



UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES
Departamento de Electrónica – Año Lectivo 2019

PROYECTO FINAL

SmartSeedAnalyzer

Lucas Leiva	138408-9
Matías Pariente	135448-6
Gerardo Zoffoli	134524-2

Índice

AGRADECIMIENTOS	3
EQUIPO DOCENTE DE UTN FRBA	3
EQUIPO DOCENTE DE FACULTAD DE AGRONOMÍA (UBA)	3
ABSTRACT	4
RESUMEN EJECUTIVO	4
INTRODUCCIÓN TEÓRICA.....	5
¿QUÉ ES OPENCV?	5
¿POR QUÉ PYTHON?	5
¿POR QUÉ UTILIZAMOS LA <i>JETSON NANO DE NVIDIA</i> COMO KIT DE DESARROLLO?	6
PROCESAMIENTO DIGITAL DE IMÁGENES.....	7
ASPECTOS ECONÓMICOS	8
INTRODUCCIÓN	8
ACTUALIDAD DEL CLIENTE.....	8
COMPETENCIA EN ARGENTINA.....	8
SOLUCIONES ACTUALES (FUERA DE ARGENTINA).....	9
INSUMOS.....	10
ASPECTOS LEGALES	11
PATENTES	11
ANÁLISIS FODA	12
INTRODUCCIÓN	12
PLAN DE GESTIÓN DEL PROYECTO.....	14
GESTIÓN DEL ALCANCE	14
MATRIZ DE TRAZABILIDAD DE REQUISITOS.....	15
VERIFICACIÓN DEL ALCANCE	16
GESTIÓN DEL TIEMPO (GANTT)	17
GESTIÓN DEL RIESGO	18
GESTIÓN DE LA CALIDAD	19
DESARROLLO DE LA INGENIERÍA	20
ELECCIÓN DE LA CÁMARA Y TAMAÑO DE LA BANDEJA	20
SOFTWARE	21
PROCESAMIENTO DE IMÁGENES	22
INTERFAZ GRÁFICA.....	29
BASE LUMINOSA CON VIBRACIÓN	32
ESTRUCTURA	34
SIMULACIONES	36
MEDICIONES.....	38
BIBLIOGRAFÍA	42

Agradecimientos

Equipo docente de UTN FRBA

Queremos hacer extensivos nuestros agradecimientos al equipo docente de la cátedra de los viernes de la materia Proyecto Final, quienes en todo momento nos brindaron ayuda y consejos con la mejor predisposición. Los ingenieros Silvio Tapino, Juan Pablo González, Matías Hampel y Basilio Robinio.

En especial un afectuoso agradecimiento al Ing. Silvio Tapino de quien destacamos no solo sus cualidades técnicas/académicas sino también el aspecto humano que ha sido fundamental para nosotros al transitar el año 2019. ¡Gracias Silvio!

Equipo docente de Facultad de Agronomía (UBA)



A los docentes Ing. Agr. Dr. Daniel J. Miralles e Ing. Agr. Dra. Betina Kruk quienes con la mejor predisposición y amabilidad nos brindaron información, recomendaciones y atendieron siempre con una celeridad notable todas nuestras dudas/consultas. Un real placer haber transitado este camino con ellos. ¡Gracias Daniel y Betina!

Abstract

Resumen ejecutivo

La motivación principal del proyecto *SmartSeedAnalyzer* fue resolver la necesidad planteada por nuestro cliente, la Catedra de Cerealicultura y Fisiología Vegetal de la Facultad de Agronomía (UBA).

Esta necesidad radica en caracterizar mediante procesamiento digital de imágenes, es decir pesar, contar y determinar el área de distintas especies de semilla con fines académicos o de laboratorio. Por ejemplo, una forma de determinar el rendimiento de determinado cultivo es obtener la cantidad de semilla por metro cuadrado que dicho cultivo produce.

El cliente posee actualmente una balanza de precisión propia, con lo cual, se adaptó el software del *SmartSeedAnalyzer* para recibir por puerto serie el peso directamente obtenido con dicha balanza externa. También se deja como opcional la posibilidad de ingresar por teclado el peso de la muestra en caso de no contar con una balanza externa.

Además de obtener la cantidad de semillas de una forma rápida y precisa (contemplada en la especificación), también el cliente precisaba que el sistema sea capaz de determinar el área promedio de la muestra de semillas analizada. Con esto se buscaba medir, comparar y generar estadísticas que evidencien diferencias de áreas para distintas poblaciones de semillas de la misma especie. Por ejemplo, un cultivo de soja con agregados de fertilizantes y otros químicos en un tipo de suelo determinado puede producir semillas de tamaño diferente a otro cultivado bajo otras condiciones. El posterior análisis que el cliente realice con los datos obtenidos de cantidad y área con el *SmartSeedAnalyzer* corren exclusivamente por su cuenta.

El cliente también nos comentó que en una futura versión o upgrade del *SmartSeedAnalyzer* les resultaría útil contar con el ancho y largo de cada semilla (relación de aspecto promedio) para generar estadísticas de laboratorio para semillas de una misma especie.

En la actualidad, nuestro cliente realiza el conteo de semillas a mano (una por una) con lo tedioso que eso implica. Tiempo atrás contaban con un equipo muy sencillo que contaba las semillas lentamente una por una a medida que iban cayendo desde un pequeño plato giratorio atravesando un sensor óptico de cruce. Ese equipo está dañado y no se usa actualmente.

La solución que le hemos brindado a nuestro cliente ha sido alcanzada con un sistema integral de procesamiento de imágenes que consta de una estructura rígida con perfiles de aluminio que contiene la bandeja con retroiluminación led sobre la cual es analizada la muestra, una mini-PC Nvidia Jetson nano con sistema operativo basado en Linux (Ubuntu 18.04) en donde está instalado el software *SmartSeedAnalyzer*, una cámara digital de 16MP controlada por software, y dos motores internos cuya misión es vibrar la bandeja para esparcir la muestra de semillas.

Introducción teórica

El desafío y core de nuestro proyecto *SmartSeedAnalyzer* es el procesamiento digital de imágenes. Para ello se ha hecho uso de la librería multiplataforma open-source OpenCV y se eligió Python como lenguaje de programación interfaz para los scripts.

OpenCV y Python corren sobre un sistema operativo Linux (Ubuntu utilizando como hardware la mini-PC Nvidia Jetson Nano escogida para como kit de desarrollo.



18.04)
tal fin

¿Qué es OpenCV?

OpenCV (*Open Source Computer Vision*) es una librería software open-source de visión artificial y machine learning. Provee una infraestructura para aplicaciones de visión artificial.

La librería tiene más de 2500 algoritmos, que incluye algoritmos de machine learning y de visión artificial para usar. Estos algoritmos permiten identificar objetos, caras, clasificar acciones humanas en vídeo, hacer tracking de movimientos de objetos, extraer modelos 3D, encontrar imágenes similares, etc. Se utiliza en aplicaciones como la detección de intrusos en vídeos, monitorización de equipamientos, ayuda a navegación de robots, inspeccionar etiquetas en productos, etc. OpenCV está escrito en C++, tiene interfaces en C++, C, Python, Java y MATLAB interfaces y funciona en Windows, Linux, Android y Mac OS.



¿Por qué Python?

Python es muy sencillo de utilizar, favoreciendo el código legible gracias a su sintaxis sencilla. Debemos ser conscientes que el lenguaje nativo de OpenCV es C/C++, con la complejidad que ello conlleva si queremos utilizar esta biblioteca.



Algo destacable de Python es que es un lenguaje fácilmente portable a otras plataformas entre las que se incluye Raspberry Pi o Jetson Nano de NVIDIA.

¿Qué es gPhoto2?

gPhoto2 es un paquete gratuito, redistribuible, listo para usar, de aplicaciones de software de cámara digital para sistemas tipo Unix. Admite más de 2500 cámaras.



¿Por qué utilizamos la *Jetson Nano de NVIDIA* como kit de desarrollo?

En la etapa de iniciación de nuestro proyecto no teníamos bien claro qué tipo de hardware utilizar o como dimensionarlo de forma tal que el procesamiento de imágenes requerido por las librerías de OpenCV escritas en Python no exceda el límite de poder de cómputo de CPU y uso de memoria RAM. Las imágenes para procesar son de una resolución de 16MP tomadas con una cámara digital conectada a la mini-Pc, con lo cual el tiempo de procesamiento y uso de CPU/GPU es un aspecto fundamental para tener en cuenta (procesamiento multi core).



Indagando y estudiando algunos artículos en internet, optamos usar el kit de desarrollo Jetson Nano de NVIDIA ya que cuenta con un hardware que estimábamos cumplía de forma óptima nuestros requerimientos.



GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD (not included)
Video Encode	4K @ 30 4x 1080p @ 30 9x 720p @ 30 [H.264/H.265]
Video Decode	4K @ 60 2x 4K @ 30 8x 1080p @ 30 18x 720p @ 30 [H.264/H.265]
Camera	2x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI and display port
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I ² C, I ² S, SPI, UART
Mechanical	69 mm x 45 mm, 260-pin edge connector

Cabe destacar que, si el cliente utilizaba una Pc convencional de escritorio, no hubiera sido necesaria la elección de una mini-Pc, ya que hubiera cumplido sobradamente los requerimientos dado su CPU, disco y memoria RAM.

Procesamiento digital de imágenes

Este proyecto se basa en el procesamiento de imágenes digitales, es el conjunto de técnicas que se aplican con el objetivo de mejorar la calidad o facilitar la búsqueda de información que ellas aportan. En nuestro caso la información buscada es la cantidad de semillas y sus respectivas áreas.

Aspectos económicos

Introducción

Como en todo proyecto, es sabido que el aspecto económico es un ítem fundamental para garantizar la realización y viabilidad de este. Más aun cuando son requeridos componentes o diseños que no se pueden conseguir en el mercado interno.

Para el SmartSeedAnalyzer se intentó conseguir todos los componentes en Argentina. Esto ha sido posible a excepción de la mini-PC Nvidia Jetson Nano que fue conveniente comprarla en el exterior.

