el diseño ya cortado. Para lograr una corroboración visual del corte efectuado.

  - Panel secundario Medir: Utiliza la cámara del celular, para lograr efectuar medidas sobre la imagen que muestra. A partir de 2 líneas. Una línea de referencia que se traza con una medida que se ingresa. Para luego poder trazar una línea a medir. Así obtener por pantalla el resultado de la medida que se busca.

2.1.2. Metodología practica para la creación

Para lograr llevar a cabo el flujo de la aplicación recién mencionado, luego de varias iteraciones y trabajo arduo en comprender el funcionamiento de Unity, con el tratamiento de las primitivas de renderizado en 2D y 3D. Se llego al siguiente procedimiento.

Figura 5: Escena Pincipal RV5 OK en unity, con arbol de jerarquias

Primeramente, partimos de la creación de un Canvas 1, el cual se va a encargar de contener el Menú Principal y el Panel de Medir. Este Canvas 1, tiene la particularidad de que se encuentra renderizado por una Main Camera (llamado Camera en la figura 4), por ende, todo lo que este proyectado en el plano de esta misma, se podrá observar. Esto es debido a que dentro del Panel medir, se necesitara crear las líneas, y estas necesitan ser renderizadas. Esto nos permite utilizar objetos tanto de interfaz de usuario (UI), como objetos 3d.

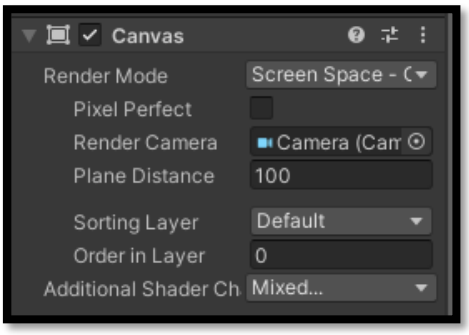


Figura 6: Configuracion de Canvas 1 para ser renderizado mediante Camera

Luego, se crea un Canvas 2, el cual contiene a los Paneles Verificar Material y Verificar Diseño de Corte. Este canvas tiene la particularidad de que se encuentra renderizado en el espacio de la misma ventana que ocupa el canvas. Por ende, cualquier objeto que no sea UI, no será observable. Este tipo de renderizado es necesario, para poder lograr obtener al momento de utilizar la RA de Vuforia, los componentes UI que provee el cambas, sin que se dejen de observar en la consola. Esto se debe a que, si se elige la ARCamera de Vuforia para renderizar el canvas, al momento en que se corre la aplicación, Vuforia cambia las coordenadas de orientación y posición de esta misma ARcamera, por ende, el canvas y los elementos UI que contienen se dejan de renderizar, porque no son observables.

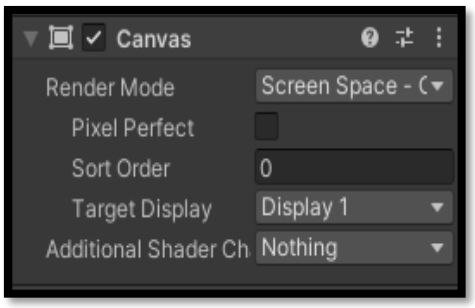


Figura 7: Configuracion de Canvas 2 para ser renderizado mediante el modo de Screen Space

2.1.3. Canvas 1: Menú principal y de Medir

Se procedió a crear el primer cambas, otorgándole la característica de resolución 1280x720, encontrando esta resolución equilibrada, tanto para el procesamiento del celular ya que, al agregar la RA, la capacidad de cómputo y el consumo de batería aumentan drásticamente.

Como ya se mencionó, el canvas 1 contiene a el Panel de Menú Principal y Medir. Estas se crearon mediante objetos UI, del tipo paneles.

Componentes del Panel Menú Principal:

Figura 8: Componentes del Menu Principal

Figura 9: Panel Menu Principal

- Botón Salir: Objeto tipo UI Imagen, el cual se carga la imagen del icono salir. Este tiene un script asociado, para que la aplicación se cierre en android. Utiliza una método sencillo del tipo Application.Quit().
- Logo: Objeto tipo UI Button, el cual se carga la imagen del mismo, posicionándolo correctamente en pantalla. Tiene la capacidad de al hacer click, se llama al script AbrirLogMetal, el cual redirecciona a la página de la empresa Log Metal. Lo hace mediante la creación de un método público, que utiliza el método Application.OpenURL

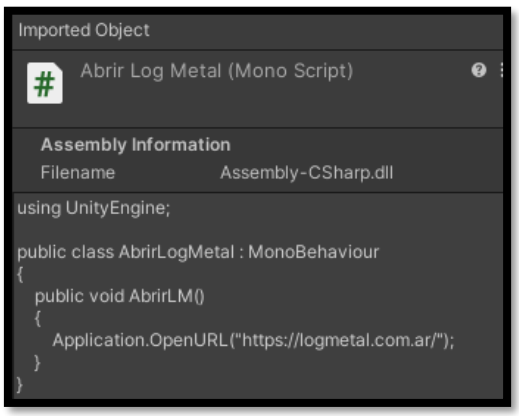


Figura 10: Script AbrirLogMetal

- Video Fondo: Objeto UI Raw Image. Este tipo de objeto, permite cargar una textura para ser renderizada dentro del camvas. Es un video que se proyecta sobre este canvas. Para lograr que se renderice este video. Para lograr esto fue necesario crear un Game Objet fuera del Canvas 1 dentro del árbol de prioridades (Figura 5), a este Game Objet, se le agrego el componente de Video Player, el cual permite cargar un video en formato .mp4 y genera las texturas del mismo (pero no lo reproduce en pantalla). Lego se enlazan ambos para lograr que Raw Imagen proyecte los renderizados que crea el Game Objet llamado ObjetoVideoRender.
- Nombre: Objeto tipo UI TextMeshPro. Permite crear el nombre de LOG Coutting Checker, de manera renderizada, utilizando varias herramientas de sombreado, y tipografías.
- Panel de botones: Objeto tipo UI Panel, el cual se le atribuye un componente llamado Vertical Layout Group, que nos permite localizar de manera simétrica los botones que se encuentran dentro.
- Botones: Todos son objetos tipo UI Botón. Al cual se les carga una imagen de fondo para botón rojo. Y se les atribuyen en las cualidades del componente Button de cambiar el color al presionar el botón, por un lado. Por otro lado, en la sección de activación cuando se hace click. Se agregan toda la lista de objetos principales de todo el árbol. Para luego utilizar el método GameObjet.SetActive la cual nos permitirá activar y desactivar los objetos que se ejecutan cuando la aplicación corre.

