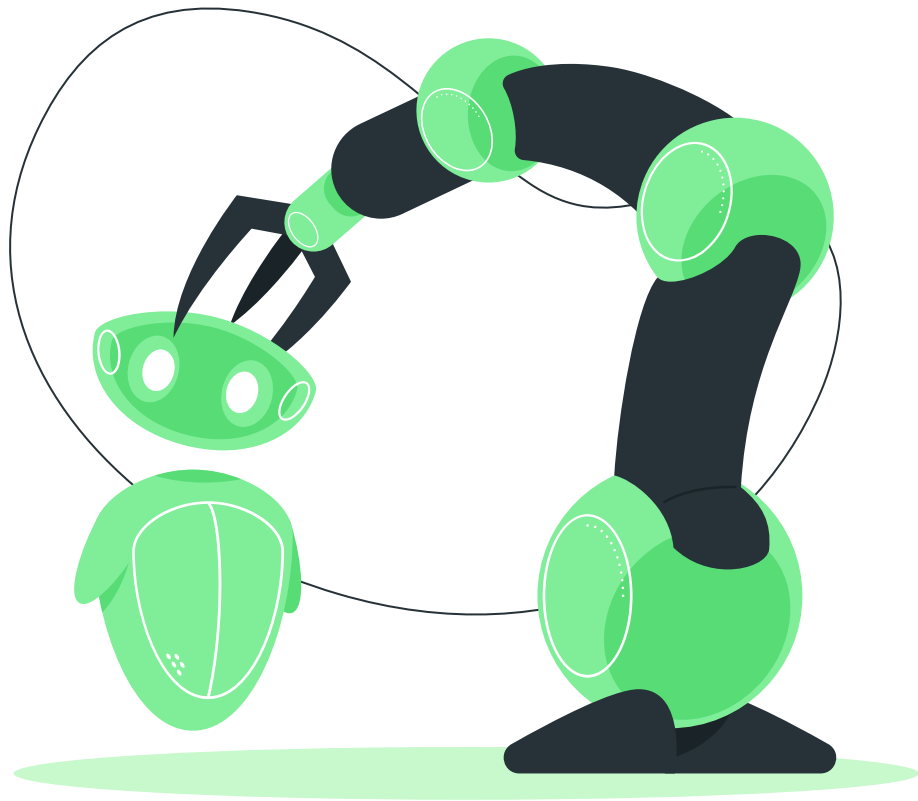


Automación Industrial

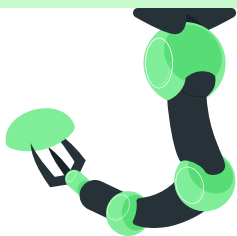
Trabajo Práctico Final

Tomás Guillermo Londero Bonaparte
Guido Enrique Lambertucci
Alan Mechoulam
Carlos Javier Maselli

58150
58009
58438
59564



Contenido



1

Modelización

Modelado del autómatas utilizando parámetros DH

3

Visión

Interpretación y procesamiento de imagen

2

Trayectoria

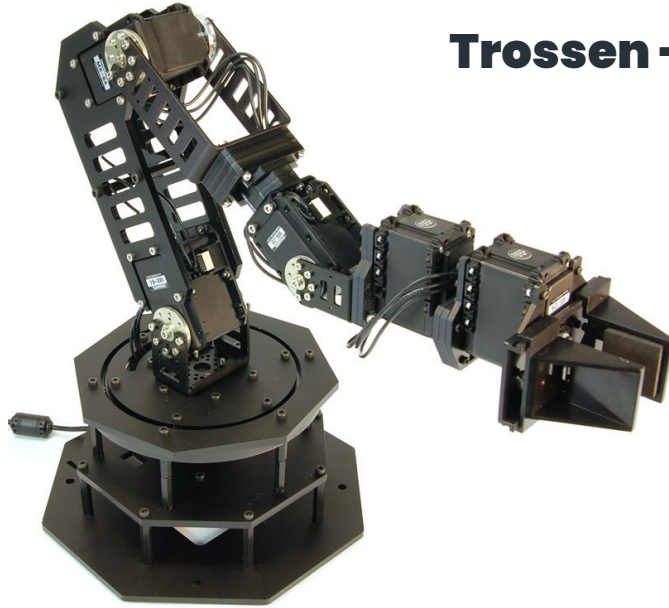
Generación de trayectoria dado inicio y final