La cámara digital Nikon de 16MP es propia (usada) con lo cual no ha sido necesario incurrir en esa inversión.

Actualidad del cliente

El cliente actualmente se ve obligado a realizar el conteo de las semillas a mano. Tiempo atrás contaba con un equipo rudimentario que contaba las semillas lentamente una por una sobre un plato giratorio. Este equipo se dañó y no ha sido reparado nuevamente.



Competencia en Argentina

Según la búsqueda que realizamos por distintos medios y conversando también con el cliente, actualmente no hay en el país empresas establecidas que produzcan equipos de laboratorio o investigación que realicen caracterización de semillas mediante procesamiento digital de imágenes.


Soluciones actuales (fuera de Argentina)

El cliente nos informó acerca de la siguiente solución que puede importarse desde el exterior (Estados Unidos). El equipo se llama *OptiCount* y es desarrollado/fabricado por la empresa Process Vision (Richmond, Virginia, Estados Unidos).

Web: <http://processvis.com/>



El precio del equipo en Octubre 2018 era el siguiente (gastos de importación aparte):

		Process Vision, LLC 7006 Monument Avenue Richmond, VA 23226 804-514-9189 fax 804-673-0587		Quote No. PVQ101218A	
Client					
Name		CIMMYT		Date	
Address		Carretera Mexico-Veracruz Km 45		Attn	
City		El Batán		Terms	
Phone		Country Mexico		FOB	
				Richmond, VA, USA	
Qty	Description	Unit Price	TOTAL		
1	Process Vision, LLC is pleased to offer the following: OptiCount Lab Seed Counter - OCLG-MAX Single Camera Configuration. Unit will have one 5MP mono camera set up to inspect a stage area of about 9"x12" or 12"x16". Unit will include one Ohaus SJX622N_E lab scale with serial interface. A compact embedded PC will be mounted inside the cabinet. External connections will be provided for a VGA monitor and a USB keyboard and mouse. (Monitor, keyboard and mouse to be furnished by the customer, SOLA power stabilizer to be furnished by customer.)	\$17,514.00	\$17,514.00		

Como puede observarse, la catedra de FAUBA no puede afrontar dicho gasto dado que se encuentra fuera del presupuesto asignado.

Insumos

Nuestro objetivo con el SmartSeedAnalyzer fue lograr un equipo accesible y alcanzable tanto para nosotros como para FAUBA. La misma facultad UBA accedió a financiar los gastos implicados en la realización del proyecto.

A continuación, se detalla el listado de los insumos utilizados en el presente proyecto:

Detalle	Precio	Observación
Mini PC NVIDIA JETSON NANO	\$ 7.000	PC de desarrollo comprada afuera (opcional para el cliente)
Estructura de aluminio	\$ 10.000	Perfiles 2020 + Tuercas para armado
Plásticos yacrílico de base luminosa y vibradora	\$ 1.900	2KG de PLA filamento de impresión 3D yacrílico luminoso
Electrónica Base Luminosa y Vibradora	\$ 2.400	Motores de Vibración, tiras de led, circuitos para automatización, vibración iluminación, puerto serie y batería.
Fuente Alimentación 12V 10A	\$ 600	Fuente de alimentación para el equipo
Monitor Philips 223V5LHSB2 LCD 21.5"	\$ 9.000	Pantalla para operar el equipo desde la mini PC (opcional para el cliente)
Kit Teclado Y Mouse Inalámbrico Logitech Mk235	\$ 1.000	Periféricos para operar el equipo desde la mini PC (opcional para el cliente)

Aspectos legales

Patentes

Los docentes de FAUBA nos comentaron su idea de realizar un resguardo intelectual de nuestro proyecto generando una patente de utilidad compartida entre CONICET, UBA y UTN; y como autores todos los que estuvieron involucrados en el desarrollo. Para ello nos sugirieron que el nombre del SmartSeedAnalyzer debería cambiarse por uno no registrado, con lo cual, a fines de patentamiento futuro optamos por cambiarle el nombre a “Analizador automático de granos”.

Los logos de CONICET, UBA y UTN ahora figuran en la pantalla principal de la nueva versión de software del equipo.



Seguiremos en contacto con los docentes de FAUBA para que nos asesoren en los futuros pasos a seguir con el patentamiento del proyecto.

Análisis FODA

Introducción

El análisis FODA (test para conocer las fuerzas, oportunidades, debilidades y amenazas de un proyecto ante su competencia) es la base del diagnóstico de un plan de negocios, ya que sistematiza la información del proyecto y su entorno, la cual se utiliza para definir objetivos realistas y diseñar estrategias competitivas para alcanzarlos.



Fortalezas

- Contamos con el contacto de Leandro Di Matteo (GIAR), especialista en Visión Artificial, para posibles consultas.
- Disponibilidad de librerías open-source para procesamiento de imágenes tales como OpenCV
- Posibilidad de inversión para el desarrollo del proyecto por parte de la Facultad de Agronomía

Oportunidades

- No existen fabricantes en Argentina de estos tipos de productos.
- El producto similar (completo) tiene un costo de 18.000 U\$S.
- La Cátedra de Agronomía podría colaborar en ciertos costos del proyecto.

Debilidades

- No contamos con experiencia en procesamiento de imágenes.
- Deberemos realizar muchas pruebas por falta de experiencia.
- Necesidad de una estructura mecánica incluyendo una base con sistema de vibración.
- Calibrar la cámara, condiciones de iluminación, etc

Amenazas

- Posible costo de inversión alto en dólares, lo cual puede ser un problema en el contexto de la economía actual.
- El tiempo es un recurso escaso dada la actividad laboral y carga académica de los integrantes del grupo.

Desafíos tecnológicos

- Detectar semilla por semilla, incluso semillas muy pequeñas.
- Obtener el área de las semillas
- Detectar el solapamiento de semillas e intentar separarlas por vibración o bien indicar la zona de conflicto para evitar un procesamiento erróneo.
- Adecuada configuración del sistema cámara, muestra e iluminación.

Plan de Gestión del Proyecto

Gestión del alcance

Listado de requisitos solicitados por el cliente

- 1- Error de conteo < 1%
- 2- Error del área < 10%
- 3- Cantidad nominal de semillas: 1000
- 4- Peso mínimo de cada semilla: 1 mg
- 5- Altura mínima de cada semilla: 1 mm
- 6- Ancho mínimo de cada semilla: 1mm
- 7- Área máxima del total de semillas: 600 cm²
- 8- Peso máximo de la muestra: 300g (puede variar según la balanza)
- 9- Error en el peso: Depende de la balanza que se utilice.
- 10- Conexión para balanza: RS-232.
- 11- Sistema de Vibración para separar las semillas.
- 12- Identificación de semillas solapadas

Matriz de trazabilidad de requisitos

Matriz de Trazabilidad de Requisitos				17 de Mayo 2019		
Nombre del Proyecto: SmartSeedAnalyzer						
Descripción del Proyecto: Caracterización de distintos tipos de semillas a través de procesamiento de imágenes						
Identificación Requisito	Código	Descripción del Requisito	Objetivo	Importancia	fecha	Comentario
Error maximo en medicion de cantidad	1.0	El error maximo admitido en la cantidad es de 1%	Lograr una medicion con un error menor al 1%	Alta	5/17/2019	
Error maximo en medicion de area	2.0	El error maximo admitido del area es de 10%	Lograr una medicion con un error menor al 10%	Alta	5/17/2019	
Conexión a balanza	3.0	Interfaz serie para conexión de balanza externa	Poder conectar cualquier balanza de laboratorio	Media	5/17/2019	
Estructura	4.0	Estructura del equipo	Construccion de una estructura estable que alojara todos los componentes del equipo	Media	5/17/2019	
Sistema de Vibracion	5.0	Sistema de vibracion para la separacion de semillas	Lograr separar las semillas para facilitar la lectura y analisis	Media	5/17/2019	
Sistema de iluminacion	6.0	Sistema de iluminacion para vision correcta de las semillas	Lograr que la camara pueda ver de manera correcta las semillas.	Alta	5/17/2019	