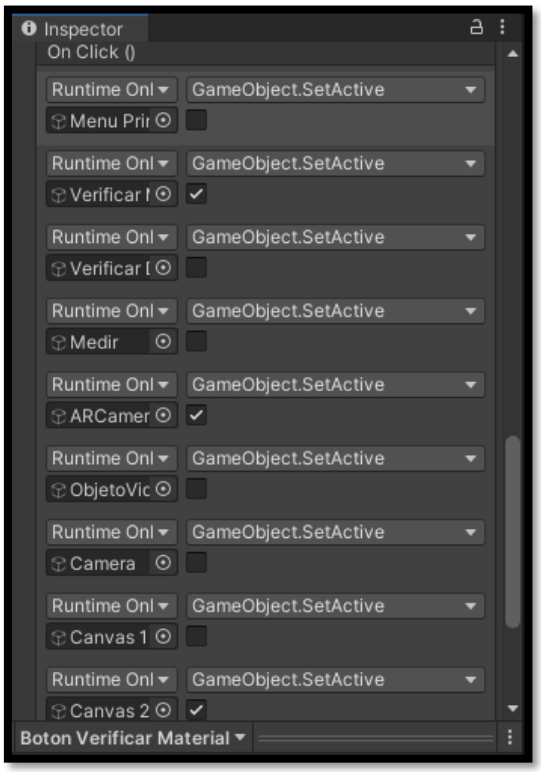


Figura 11: Metodos ejecutados al hacer click en un boton. Ejemplo para Boton Verificar Material

Componentes del Panel Medir:

Figura 12: Componentes del panel Medir

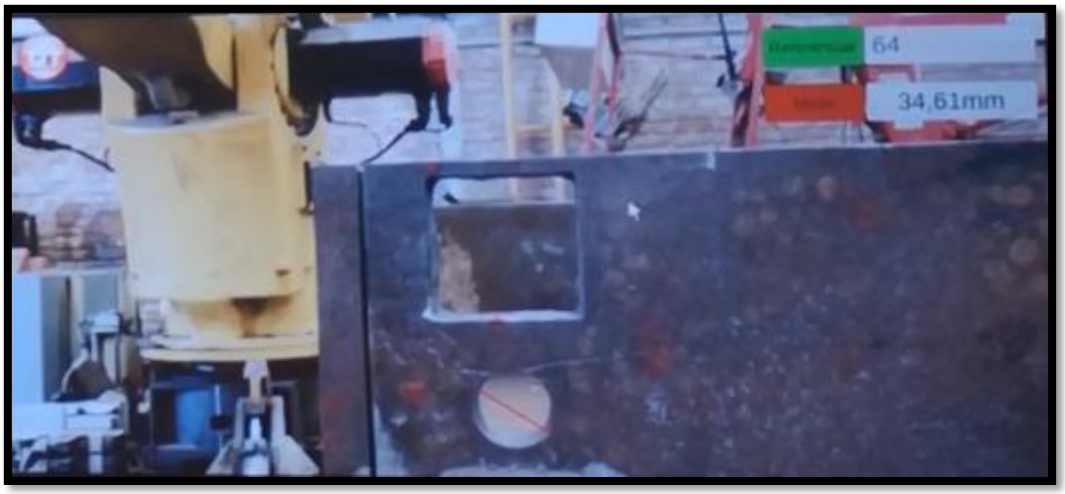


Figura 13: Panel Medir

- Medir: Objeto tipo UI Panel, el cual se le asigna un componente desde un script llamado Activar Camara 2. Este hace uso de las clases WebCamTexture, Texture, RawImage y AspectRatioFitter. Donde dentro de la inicialización de esta clase ActivarCamara 2, asigna las texturas que entrega la clase WebCamTexture mediante el dispositivo de la cámara encontrado (para el ancho y largo del canvas1) a el próximo objeto Background que es de tipo Raw Image. Este procedimiento es algo parecido al del video de fondo recién explicado, pero en vez de generar las texturas a partir de un video .mp4, se logra mediante la clase recién mencionada. Para que luego el objeto Raw Image renderice las texturas creadas. A demás se agrega en la actualización continua de void Update, la corrección de relación de aspecto y escala de pantalla, en caso de que se gire la misma. Luego se crean los métodos públicos prender y apagar cámara, para que dejen de utilizar el dispositivo, así no generar conflictos, cuando la RA, quiera utilizar este dispositivo

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

Script de Unity (7 referencias de recurso) | 0 referencias
public class ActivarCamara2 : MonoBehaviour
{

    private WebCamTexture webcamTexture;
    private Texture defaultBackground;

    public RawImage background;
    public AspectRatioFitter fit;
    Mensaje de Unity | 0 referencias
    private void Start()
    {
        defaultBackground = background.texture;
        WebCamDevice[] devices = WebCamTexture.devices;

        webcamTexture = new WebCamTexture(Screen.width, Screen.height);
        background.texture = webcamTexture;
        webcamTexture.Play();




    }
    Mensaje de Unity | 0 referencias
    private void Update()
    {
        float ratio = (float)webcamTexture.width / (float)webcamTexture.height;
        fit.aspectRatio = ratio;

        float scaleY = webcamTexture.videoVerticallyMirrored ? -1f : 1f;
        background.rectTransform.localScale = new Vector3(1f, scaleY, 1f);

        int orient = -webcamTexture.videoRotationAngle;
        background.rectTransform.localEulerAngles = new Vector3(0f, 0f, orient);
    }

    0 referencias
    public void PrenderCamara()
    {
        webcamTexture.Play();
    }
    0 referencias
    public void PararCamara()
    {
        webcamTexture.Stop();
    }