4

Integración

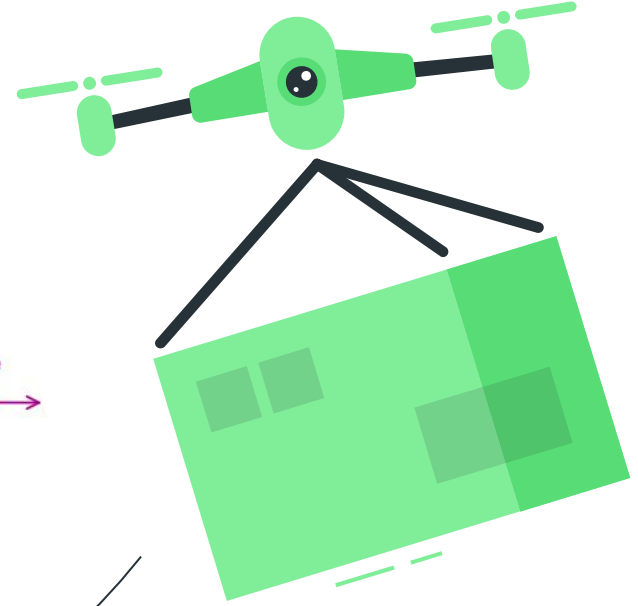
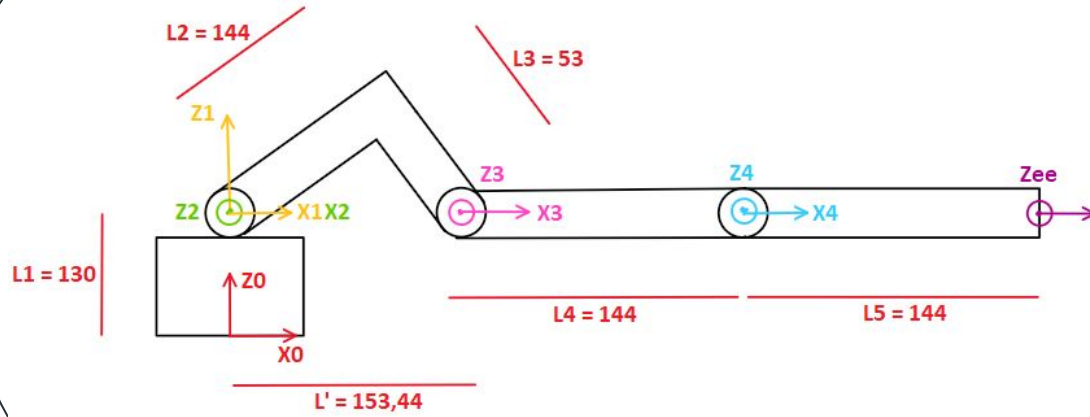
Simulación del movimiento del autómatas dada una imagen de entrada

Modelización I



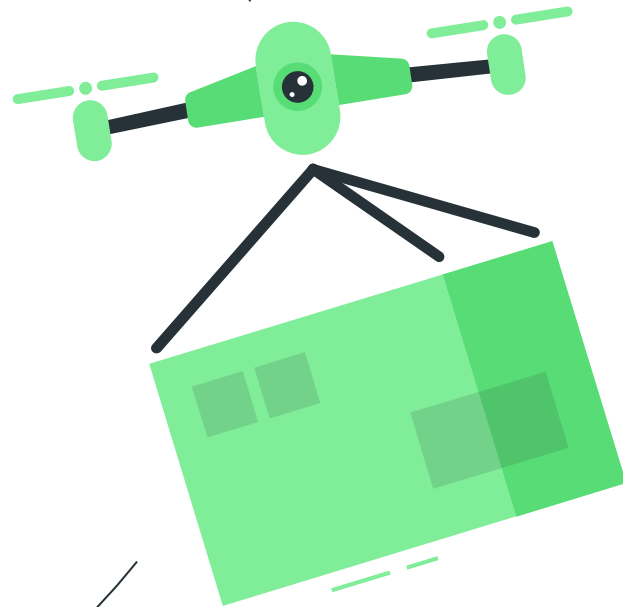
Trossen – WidowX MK-II

Modelización II



Modelización III

| i | a_{i-1} | α_{i-1} | d_i | θ_i |
|------|-----------|-----------------|-------|------------|
| 1 | 0 | 0 | L_1 | θ_1 |
| 2 | 0 | $\frac{\pi}{2}$ | 0 | θ_2 |
| 3 | L' | 0 | 0 | θ_3 |
| 4 | L_4 | 0 | 0 | θ_4 |
| EE | L_5 | 0 | 0 | 0 |