Verificación del alcance

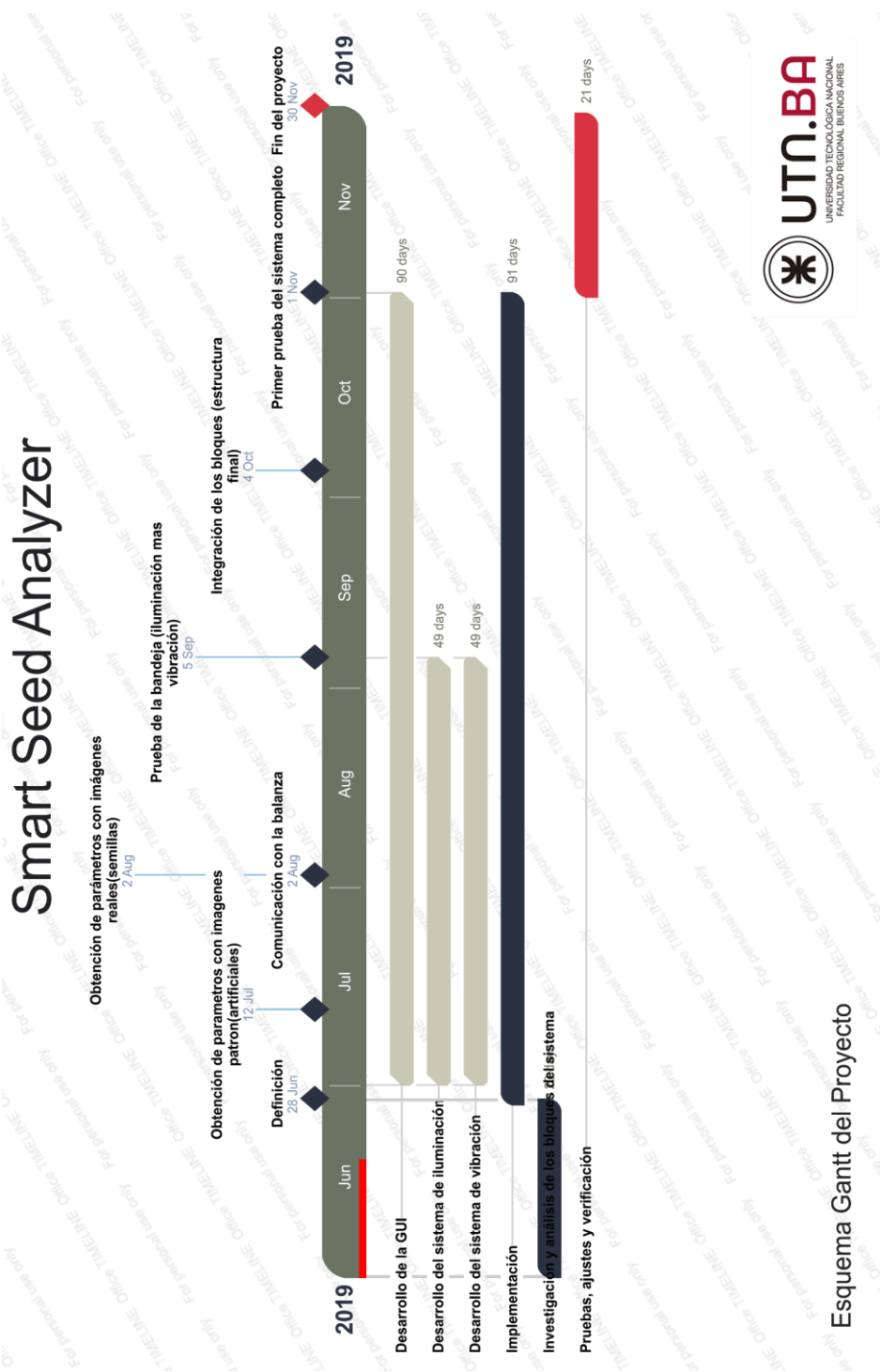
Primera etapa con imágenes Patrón

La primera etapa se realizará ingresando en el equipo imágenes patrón impresas, estas imágenes contendrán formas de semillas con cantidades y áreas determinadas (ya conocidas) de esta manera podremos verificar si el producto realiza la medición tanto de cantidad como de área de manera correcta cumpliendo con la especificación del producto. Estimamos contar con 5 imágenes patrón.

Segunda etapa con semillas reales

En la segunda etapa se realizarán varias mediciones (entre 5 y 10) de distintos tipos de semillas, repitiendo la misma medición para poder verificar la repetibilidad de la medición y que la variación que exista se encuentre dentro de las especificaciones dadas.

Gestión del tiempo (GANTT)



Gestión del Riesgo

Riesgos posibles considerados para nuestro proyecto

- 1- No llegar a realizar el proyecto en el tiempo estipulado.
- 2- Que durante el proyecto se presenten costos que excedan las expectativas.
- 3- No alcanzar los requerimientos propuestos en las especificaciones.
- 4- No lograr separar las semillas adecuadamente.
- 5- No poder identificar semillas solapadas.
- 6- Problemas para detectar las semillas y sus contornos en forma adecuada.
- 7- Inconvenientes para lograr resultados exitosos con semillas muy pequeñas.
- 8- No lograr resolver el problema de contraste entre el color de algunas semillas y el conjunto fondo e iluminación.
- 9- Dificultad para medir el área de cada semilla teniendo en cuenta una cantidad nominal de 1000 semillas y las características de estas.
- 10- Errores debidos a imperfecciones de las semillas.
- 11- Encontrarnos con problemas no previstos que no podamos resolver o excedan las posibilidades del grupo.

Gestión de la Calidad

- ✓ El producto contará con un **manual de ingeniería** donde se explicará con detalle el proceso de medición, como a través de una imagen podemos determinar la cantidad y el área de las semillas y se incluirá también el estudio estadístico de incertidumbre de los valores que mide el equipo.
- ✓ El producto contará con un **manual de usuario y de mantenimiento**, donde estará detallado todo el proceso de funcionamiento del equipo, desde el encendido hasta como realizar la medición. También contendrá información del mantenimiento del equipo tanto del ajuste de la estructura como la calibración del mismo, también contará con una sección de preguntas frecuentes con distintas situaciones problemáticas y la forma de resolverlas.
- ✓ El sistema contará con una interfaz gráfica amigable donde de manera simple el operario podrá ingresar los datos que necesiten ser suministrados, así como también podrá ver de forma simple los datos obtenidos de la medición.

Desarrollo de la Ingeniería

Elección de la cámara y tamaño de la bandeja

Se optó por trabajar con imágenes de alta resolución, para poder distinguir mejor los contornos de las semillas y medir con mayor precisión el área de estas.

Para dimensionamiento de la cámara y la bandeja, nos basamos en las semillas más pequeñas con un largo y ancho de hasta 1 mm según las especificaciones límites acordadas con el cliente.

El criterio fue contar con una resolución no menor a 10 pixeles/mm.

Se propuso usar una relación de aspecto de 4:3 para la bandeja ya que es soportada por la cámara y resulta práctica para el diseño y cálculos.

Para cumplir con la condición de 10 pixeles/mm se propuso una bandeja de 400 mm x 300 mm.

En cuanto a la cámara, se optó por una *Nikon Coolpix AW100* ya que tiene una resolución de 16 Mpx que cumple con lo requerido. Además, es compatible con la librería de software libre **gphoto2** para automatización y así permitir controlar la cámara desde nuestro Software.



Sensor de Imagen

Píxeles Efectivos (Megapíxeles)	16 millones
Sensor de Imagen	CMOS
Tamaño del sensor	1/2.3 pulg.
Píxeles totales	16.79 millones (aprox.)
Tamaño de imagen (píxeles)	4808 x 3456 (19M)

Lente

Lente	Lente de cristal NIKKOR ED con zoom óptico de 5x
Distancia Focal del Lente	5.0-25.0 mm (ángulo de visión equivalente al del lente de 28-140 mm en formato de 35 mm [135])
Número f/ del lente	f3.9-4.8
Construcción del Lente	12 elementos en 10 grupos
Zoom del Lente	5x
Zoom Digital	Hasta 4x (ángulo de visión equivalente al del lente de 560 mm en formato de 35 mm [135])
Reducción de la Vibración (VR)	VR por desplazamiento del lente
Apertura	Selección de filtro ND (-2 EV) controlada electrónicamente

Software

Automatización de la cámara

El siguiente script es el utilizado para realizar la captura y guardado de fotos desde la cámara digital *Nikon Coolpix AW100*

#Lista los vfs en uso y desmonto los de gphoto2 para poder utilizar la cámara

```
gvfs-mount -l
```

```
gvfs-mount -s gphoto2
```

```
gphoto2 --auto-detect
```

#Hace que la imagen se guarde en RAM de la camara (no usa la SD)

```
gphoto2 --set-config capturetarget=0
```

#Usa la SD de la camara

```
gphoto2 --set-config capturetarget=1
```

#Captura y guarda la imagen en el path especificado (sobre-escribe si existe el mismo archivo)

```
gphoto2 --capture-image-and-download --filename 'Capturas/myphoto.jpg' --  
force-overwrite
```

Procesamiento de imágenes

Para el procesamiento digital de las imágenes de las semillas se ha desarrollado un script en Python (archivo .py) que invoca a las librerías de OpenCV por cada tipo de semilla requerido por el cliente. Se ha optado por esta solución porque el tamaño, color, y forma de cada semilla hace que la parametrización difiera entre una especie y otra (por ejemplo: umbrales, filtros, etc).

El procesamiento digital de las imágenes involucra varias instancias las cuales serán descriptas de modo teórico a continuación.

Importación de librerías OpenCV necesarias para el procesamiento

Comenzamos importando nuestros paquetes necesarios.

```
# import the necessary packages
from skimage.feature import peak_local_max
from skimage.morphology import watershed
from scipy import ndimage
import numpy as np
import argparse
import cv2
import math
import os
```

Las líneas siguientes luego analizan nuestros argumentos en línea de comando. Necesitaremos un solo switch aquí, --image, que es el path a la imagen que queremos procesar:

```
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True, help="path to input image")
args = vars(ap.parse_args())
```

Lectura de la imagen como argumento

Procederemos a cargar nuestra imagen de las semillas desde el disco (previamente capturada por la cámara digital), aplicaremos el filtro de desplazamiento medio piramidal para ayudar a la precisión de nuestro paso de umbral.

```
# load the image and perform pyramid mean shift filtering to aid the thresholding step
image = cv2.imread(args["image"])
(h,w)=image.shape[:2]
center=(w/2,h/2)
M2=cv2.getRotationMatrix2D(center,180,1)
image=cv2.warpAffine(image,M2,(w,h))
imagesal=image
```

```
image=cv2.bitwise_not(image)  
shifted = cv2.pyrMeanShiftFiltering(image, 21, 51) #original tenia 21,51
```

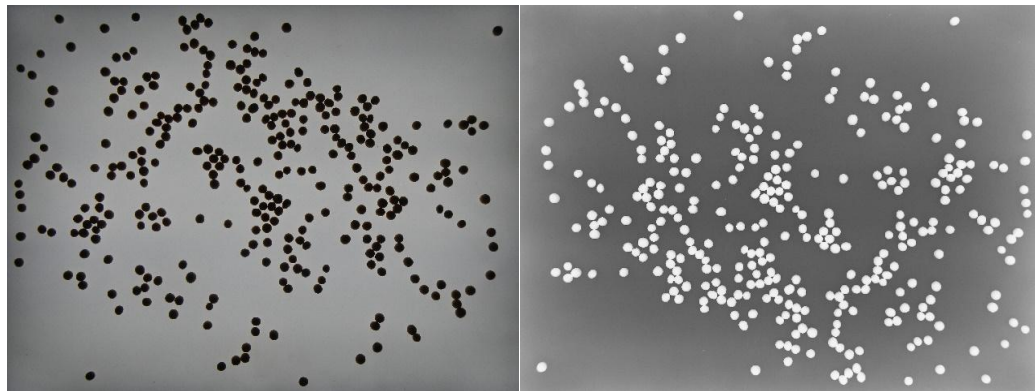
Cambio de la imagen a escala de grises

La imagen será convertida desde la original hacia una en escala de grises como se muestra a continuación. La información del color no es necesaria para los procesamiento subsiguientes ni tampoco aporta otro tipo de información útil.

```
# convert the image to grayscale  
gray = cv2.cvtColor(shifted, cv2.COLOR_BGR2GRAY)
```