}
```

Figura 14: Script Activar Camara 2

- Background: Objeto tipo UI Raw Image, encargado de renderizar las texturas que provienen del objeto Medir, que accede mediante el script ActivarCamara2. De esta manera logra proyectar la imagen de la cámara en el canvas.
- Botón a menú de Medir: tiene las mismas cualidades y características de los botones ya mencionados. Simplemente que se le otorgan diferentes activadores de objetos, para que la navegación sea correcta cuando se tocan los botones en toda la aplicación.
- Botón Referencia: Este botón tiene las mismas característica que los anteriores, pero en el árbol de prioridades, contiene la activación y desactivación de los objetos que se contienen dentro de Botón Referencia y Botón Medir, para que alternen entre sí.
- Text: Objeto UI Text, se ingresa un texto sobre el Botón Referencia.
- Entrada Linea Referencia: Objeto UI Input Filed, encargado de ingresar por pantalla los valores de las medidas de referencia. Para lograr acceder a este valor en los otros objetos, se hace uso de un script ReadInput, el cual contiene un método público del tipo ReadStringImput, el cual asigna el strign ingresado por teclado a una nueva variable pública.

Read Input (Mono Script)

Assembly Information
Filename Assembly-CSharp.dll

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

public class ReadInput : MonoBehaviour
{
    public string medidaReferencia = "0 mm";
    // Start is called before the first frame update
    void Start()
    {


    }

    // Update is called once per frame
    void Update()
    {

    }

    public void ReadStringInput(string s)
    {
        medidaReferencia = s;
        //Debug.Log(medidaReferencia);
    }
}
```

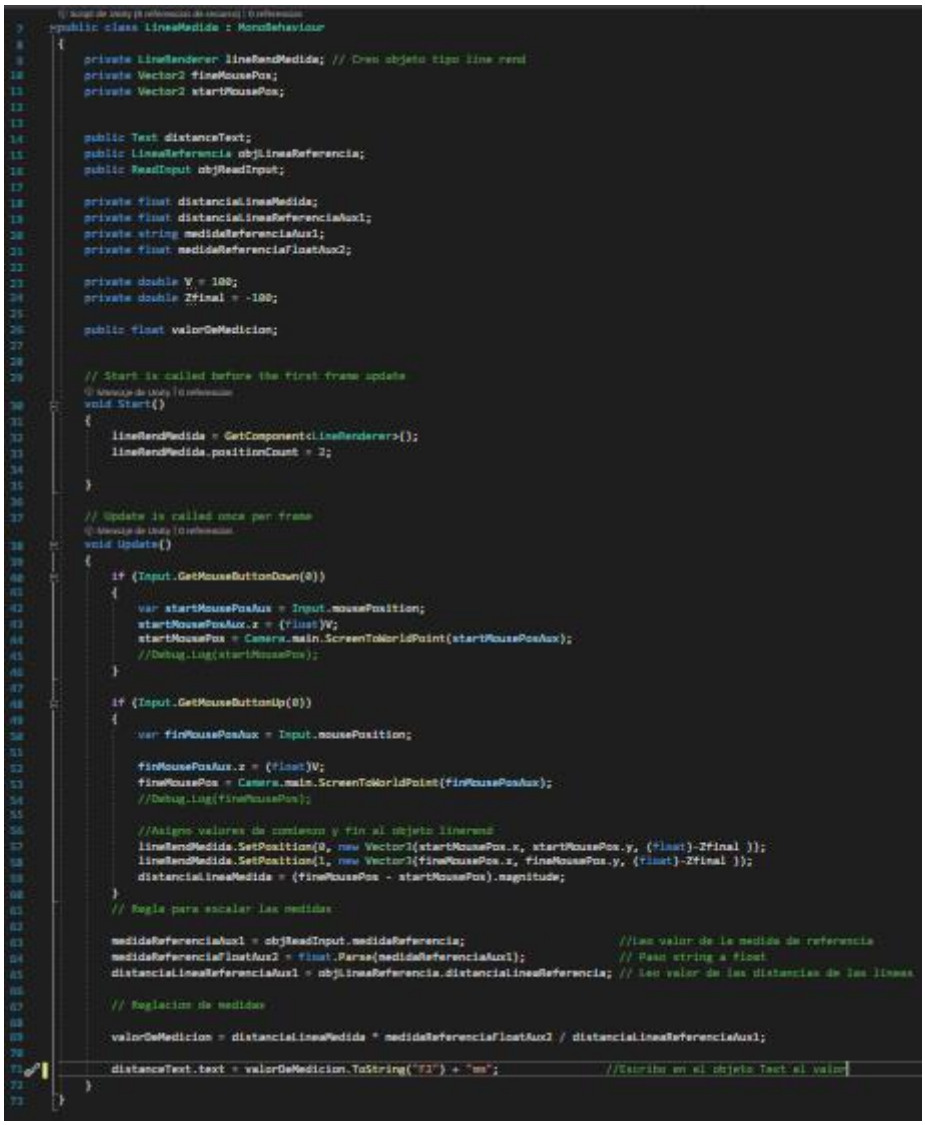
Figura 15: Script ReadInput

- Linea de Referencia: Game Objet creado con el atributo Line Renderer. El cual nos permite renderiza y crear una línea 3D sobre la pantalla. Tiene asociado como componentes dos script. Uno el script Linea Referencia. Este script hace uso de las clases LineRender y Vector2. El cual se encarga en la actualización de los frames dentro del Void Update, de estar monitoreando continuamente el click del mouse. Cuando detecta que baja el click, asigna las coordenadas donde este se encontraba respecto a la pantalla. Es decir, proyecta con la Camera, en la distancia que se encuentra el canvas, y transforma las coordenadas automáticamente. Para luego esperar a que se suelte el click generar el mismo procedimiento. Entonces con los dos puntos y sus coordenadas referidas al canvas, se asignan los métodos SetPosition de la LineRender generada. De esta forma se renderiza una línea verde sobre la pantalla a mostrar También calcula la magnitud de la distancia de la línea, haciendo la diferencia entre ambos puntos. Cabe resaltar la importancia de asignar una capa diferente a la capa (Layer) de UI, para que se puedan mostrar las líneas antes que el canvas.

```
Script de Unity (8 referencias de recurso) | 1 referencia
public class LineaReferencia : MonoBehaviour
{
    private LineRenderer lineRend;
    private Vector2 fineMousePos;
    private Vector2 startMousePos;

    private double V = 100;
    private double Zfinal = -100;


    public float distanciaLineaReferencia;

    // Start is called before the first frame update
    Mensaje de Unity | 0 referencias
    void Start()
    {
        lineRend = GetComponent<LineRenderer>();
        lineRend.positionCount = 2;
    }

    // Update is called once per frame
    Mensaje de Unity | 0 referencias
    void Update()
    {

        if (Input.GetMouseButtonDown(0))
        {
            var startMousePosAux = Input.mousePosition;
            startMousePosAux.z = (float)V;
            startMousePos = Camera.main.ScreenToWorldPoint(startMousePosAux);
            //Debug.Log(startMousePos);
        }

        if (Input.GetMouseButtonUp(0))
        {
            var finMousePosAux = Input.mousePosition;
            finMousePosAux.z = (float)V;
            fineMousePos = Camera.main.ScreenToWorldPoint(finMousePosAux);


            lineRend.SetPosition(0, new Vector3(startMousePos.x, startMousePos.y, (float)-Zfinal ));
            lineRend.SetPosition(1, new Vector3(fineMousePos.x, fineMousePos.y, (float)-Zfinal ));
            distanciaLineaReferencia = (fineMousePos - startMousePos).magnitude;
        }
    }
}
```