Imagen original

Imagen en escala de grises

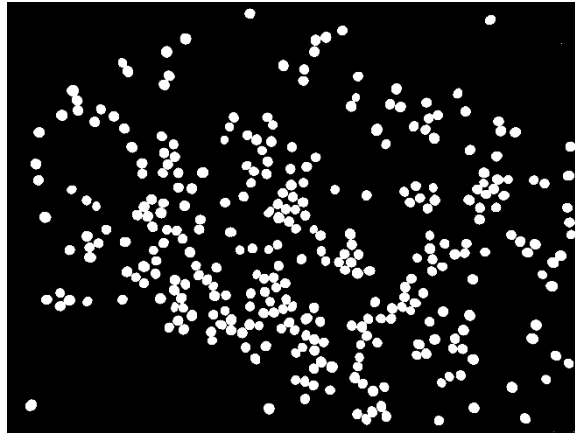


Aplicación de thresholding en la imagen

Si el valor del píxel es mayor que un valor umbral, se le asigna un valor (puede ser blanco), de lo contrario se le asigna otro valor (puede ser negro). La función utilizada es **cv2.threshold**. El primer argumento es la imagen de origen, que debería ser una imagen en escala de grises. El segundo argumento es el valor umbral que se utiliza para clasificar los valores de píxeles. El tercer argumento es maxVal, que representa el valor que se debe dar si el valor de píxel es mayor que (a veces menor que) el valor umbral. OpenCV proporciona diferentes estilos de umbral y se decide por el cuarto parámetro de la función.

```
# apply Otsu's thresholding  
thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]
```

A continuación, luego de aplicar el thresholding podemos observar cómo se logra “diferenciar” las semillas respecto de lo que es el background (fondo). El resultado es una imagen binaria (negro es fondo y blanco es semilla):



Aplicación de Transformada de Distancia

El operador transformada de distancia generalmente toma imágenes binarias como entradas (como la del ítem anterior). En esta operación, las intensidades de nivel de gris de los puntos dentro de las regiones de primer plano se cambian para diferenciar sus distancias respectivas del valor 0 más cercano (límite). Este es uno de los primeros pasos para iniciar la segmentación (separación) de los objetos en la imagen.



Suavizado de la imagen con filtro Gaussian Blur

Como en cualquier otra señal, las imágenes también pueden contener diferentes tipos de ruido, especialmente debido al sensor de la cámara. Las técnicas de suavizado de imagen ayudan a reducir el ruido. En OpenCV, el suavizado de imagen (también llamado desenfoque) se puede hacer de muchas maneras.

Los filtros gaussianos tienen las propiedades de no tener sobreimpulso a una entrada de función escalonada mientras minimizan el tiempo de subida y bajada. En términos de procesamiento de imágenes, los bordes afilados de las imágenes se suavizan y minimizan el desenfoque.

OpenCV proporciona la función *cv2.gaussianblur ()* para aplicar el suavizado gaussiano en la imagen de origen de entrada. La siguiente es la sintaxis de la función *GaussianBlur ()*:

<pre>dst = cv.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType=BORDER_DEFAULT]]])</pre>	
Parameter	Description
src	input image
dst	output image
ksize	Gaussian Kernel Size. [height width]. height and width should be odd and can have different values. If ksize is set to [0 0], then ksize is computed from sigma values.
sigmaX	Kernel standard deviation along X-axis (horizontal direction).
sigmaY	Kernel standard deviation along Y-axis (vertical direction). If sigmaY=0, then sigmaX value is taken for sigmaY
borderType	Specifies image boundaries while kernel is applied on image borders. Possible values are : cv.BORDER_CONSTANT cv.BORDER_REPLICATE cv.BORDER_REFLECT cv.BORDER_WRAP cv.BORDER_REFLECT_101 cv.BORDER_TRANSPARENT cv.BORDER_REFLECT101 cv.BORDER_DEFAULT cv.BORDER_ISOLATED

A continuación, se puede apreciar una imagen antes y después de ser aplicado el *GaussianBlur*:



Aplicación del algoritmo Watershed

Watershed es un algoritmo clásico utilizado para la segmentación y es especialmente útil cuando se extraen objetos que se tocan o se superponen en imágenes, como las semillas en nuestro proyecto.

Utilizando métodos tradicionales de procesamiento de imágenes, como la detección de contornos y umbrales, no podríamos extraer cada semilla individual de la imagen, pero al aprovechar el algoritmo de la cuenca hidrográfica, podemos detectar y extraer cada moneda sin ningún problema.

Cuando se utiliza el algoritmo watershed, debemos comenzar con marcadores definidos por el usuario. Estos marcadores pueden definirse manualmente mediante apuntar y hacer clic, o podemos definirlos de forma automática o heurística utilizando métodos como la umbralización y / o las operaciones morfológicas.

Basado en estos marcadores, el algoritmo trata los píxeles en nuestra imagen de entrada como elevación local (llamada topografía): el método “inunda” los valles, comenzando desde los marcadores y avanzando hacia afuera, hasta que los valles de los diferentes marcadores se encuentran entre sí. Para obtener una segmentación precisa con watershed, los marcadores deben colocarse correctamente.

Máximos locales

Primero tomamos D, nuestro mapa de distancia generado anteriormente con la transformada de distancia, y encontramos picos (es decir, máximos locales) en el mapa. Nos aseguraremos de que haya al menos una distancia de 20 píxeles entre cada pico.

```
localMax = peak_local_max(D, indices=False, min_distance=20, labels=thresh)
```

La variable localMax toma la salida de la función peak_local_max y aplica un análisis de componentes conectados usando la conectividad 8.

```
markers = ndimage.label(localMax, structure=np.ones((3, 3)))[0]
```

La salida markers de esta función nos da nuestros marcadores para luego alimentar al algoritmo watershed. Dado que el algoritmo supone que nuestros marcadores representan mínimos locales (es decir, valles) en nuestro mapa de distancia, tomamos el valor negativo de D, es decir, -D.

```
labels = watershed(-D, markers, mask=thresh)
```

La función watershed devuelve labels que es una matriz de etiquetas, una matriz NumPy con el mismo ancho y alto que nuestra imagen de entrada. Cada valor de píxel como un valor de etiqueta único. Los píxeles que tienen el mismo valor de etiqueta pertenecen al mismo objeto.

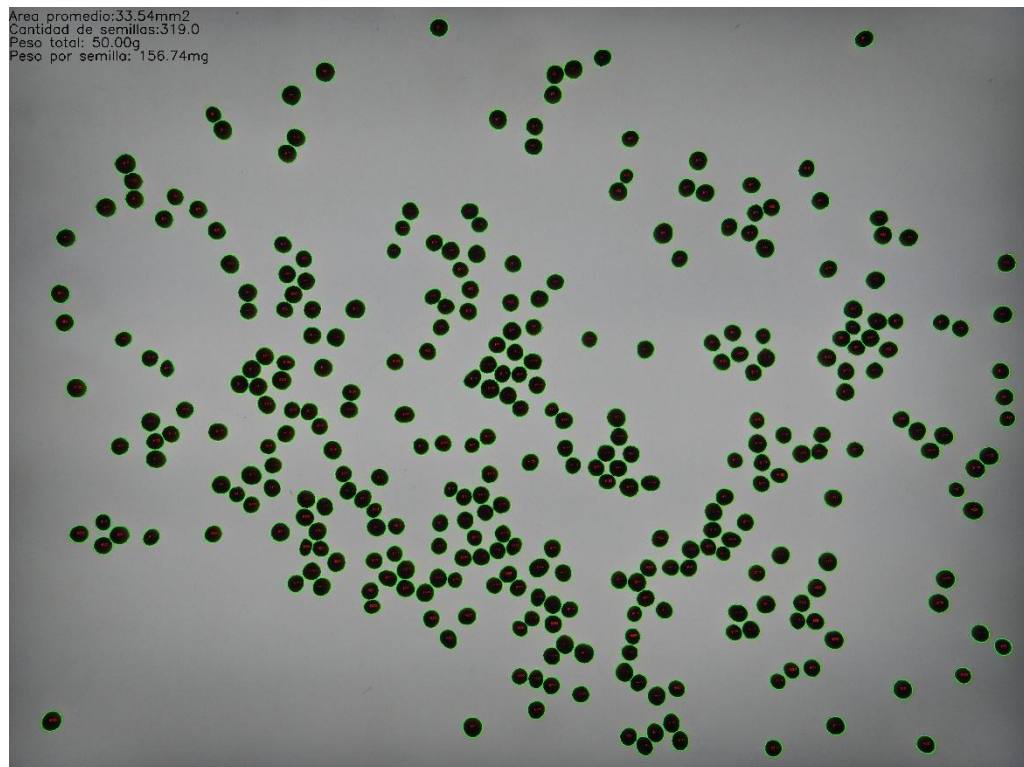
El último paso es simplemente recorrer en loop los valores de etiqueta únicos y extraer cada uno de los objetos:

```
1- # loop over the unique labels returned by the Watershed
2- # algorithm
3- for label in np.unique(labels):
4- # if the label is zero, we are examining the 'background'
5- # so simply ignore it
6- if label == 0:
7- continue
8- # otherwise, allocate memory for the label region and draw
9- # it on the mask
10-mask = np.zeros(gray.shape, dtype="uint8")
11-mask[labels == label] = 255
12-# detect contours in the mask and grab the largest one
13-cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
14-cv2.CHAIN_APPROX_SIMPLE)
15-cnts = imutils.grab_contours(cnts)
16-c = max(cnts, key=cv2.contourArea)
17-# draw a circle enclosing the object
18-((x, y), r) = cv2.minEnclosingCircle(c)
19-cv2.circle(image, (int(x), int(y)), int(r), (0, 255, 0), 2)
20-cv2.putText(image, "{}".format(label), (int(x) - 10, int(y)),
21-cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
22-# show the output image
23-cv2.imshow("Output", image)
24-cv2.waitKey(0)
```

En la línea 3 comenzamos a recorrer cada una de las etiquetas únicas. Si la etiqueta es cero, entonces estamos examinando el "componente de fondo", por lo que simplemente lo ignoramos. De lo contrario, en las líneas 10 y 11 se asigna memoria para nuestra máscara y se establecen los píxeles que pertenecen a la etiqueta actual en 255 (blanco).