Figura 16: Scrip Linea de Referencia

- Boton Medir: idem Boton Referencia
- Text: idem Text de Boton Referencia
- Fondo para Texto: Objeto UI Image para dar un fondo de cierta trasparencia al texto.
- Distance Text: Objeto UI Text, es utilizado por el objeto Linea Medida que contiene el script Medida el cual se le asignara el atributo de este, para que sea escrito cuando se calcule la distancia.
- Línea Medida: Es un Game Objet con la mayoría de características que tiene Linea Referencia. Contiene. Contiene un script Linea Medida, el cual se encarga de la misma manera que el script Linea Referencia, de renderizar la línea roja para que se muestre por pantalla. Luego de esto para lograr calcular la distancia de esta línea nueva roja creada a partir de la línea de referencia verde. Es necesario traer los valores de estos objetos mediante sus atributos. Por eso se hace usos de las clases Text, LineaReferencia y ReadImput. De esta manera y transformado correctamente los tipo de datos Sting a float, se logra hacer la relación de medidas, para luego mostrarlas en el texto.



```
public class LineaMedida : MonoBehaviour
{
    private LineRenderer lineRendMedida; // Creo objeto tipo line rend
    private Vector2 finalMousePos;
    private Vector2 startMousePos;


    public Text distanceText;
    public LineaReferencia objLineaReferencia;
    public ReadInput objReadInput;

    private float distanciaLineaMedida;
    private float distanciaLineaReferenciaAux1;
    private string medidaReferenciaAux1;
    private float medidaReferenciaFloatAux2;

    private double V = 100;
    private double Zfinal = -100;

    public float valorDeMedicion;


    // Start is called before the first frame update
    void Start()
    {
        lineRendMedida = GetComponent<LineRenderer>();
        lineRendMedida.positionCount = 2;

    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            var startMousePosAux = Input.mousePosition;
            startMousePosAux.z = (float)V;
            startMousePos = Camera.main.ScreenToWorldPoint(startMousePosAux);
            //Debug.Log(startMousePos);
        }

        if (Input.GetMouseButtonUp(0))
        {
            var finMousePosAux = Input.mousePosition;

            finMousePosAux.z = (float)V;
            finalMousePos = Camera.main.ScreenToWorldPoint(finMousePosAux);
            //Debug.Log(finalMousePos);

            //Asigno valores de comienzo y fin al objeto linerend
            lineRendMedida.SetPosition(0, new Vector3(startMousePos.x, startMousePos.y, (float)-Zfinal ));
            lineRendMedida.SetPosition(1, new Vector3(finalMousePos.x, finalMousePos.y, (float)-Zfinal ));
            distanciaLineaMedida = (finalMousePos - startMousePos).magnitude;
        }
        // Regla para escalar las medidas

        medidaReferenciaAux1 = objReadInput.medidaReferencia;                   //Leo valor de la medida de referencia
        medidaReferenciaFloatAux2 = float.Parse(medidaReferenciaAux1);          // Paso string a float
        distanciaLineaReferenciaAux1 = objLineaReferencia.distanciaLineaReferencia; // Leo valor de las distancias de las lineas

        // Reglacion de medidas

        valorDeMedicion = distanciaLineaMedida * medidaReferenciaFloatAux2 / distanciaLineaReferenciaAux1;

        distanceText.text = valorDeMedicion.ToString("f1") + "mm";              //Escribo en el objeto Text el valor
    }
}
```

Figura 17: Script Linea Medida

2.1.4. Canvas 2: Verificar Material y Verificar Diseño de Corte

Como utilizar Vuforia en Unity

Para utilizar esta herramienta, se hico uso de los tutoriales que se encuentran en la bibliografía, se describirán los pasos muy brevemente, ya que estos son un procedimiento muy conocido y de facil acceso. Pero si se resaltara los métodos necesarios para implementarlo correctamente dentro de la APK, en las próximas etapas.

Por un lado, se necesita crear una cuenta en la página de Vuforia, para luego poder por un lado cargar el código QR seleccionado que luego es procesado por ellos. Luego descargar la base de datos necesaria para que, por medio de Target, se ingrese en el proyecto de Unity. Como así también es necesario en la ARCamera creada, ingresar la licencia de pruebas que otorga Vuforia.

El código QR seleccionado, es de un mapa de 8 bits, que Vuforia le otorgo 5 estrellas al reconocimiento.

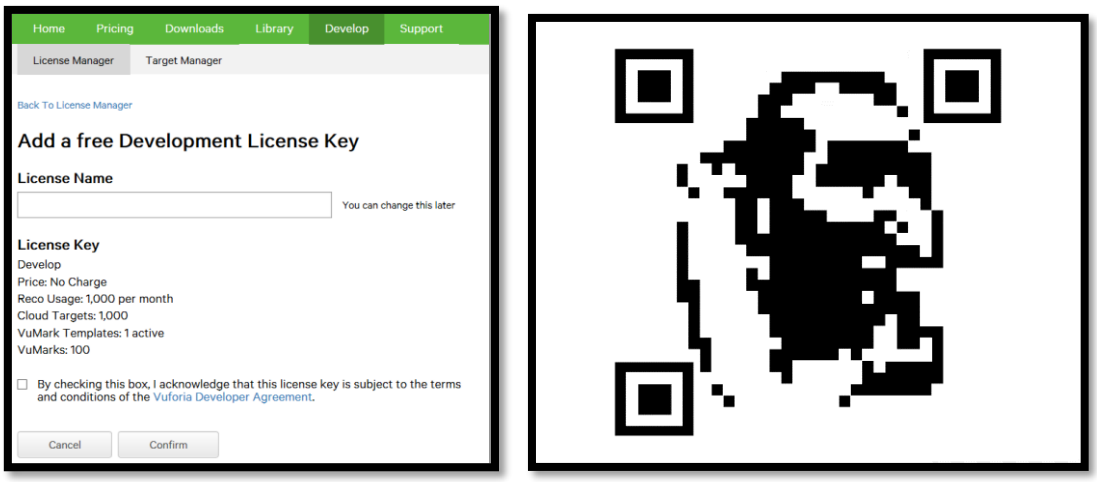


Figura 18: Web de Vuforia y Codigo QR utilizado

Componentes de Canvas 2

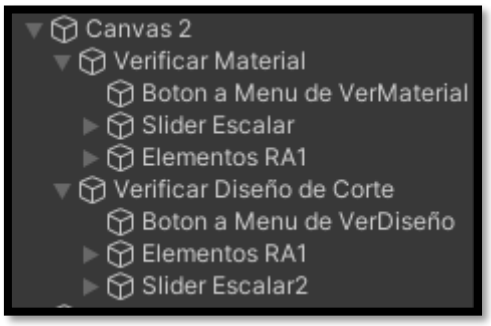


Figura 19: Componentes de Canvas 2

Cabe resaltar nuevamente que este Canvas 2, está siendo renderizado continuamente en la pantalla, a diferencia de Canvas 1, este no utiliza en los elementos UI cámaras para ser renderizado. Pero si se hace uso independientemente de la ARCamera de Vuforia.

Componentes del Panel Verificar Materia

Figura 20: Panel Verificar Material

Figura 21: Componentes Verificar Material

- Botón a Menú de Ver Material: es del mismo tipo ya mencionado
- Slider Escalar: Objeto tipo UI Slider. Encargado de generar una barra para escalar el objeto 3D de la RA en caso de que se necesite. Para lograr esto, se crea un script el cual se le adhiere como un componente. Este script que hace uso de las clases GameObjet para poder acceder al valor de la escala seleccionada y luego atribuirlo al modelo3D modificando su escala. Esto lo hace mediante un método publico Escalar Modelo

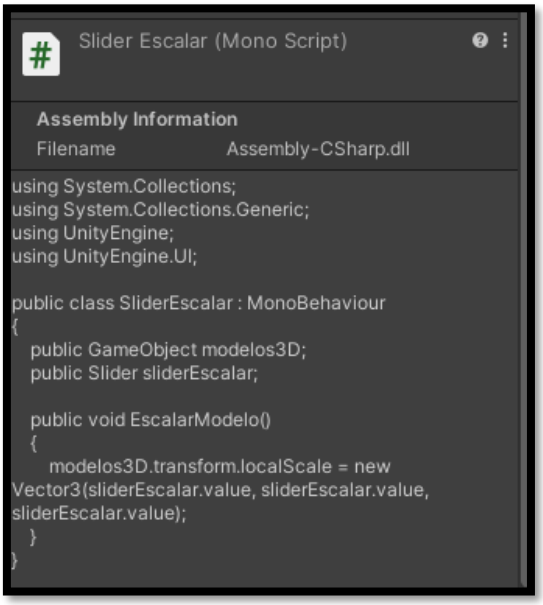


Figura 22: Script Slider Escalar

- Elementos RA1: Game Objet que contiene el la imagen target generado por Vuforia
- Imagen Target: Objeto tipo Image, que genera Vuforia, para colocar la imagen del código QR.
- Viga Virgen: Objeto 3D, es el modelo cargado previamente creado en el programa Solid Edge y exportado como archivo (.fbx). En las siguientes imágenes se pueden apreciar los mismos.

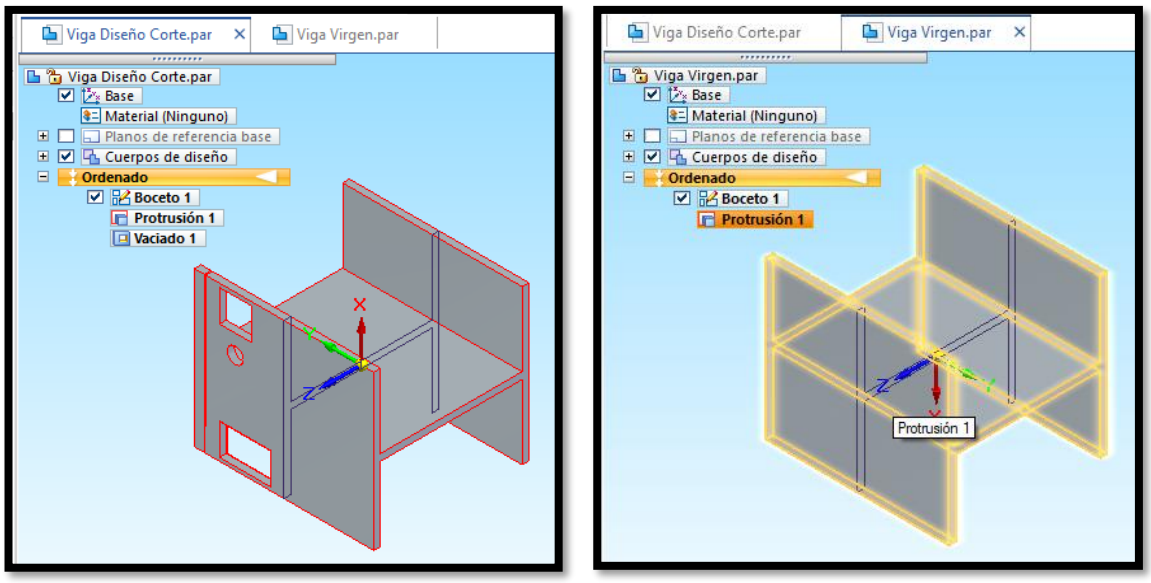


Figura 23: Modelos 3D de la Viga virgen y el Diseño de Corte generados en el programa Solid Edge