En las líneas 13 a 16 detectamos los contornos en la máscara y extraemos el más grande; este contorno representará el límite de un objeto dado en la imagen. Luego, dado el contorno del objeto, todo lo que necesitamos hacer es dibujar el límite del círculo que rodea el objeto en las líneas 18 a 21.

Finalmente, las líneas 23 y 24 muestran la imagen de salida en pantalla:



Como se puede ver, hemos detectado con éxito las semillas en la imagen. Además, también hemos podido dibujar (contornos) limpiamente los límites que rodean cada una.

Determinación de área promedio de las semillas

Las áreas de los contornos detectados son guardadas en un vector:

```
for cnt in contours:  
    areas.append(cv2.contourArea(cnt))
```

Se calcula el área total de todos los segmentos, realizando la conversión de pixeles a mm² con la constante 0.0065036400258046845 que proviene de la resolución de la cámara y el tamaño de la bandeja que entra dentro del campo visual enmarcado, en otras palabras, de la relación pixeles/mm.

```
for i in range(0, semillas):  
    areat=areat+areas[i]  
    areat=(areat*0.0065036400258046845)
```

Siendo “*cant*” la cantidad de semillas detectadas se obtiene el área promedio de las semillas:

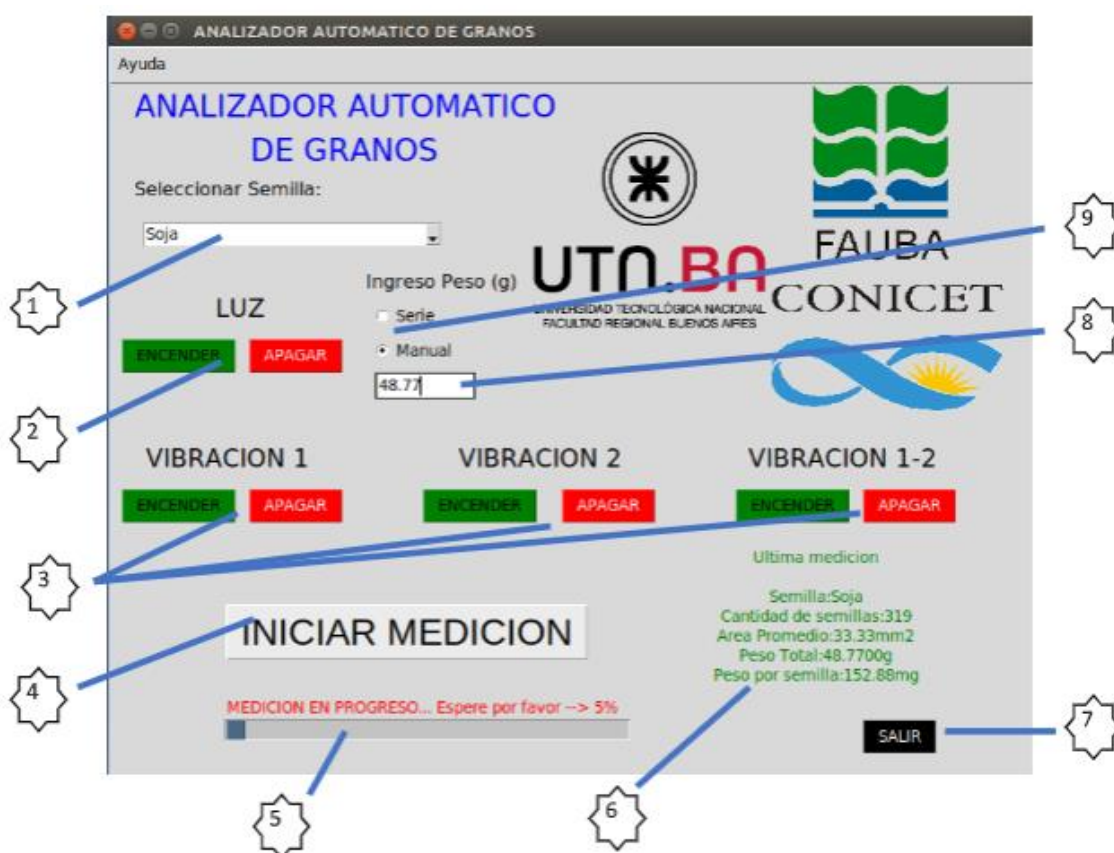
```

if cant!=0:
  Area_promedio=(areat/cant)
  
```

Interfaz gráfica

La GUI se diseñó utilizando lenguaje Python y la librería gráfica *Tkinter*.

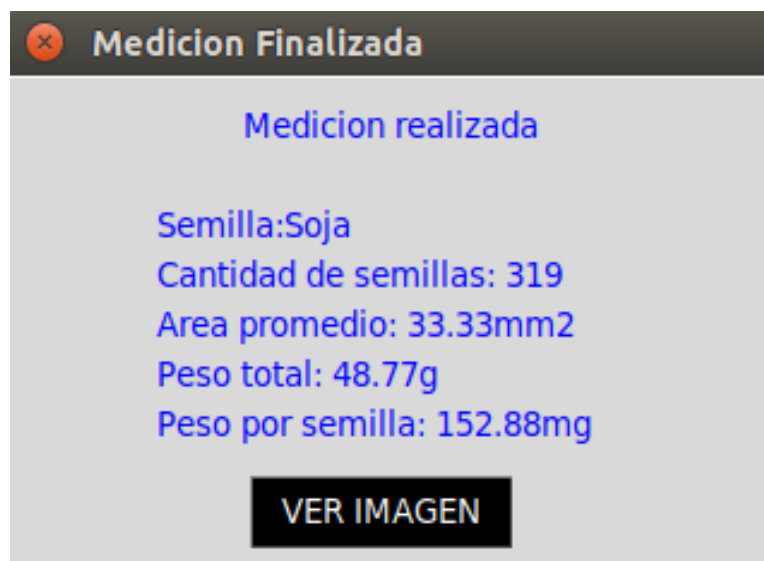
A continuación, se ilustra una imagen del aspecto de esta y sus principales componentes:



Referencias de la imagen anterior:

1. ***Menú desplegable de selección de tipo de semilla***
2. ***Encendido o apagado manual de iluminación para verificación***
3. ***Encendido o apagado de motores de vibración***
4. ***Botón de inicio de medición***
5. ***Barra de progreso de medición***
6. ***Información de última medición realizada***
7. ***Botón de salir de la aplicación***
8. ***Cuadro de ingreso de valor de peso manual***
9. ***Selección de ingreso de peso (Manual o por Puerto serie)***

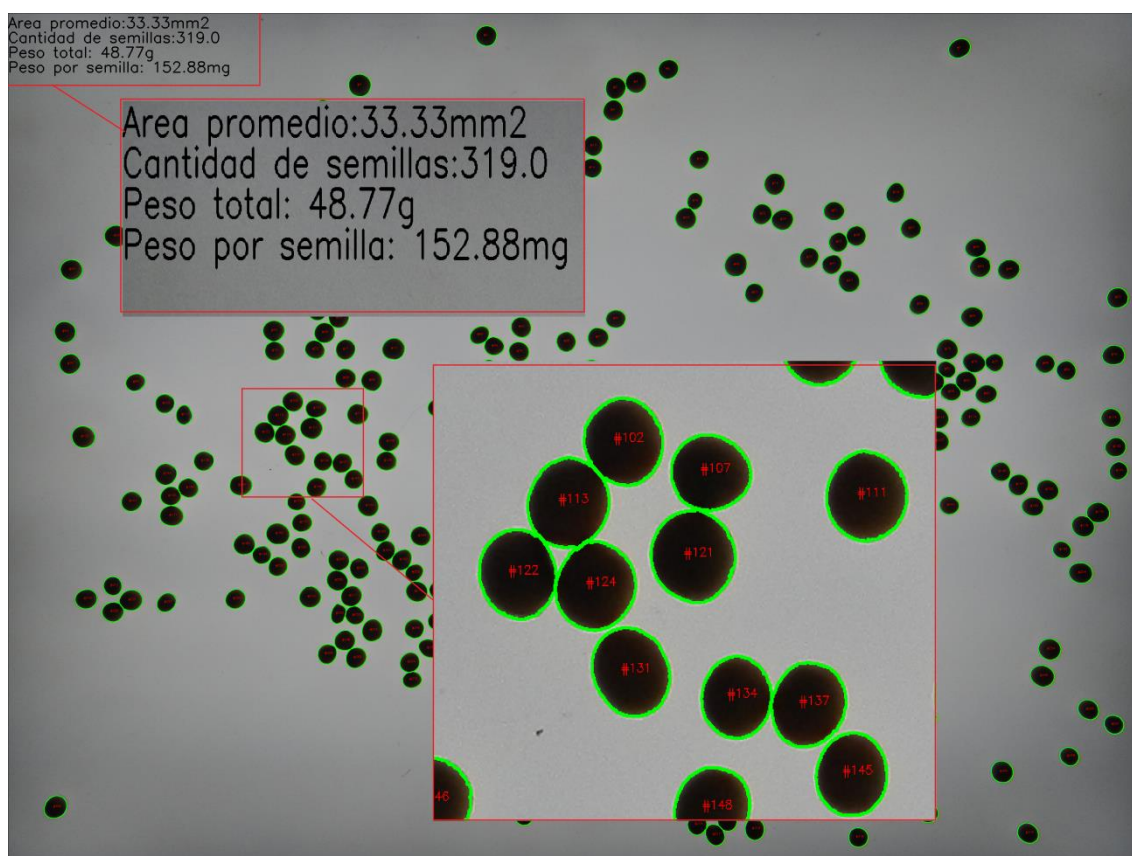
Al finalizar una medición se muestra una ventana con los resultados de esta y la opción de ver la imagen analizada presionando un botón. A continuación, se ilustra un ejemplo para una medición de semillas de soja:



Como se observa en la imagen el sistema informa todos los datos, además enumera y marca el contorno en color verde de cada semilla.