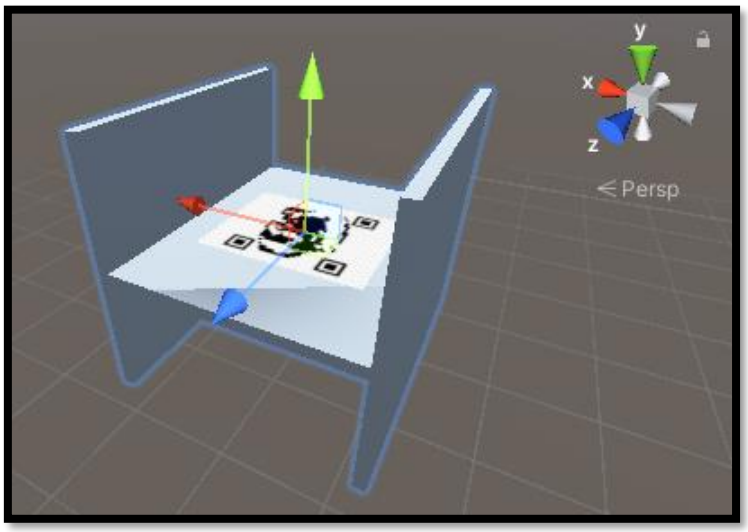


Figura 24: Imagen Target y Viga Virgen en la escena de Unity

Componentes del Panel Verificar Diseño de Corte

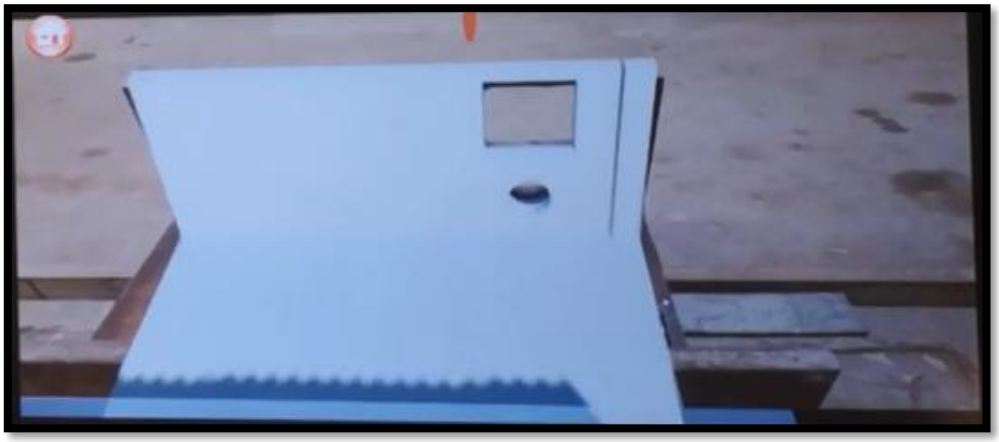


Figura 25: Panel Verificar Diseño de corte

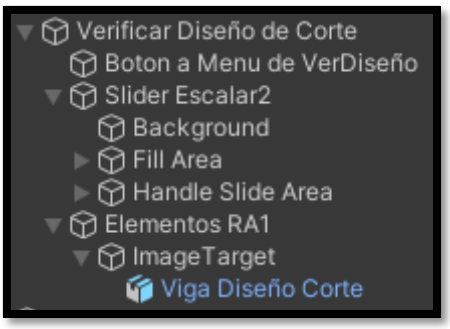


Figura 26: Componentes del Panel Verificar Diseño de Corte

Como se puede observar en las *Figura 25 y Figura 26* este Panel, es prácticamente una copia del Panel Verificar Medida. Solamente cambian algunos atributos al Botón a Menú de Diseño y el modelo 3D ahora pasa a ser el de Viga Diseño de Corte.

2.1.5. Árbol de Jerarquías completo

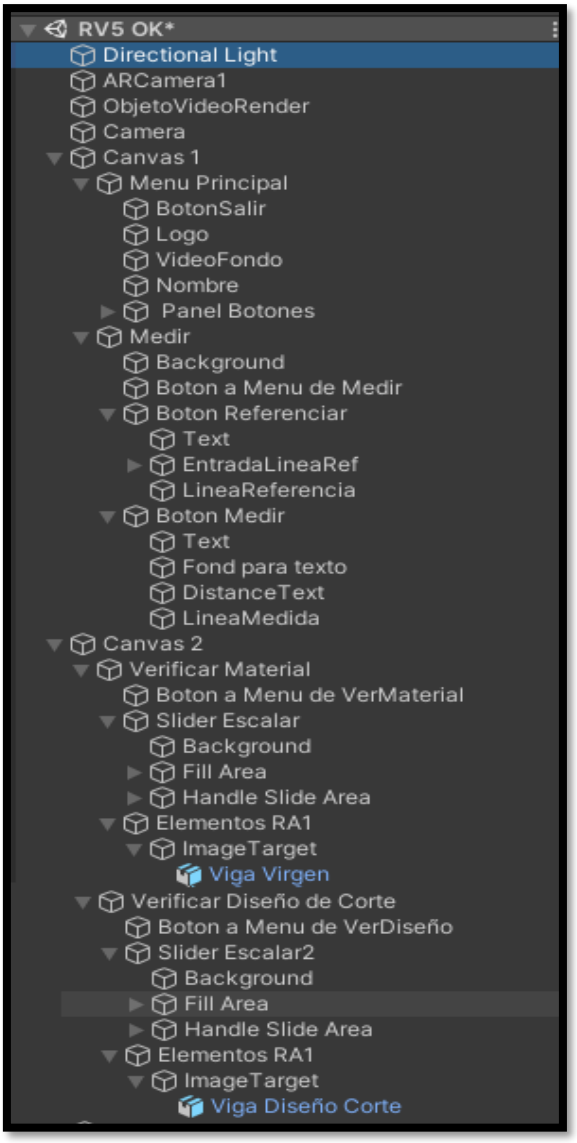


Figura 27: Arbol de jerarquias en unity con el Panel principal.

2.2. Implementación del Sistema

2.2.1. Instalación del Sistema de Verificación de Corte

En esta sección se podrá encontrar la instalación del Sistema de Verificación de Cortes, en la automatización. Como se puede observar en la siguiente imagen, con todos los elementos mencionados en la sección *1.3 Descripción del Sistema.*