Hay casos en que el sistema encierra 2 o 3 semillas en un solo contorno, estos casos el mismo software lo resuelve en base al área media del total de los contornos y estima la cantidad de semillas encerradas en dicho contorno, marcándolo de otro color (verde cuando hay 1 semilla, amarillo cuando son 2 y violeta cuando son 3). Se debe tener en cuenta que, aunque en la imagen solo se enumera el contorno completo en la cuenta final de semillas el sistema tiene en cuenta que son más de una e informa el total.

Una vez verificado esta imagen puede ser guardada de forma local en el PC o en algún dispositivo de almacenamiento conectado al puerto USB.



Base luminosa con vibración

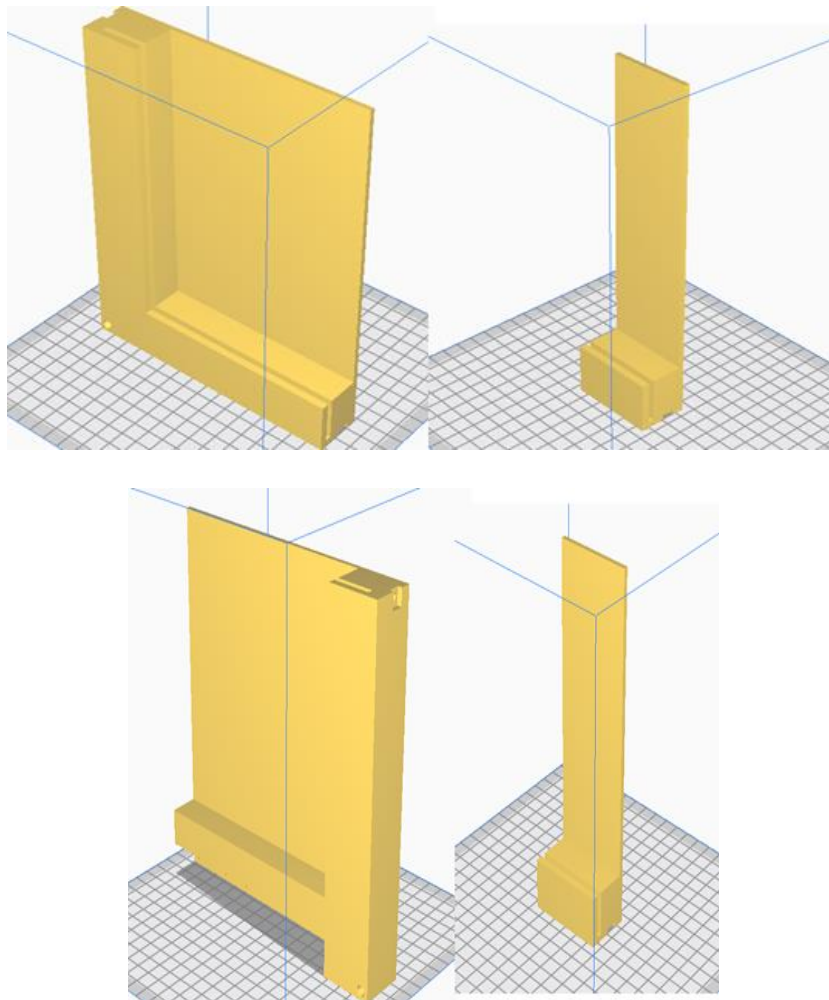
Se construyó una base luminosa que se encarga de proporcionar iluminación backlight a las semillas para que la cámara pueda detectar el contorno de estas correctamente.

Para la iluminación se ha utilizado una tira de LED 5050 que se alimenta con 12 V y para la difusión de la luz se utilizó una placa opal símil acrílico color blanco de 45cm x 35cm con 3mm de espesor.

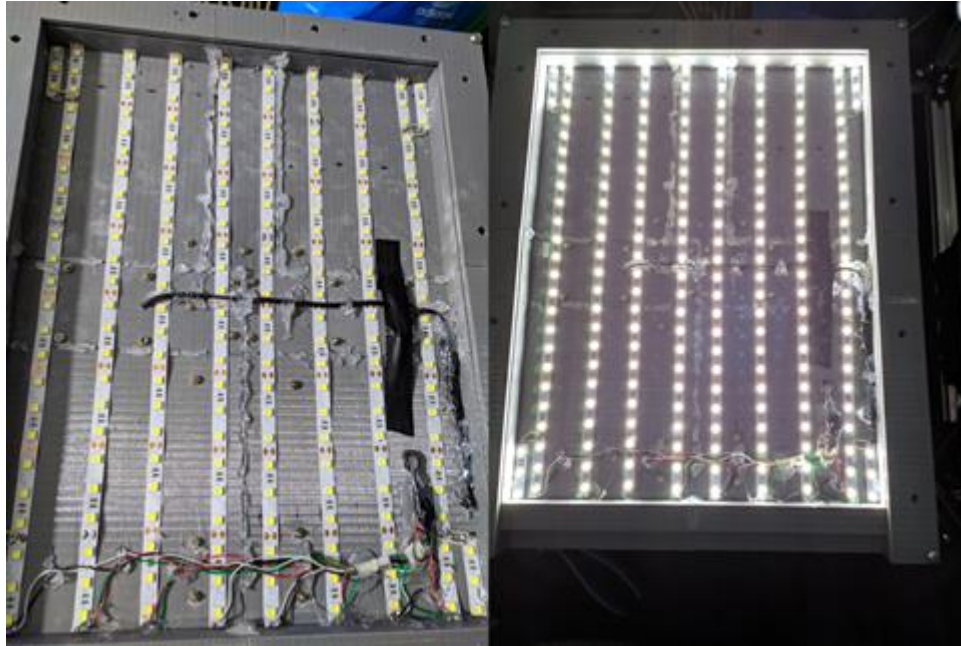
También se le han agregado 2 motores vibradores que se encargaran de mover las semillas para separarlas y obtener una mejor medición.

Se ha diseñado esta base para poder realizarse mediante impresión 3D.

Diseños de piezas impresas en 3D

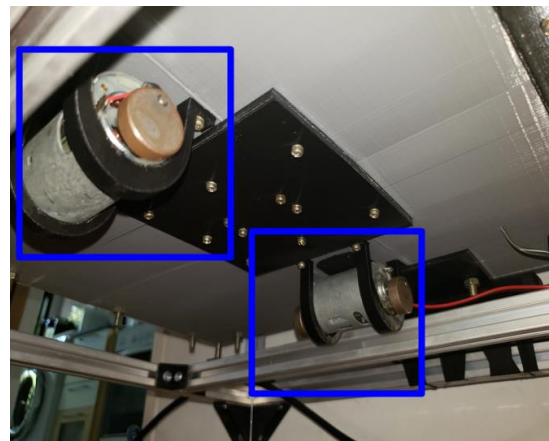


Base armada con tiras led colocadas



Motores vibradores alimentados con 24v

Los motores se encuentran sujetos en la parte posterior (abajo) de la bandeja utilizada como backlight.

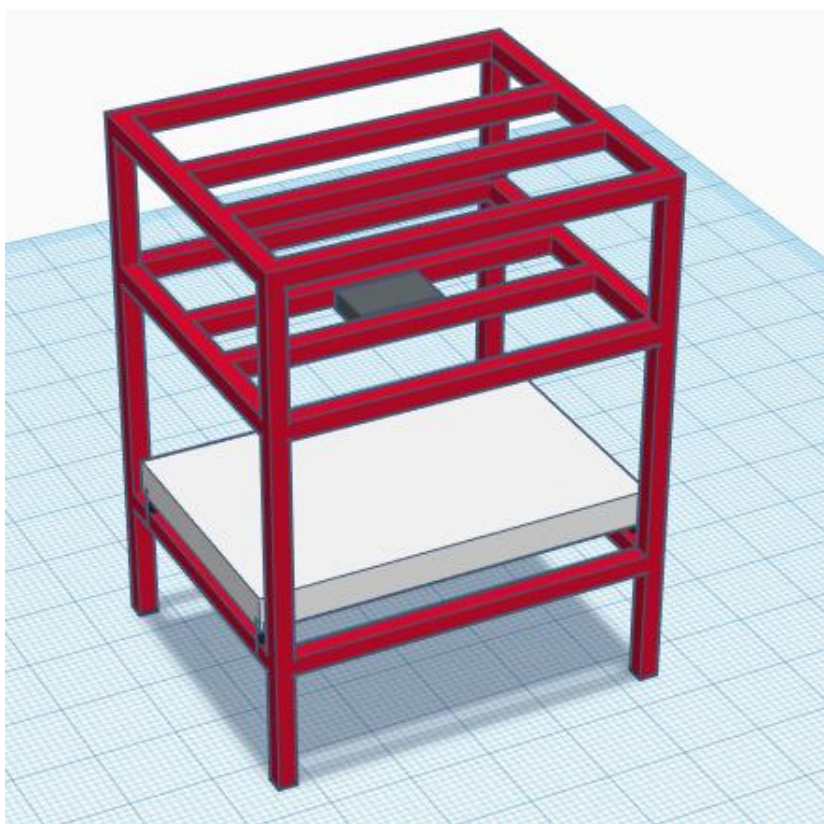


Estructura

La estructura ha sido diseñada para hacer el soporte a la base luminosa, así como también contener todas las partes del equipo como ser cámara, fuente, pc y electrónica.

Para la facilidad en el armado y que la misma no sea muy pesada se ha fabricado con perfiles de aluminio 2020.

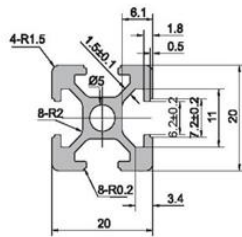
Diseño final



Nota: En rojo los perfiles de aluminio, en negro la cámara y en blanco la base luminosa.

Se utilizó para el armado 4 perfiles de 72 cm, 6 de 46 cm y 12 de 43 cm. La unión de los perfiles se realizó con 24 escuadras metálicas y se completó con piezas plásticas impresas en 3d, para estos últimos se utilizaron tornillos y tuercas M5.

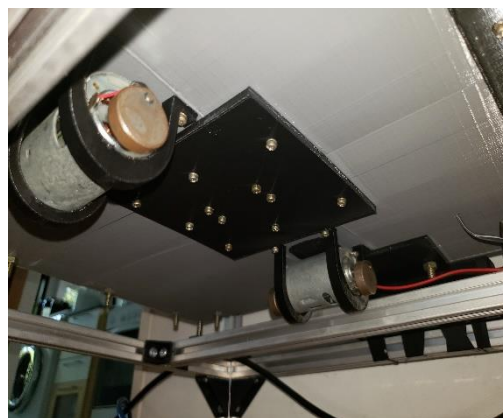
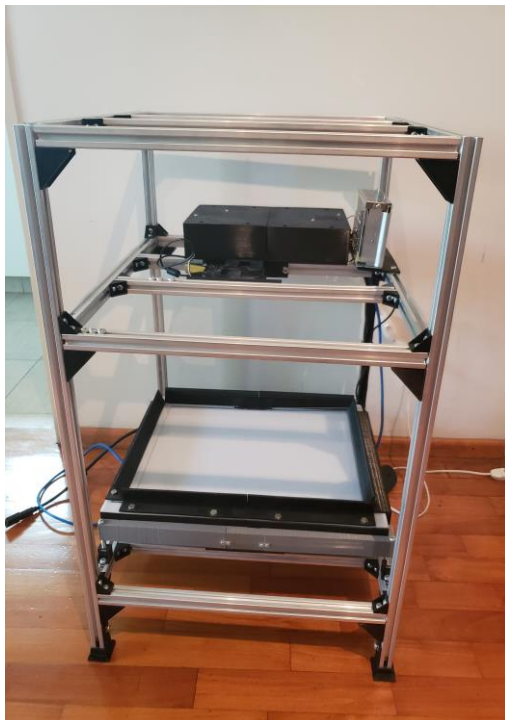
Perfil 2020



Escuadras utilizadas para el armado (unión de los perfiles)

Se agregó también un marco acrílico con bisagras que permite separar las semillas de los bordes de la base al momento de distribuir las mismas en la bandeja, y luego este marco se puede levantar quedando fuera del campo visual de la cámara, evitando así problemas en la detección de las semillas.

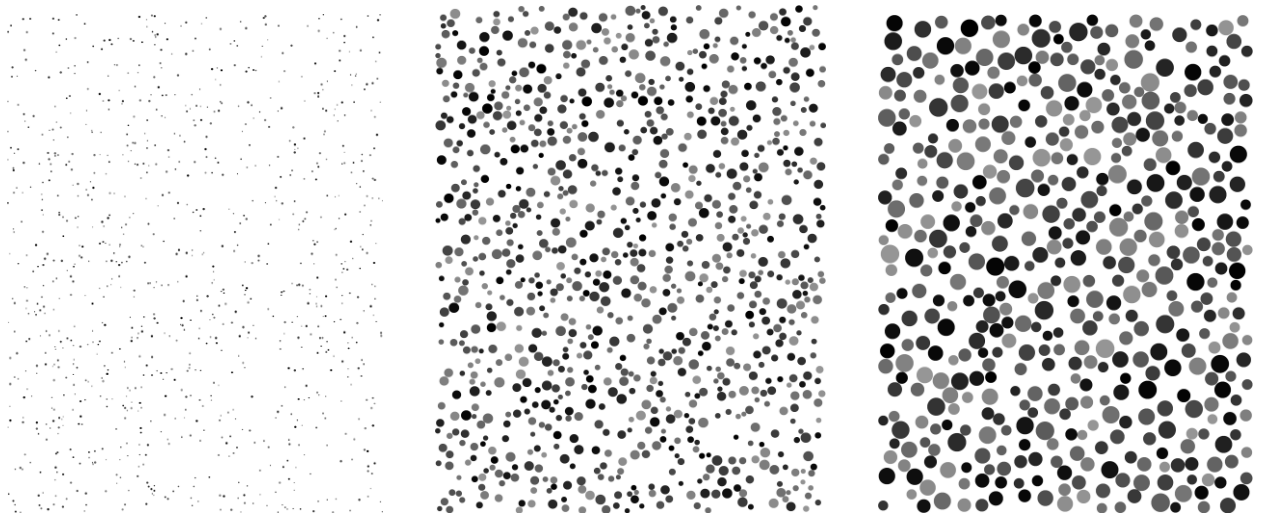
A continuación, se muestran imágenes de la estructura terminada, donde se puede apreciar también dicho marco, y la incorporación de los motores en la parte inferior de la base:



Simulaciones

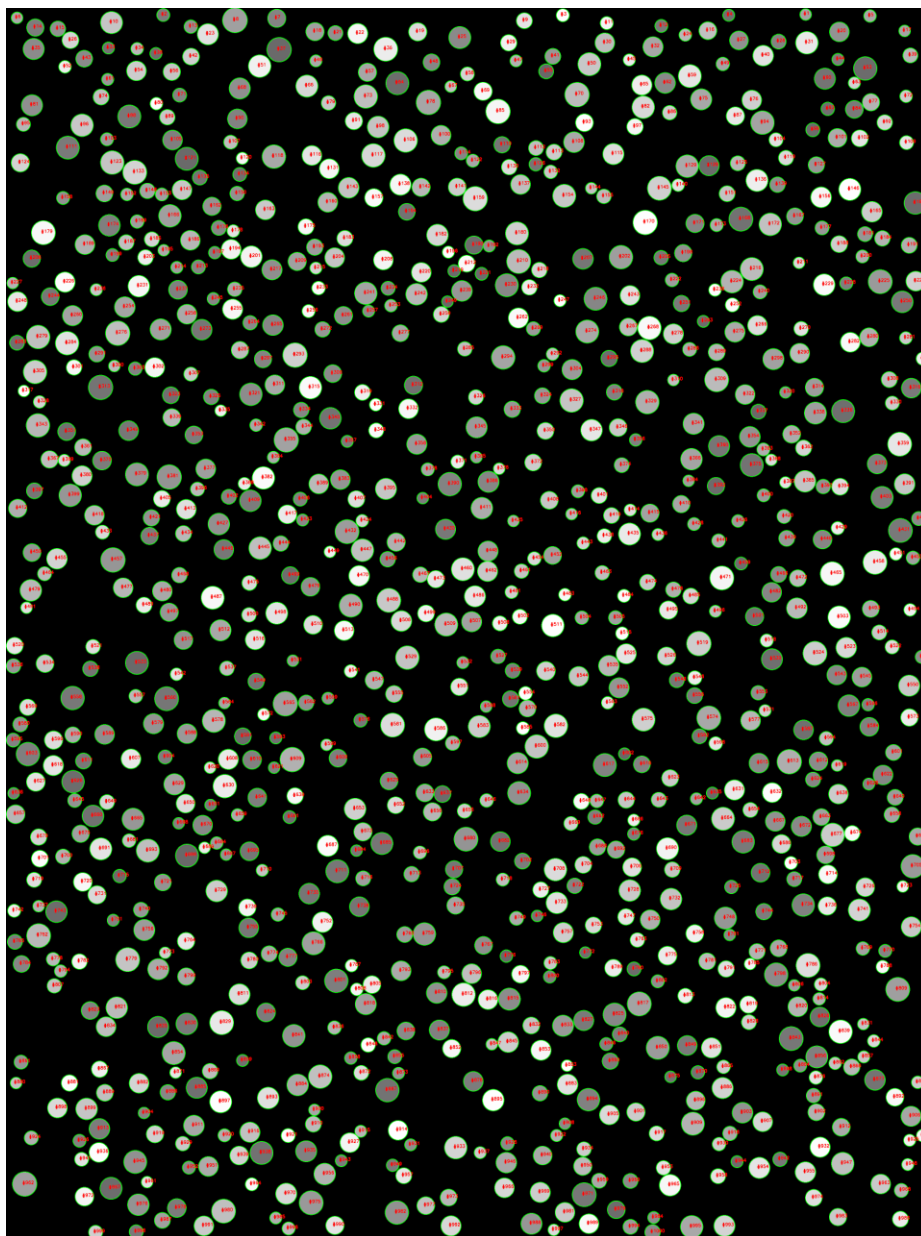
Para comenzar con las pruebas mientras se desarrollaba el equipo, se conformó un script para generar imágenes de prueba. Con esto fue posible crear imágenes con figuras tales como círculos, cuadrados y triángulos, de distintos tamaños, tonalidades y posición. Estos parámetros a su vez los hicimos variar en forma aleatoria, con lo cual por ejemplo se pudo crear varias imágenes con la misma cantidad de objetos, pero diferentes. Esto nos permitió tener una aproximación al caso real de las semillas y probar en primera instancia los resultados ante los algoritmos utilizados para el procesamiento de imágenes.

A continuación 3 ejemplos de estas imágenes de prueba con círculos:



Con este tipo de pruebas se logró mejorar la implementación de los algoritmos de procesamiento de imágenes y alcanzar resultados perfectos, es decir sin ningún tipo de error para estas imágenes generadas artificialmente.

Ejemplo de la detección de los círculos:



Mediciones

Se realizaron mediciones con varios tipos de semillas, en algunos casos los algoritmos que funcionaban bien en las simulaciones, no eran efectivos en la realidad con cierto tipo de semillas. El principal problema se debe al solapamiento de estas, a la forma y a la variación de tonalidad en las semillas de las imágenes obtenidas.

Finalmente se decidió en trabajar a cada tipo de semilla en forma distinta, con distintos valores en los parámetros usados para el procesamiento de imágenes y se fueron ajustando los mismos según cada caso.

Para todos los casos se optó por utilizar el algoritmo watershed como uno de los pasos fundamentales en el procesamiento de imágenes, ya que fue el que brinda mejores resultados para segmentar las semillas en el caso de solapamiento. La contra de este algoritmo es que puede generar sobre segmentación y por ello resulta de gran importancia la forma en que se aplica dicho algoritmo y el valor de los parámetros en juego.