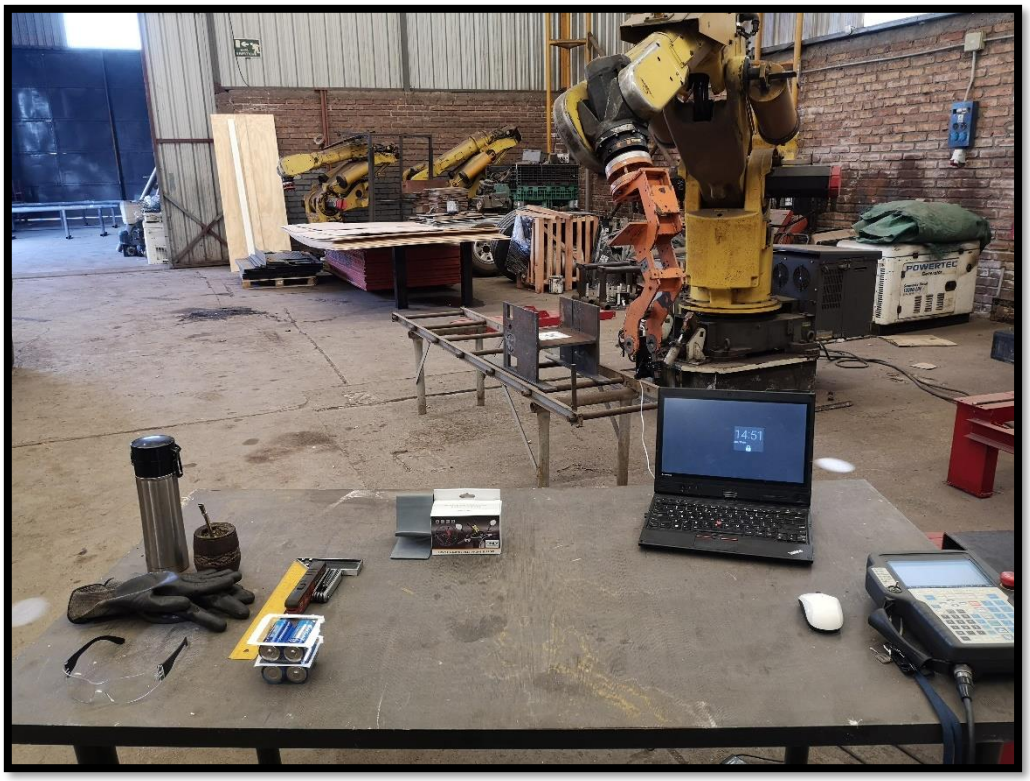


Figura 28: Instalacion del Sistema de Verificacion de Corte

Para lograr un correcto funcionamiento del sistema se resumen los pasos necesarios que fueron realizados:

- Puesta en marcha del robot: En esta instancia se necesitó generar un remasterizado del robot. Esto se debió a que el mismo tenía las pilas descargadas. Estas pilas se encargan de mantener en memoria tipo RAM, la información de los pulsos de los encoders. Por ende, al iniciar el robot, no permite moverlo. Los pasos necesarios para lograrlo, se encuentran en el manual de programación para Robot Fanucs, adjunto a este informe.

Figura 29: Pilas del robot de recambio y panel de control del operario para el robot

- Colocación del celular con la APK previamente instalada y corroborado su funcionamiento. Mediante el soporte de celulares.

- Impresión del código QR a escala 1:1, para luego colocarlo en la viga real.
- Instalación de la viga real que ya se encuentra cortada.

Figura 30: Viga y celular instalados

- Instalación y configuración de Modem/Ruter, generando una red interna, la cual permite controlar el celular mediante la notebook.
- Una vez instalados los anteriores accesorios, se procedió a generar el programa del robot, para que pueda ejecutarse. Este se logró llevando el extremo del robot, a los puntos necesarios, de una manera muy precavida. Las trayectorias usadas son del tipo lineales, asignadas con una velocidad de 500mm/s y de terminación tipo fina. Se resalta la importancia del momento en que se generan los puntos en que se va a Medir, es necesario orientar el celular con el robot para que coincida el plano del mismo con el de la viga donde se va a medir, como se puede observar en las siguientes imágenes. A demás se tomaron medidas, para lograr el nivel lateral.

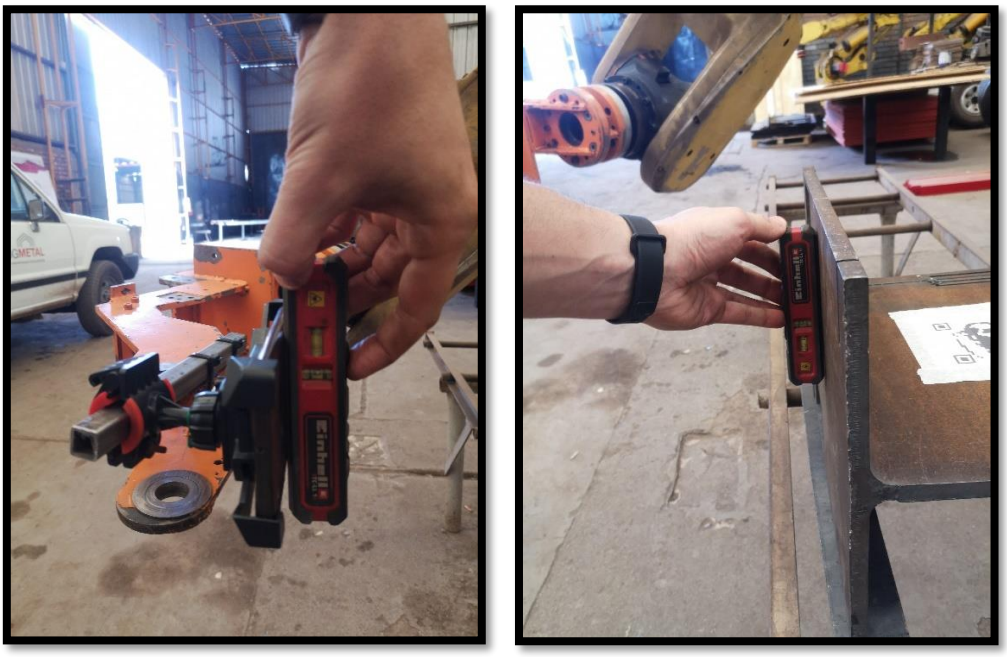


Figura 31: Nivelacion del plano de la camara respecto al plano a medir

2.3. Ejecución y Resultados Obtenidos

Luego de concluir la instalación correcta del sistema, y haber corroborado todo su funcionamiento. Se procedió a generar un ciclo de verificación sobre una viga de perfil H de 300mm de alma por 260mm de ala. La cual ya se encontraba cortada. Cabe aclarar nuevamente que el robot todavía no se encuentra actualmente cortando.

Se adjunta a este informe un video explicativo con la ejecución del sistema.

Luego de ejecutar el sistema, se puede constatar, visualmente, el trabajo de RA en los escenarios de Verificación de Material Virgen y Verificación de Diseño a Cortar. Como así también, en la sección de medición, se puede constatar que los valores obtenidos se encuentran por debajo de la tolerancia buscada en los objetivos de este proyecto. Estando en el orden de -+ 1mm.

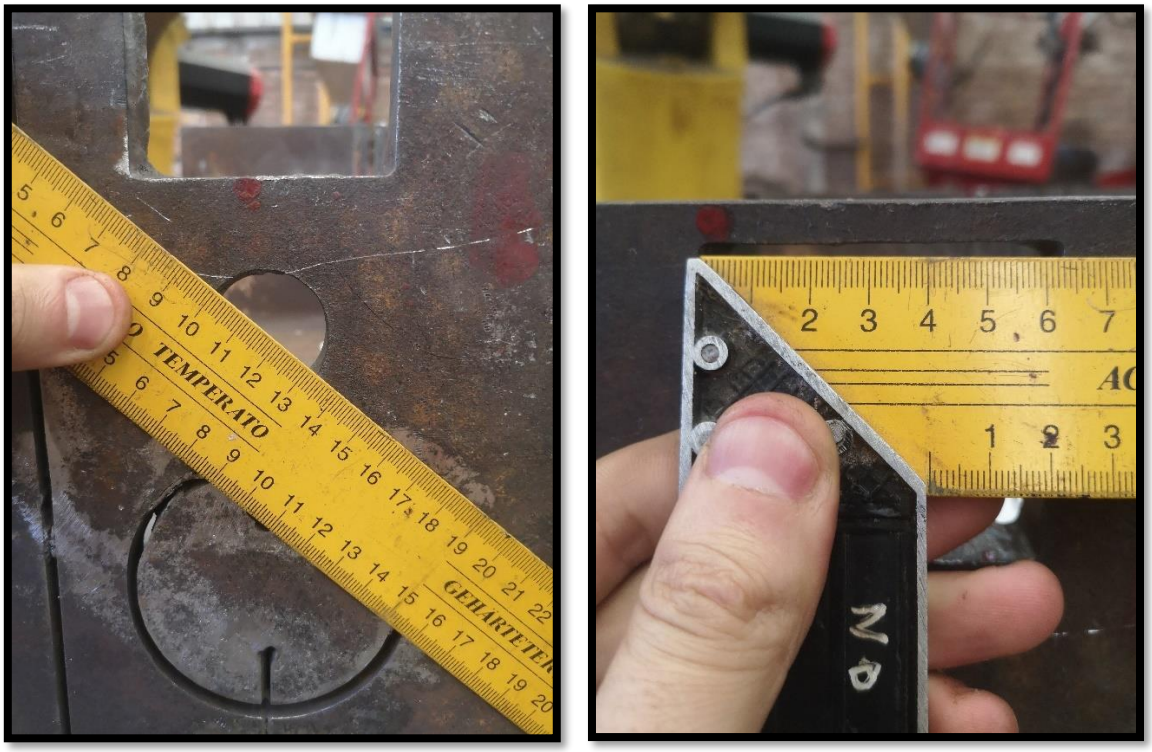


Figura 32: Medicion real de las medias tomadas mediante la aplicación.

Respecto a la interfaz y el árbol de funcionalidad de la aplicación, se encontró aceptable y cumple con los objetivos buscados.

3. CONCLUSIÓNES

Se evidencia a lo largo del informe el uso de una ¨cadena sistemática de razonamientos¨, con la cual se logró profundizar y aplicar todos los temas estudiados en la catedra. Esta perspectiva de razonamiento introdujo al alumno, a una metodología de trabajo y estudio valiosas para el futuro ingeniero.

Gracias a esto se alcanzaron los objetivos de lograr crear un Sistema de Verificación de Corte, para un robot. Por un lado, se resalta la importancia del uso de los conceptos Mecatrónicos relacionados al proyecto. Pudiendo encontrar desde el aprendizaje, programación y puesta en marcha de un robot industrial, como así también el trabajo de programación. Que mediante la plataforma Unity se resolvieron los objetivos relacionados con la interfaz de usuario gráfica y la aplicación de RA al proyecto. Cabe resaltar en la sección *2.1.2 Metodología practica para la creación,* que se logró gracias a reiteradas iteraciones y arduo estudio del problema, como así también esto dio un conocimiento más profundo la plataforma Unity. A demás se puede encontrar el logro de saber utilizar y comprender uno de los potentes softwares de realidad aumentada utilizados en la industria (Vuforia).

Luego de la implementación correcta del sistema, se logro ejecutar y efectuar las verificaciones de cortes necesarias para la el corte propuesto. Encontrando buenos resultados y mejores de los esperados por el alumno.

A pesar del buen desempeño del Sistema, el trabajo realizado evidencia la posibilidad de introducir otras mejoras. Por ejemplo, la inclusión de un mejor sistema de retrasmisión de pantallas y un celular con mayor capacidad de computo, ya que la latencia generada por estos mismos, dificulta la creación de líneas en el entorno Medir. De esta manera poder obtener resultados más precisos y próximos a la realidad en las mediciones.

Para concluir, se entiende que los objetivos alcanzados en el tiempo estipulado en el presente proyecto, resultaron satisfactorios para el alumno, de esta manera sentaron una base muy fuerte en el mundo de la Realidad Aumentada en el entorno industrial.

4. BIBLIOGRAFIA

- Material de catedra, Realidad Virtual 2019
- https://docs.unity3d.com/Manual/index.html
- https://answers.unity.com/index.html
- https://developer.vuforia.com/
- https://www.youtube.com/playlist?list=PLkbfAs8DnfZioidrCICnrGL2Q_uREwxxS
- https://assetstore.unity.com/orders/4673451238043/confirm
- https://www.youtube.com/watch?v=Ec1cV-ow4_g&ab_channel=AlexanderZotov
- https://www.youtube.com/watch?v=WBGY5TxmjrA&list=PLkbfAs8DnfZioidrCICnrGL2Q_uREwxxS&index=6&ab_channel=eLMformacion
- https://gamedevtraum.com/es/desarrollo-de-videojuegos/tutoriales-y-soluciones-unity/calcular-distancia-entre-dos-objetos-en-unity-espacio-y-plano/
- https://docs.microsoft.com/en-us/visualstudio/gamedev/unity/get-started/using-visual-studio-tools-for-unity?pivots=windows
- https://www.youtube.com/watch?v=CmPHg4XIChY&ab_channel=GameDevTraum