Se siguieron las pautas descriptas en el documento “Verificación de alcance” donde se detallaron las pruebas a realizar para la verificación.

Si bien se realizaron muchas mediciones durante la puesta a punto del equipo, en este informe se ilustran las que se constataron en la facultad junto a los profesores en la etapa de verificación de alcance, 6 semillas de 15 fueron elegidas por los profesores para la prueba de conteo:

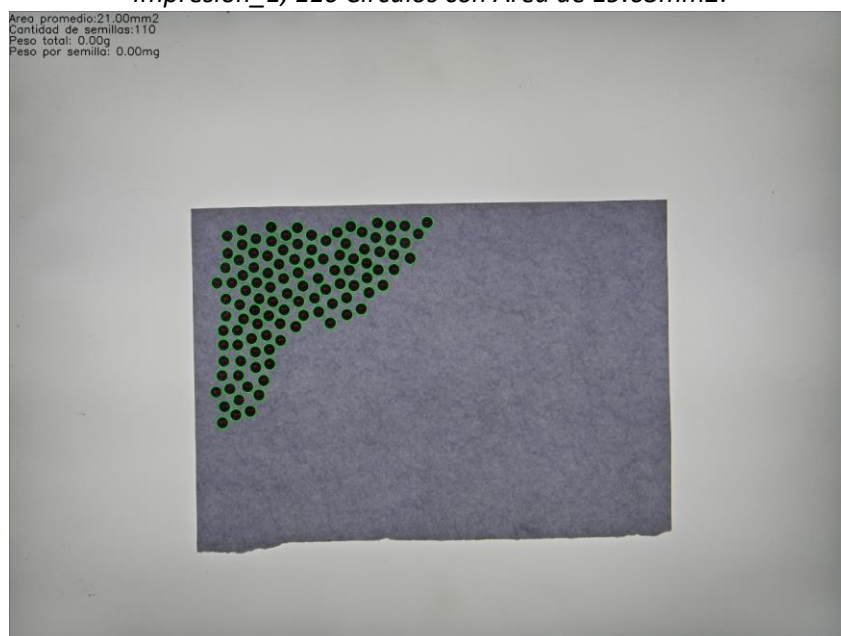
Semilla	Cantidad verdadera	Cantidad Medida	Error
Colza	1120	1119	0.09%
Vicia	680	680	0%
Maíz	220	221	0.45%
Trigo Sarraceno	740	740	0%
Cebada	870	872	0.23%
Quinoa	3960	3951	0.23%

Para el caso de la medición de área, diseñamos diferentes impresiones con imágenes con áreas conocidas como patrón con diferentes distribuciones en el campo visual de la cámara. En cada caso se constató que el valor medido estuviera dentro de lo especificado, es decir <10%, lo cual se cumplió ampliamente con la especificación.

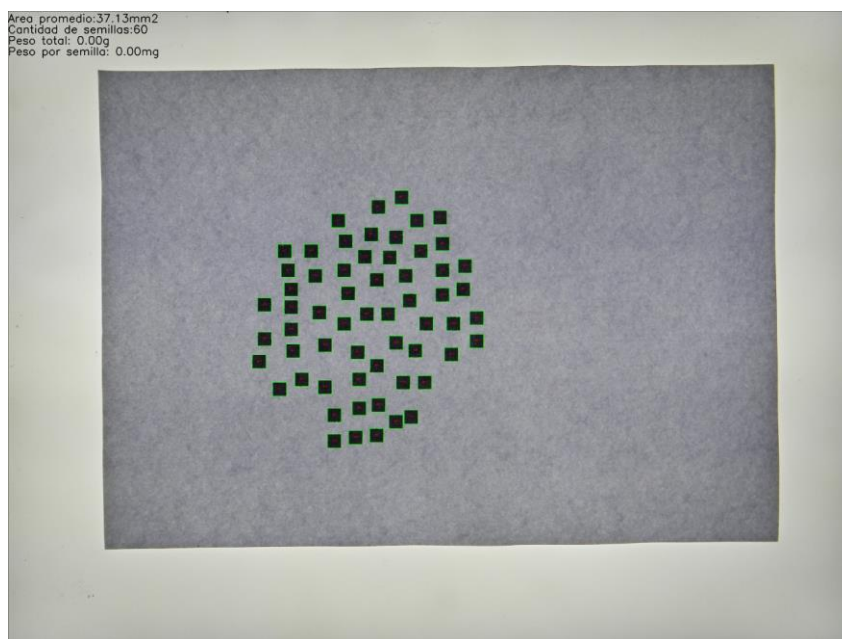
A continuación, se muestran los resultados para cuatro de las distintas impresiones usadas para la verificación con contornos patrones:

Impresión	Área verdadera	Área medida	Error
Impresión_1	19.63mm ²	21mm ²	6.979%
Impresión_2	36mm ²	37.13mm ²	3.139%
Impresión_3	19.63mm ²	20.85mm ²	6.215%
Impresión_4	36mm ²	37.54mm ²	4.278%

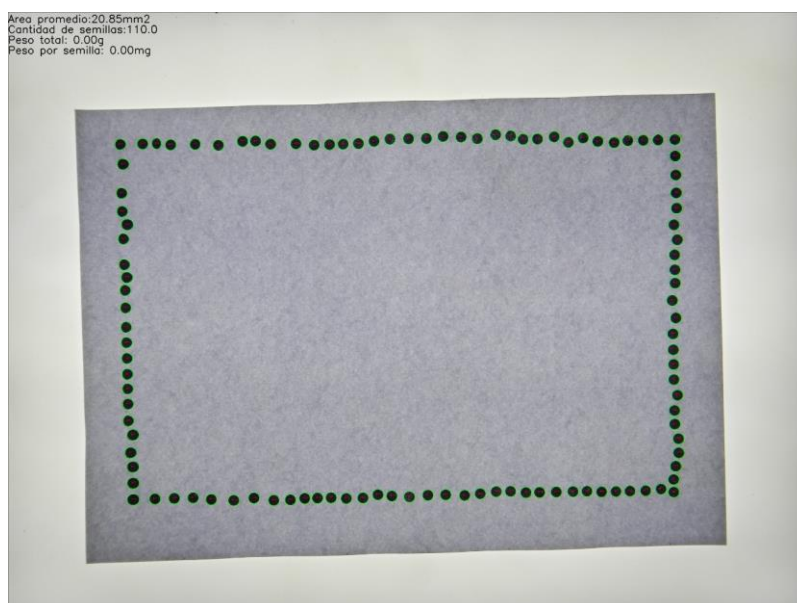
Impresión_1, 110 Círculos con Área de 19.63mm²:



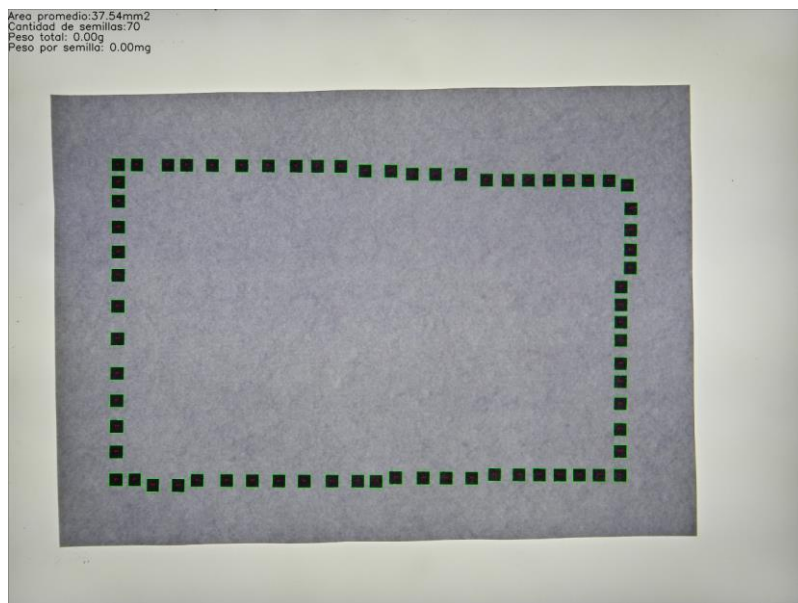
Impresión_2, 60 Cuadrados con área de 36mm²:



Impresión_3, 110 Círculos con Área de 19.63mm²:



Impresión_4, 70 Cuadrados con área de 36mm²



Bibliografía

- [1]Neilsen, Mitchell & Gangadhara, Shravan & Amaravadi, Siddharth. (2017). Extending Watershed Segmentation Algorithms for High-throughput Phenotyping on Mobile Device
- [2]Ping, Zhao & Yongkui, Li. (2009). Grain Counting Method Based On Image Processing. 10.1109/ICIECS.2009.5364719.
- [3]Beucher, Serge & Marcotegui, B.. (2014). P algorithm, a dramatic enhancement of the waterfall transformation.
- [4]Hanbury, Allan & Marcotegui, B.. (2006). Waterfall Segmentation of Complex Scenes. 3851. 888-897. 10.1007/11612032_89.
- [5]Marçal, André. (2008). Alternative Methods for Counting Overlapping Grains in Digital Images. 5112. 1051-1060. 10.1007/978-3-540-69812-8_105.
- [6]M. Faessel and F. Courtois, TOUCHING GRAIN KERNELS SEPARATION BY GAP-FILLING, Image Analysis & Stereology, vol.28, issue.3, pp.195-203, 2011.
- [7]Kornilov, A.S.; Safonov, I.V. An Overview of Watershed Algorithm Implementations in Open Source Libraries. *J. Imaging* 2018, 4, 123.
- [8]Wu, W.; Zhou, L.; Chen, J.; Qiu, Z.; He, Y. GainTKW: A Measurement System of Thousand Kernel Weight Based on the Android Platform. *Agronomy* 2018, 8, 178.
- [9]Barbedo, Jayme. (2012). A Review on Methods for Automatic Counting of Objects in Digital Images. Latin America Transactions, IEEE (Revista IEEE America Latina). 10. 2112-2124. 10.1109/TLA.2012.6362356.