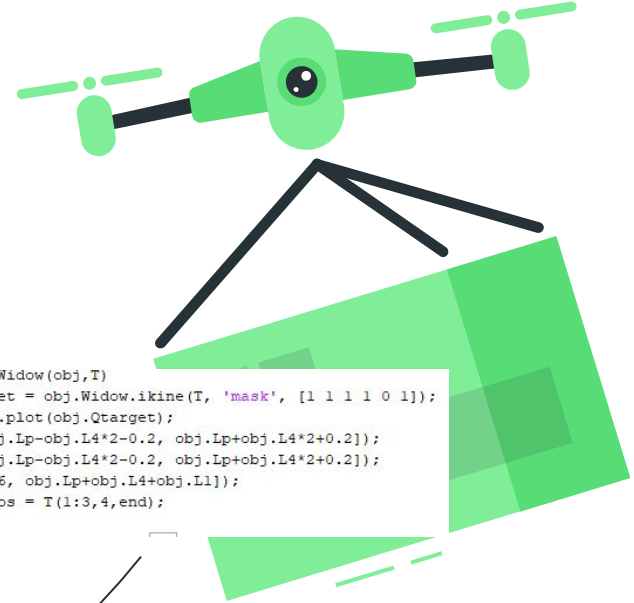
Modelización IV

Clase de Matlab del Manipulador

```
methods
%Metodos de la clase
function obj=WidowXMKII(L1,L2,L3,L4,L5,table_height,init_pos) ...
function moveWidow(obj,T) ...
function moveWidowXY(obj,pos) ...
function pos=getPosition(obj) ...
function valid=isReachable(obj,position) ...
function getWidowInPosition(obj,down) ...
function showReachableSpace(obj,abst1,abst2,abst3,abst4) ...
function P=createDownwardTrajectory(obj,xyCoords,step,downward) ...
end
methods(Static)
function T=createLineTrajectory(InitialPosition,FinalPosition,step) ...
end

properties
Widow;
currPos;
Qtarget;
Lp;
L1;
L2;
L3;
L4;
L5;
table_height;
rotation;
end
```

```
function moveWidow(obj,T)
obj.Qtarget = obj.Widow.ikine(T, 'mask', [1 1 1 1 0 1]);
obj.Widow.plot(obj.Qtarget);
xlim([-obj.Lp-obj.L4*2-0.2, obj.Lp+obj.L4*2+0.2]);
ylim([-obj.Lp-obj.L4*2-0.2, obj.Lp+obj.L4*2+0.2]);
zlim([-0.6, obj.Lp+obj.L4+obj.L1]);
obj.currPos = T(1:3,4,end);
end
```

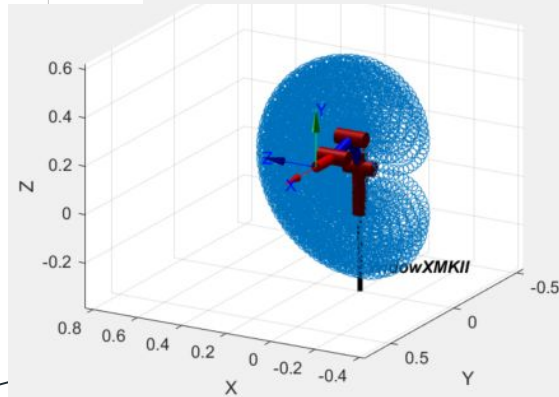
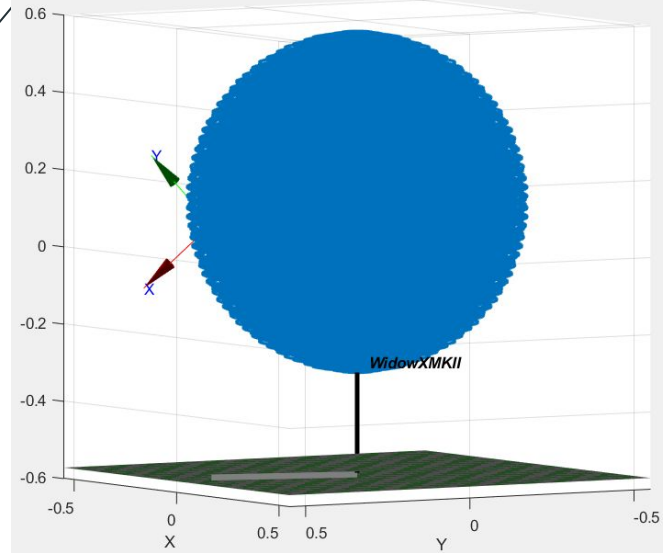


Modelización V

Espacio Alcanzable

%Se definen los largos y se aplican las formulas correspondientes para el
%calculo y dibujo del espacio alcanzable

```
L1=0.130;  
L2=0.144;  
L3=0.053;  
L4=0.144;  
L5=0.144;  
Lp=sqrt(L2^2+L3^2);  
x0=Lp+L4+L5-0.05;  
y0=0;  
z0=L1-0.05;  
t1=linspace(-180,180,180)*pi/180;  
t2=linspace(-90,90,10)*pi/180;  
t3=linspace(-90,90,10)*pi/180;  
t4=linspace(-180,180,40)*pi/180;  
[T1,T2,T3,T4]=ndgrid(t1,t2,t3,t4);  
xM = cos(T1).*(Lp*cos(T2)+L4*cos(T2+T3)+L5*cos(T2+T3+T4));  
yM = sin(T1).*(Lp*cos(T2)+L4*cos(T2+T3)+L5*cos(T2+T3+T4));  
zM = L1+Lp*sin(T2)+L4*sin(T2+T3)+L5*sin(T2+T3+T4);  
plot3(xM(:),yM(:),zM(:),'b')
```



Mostrar espacio alcanzable

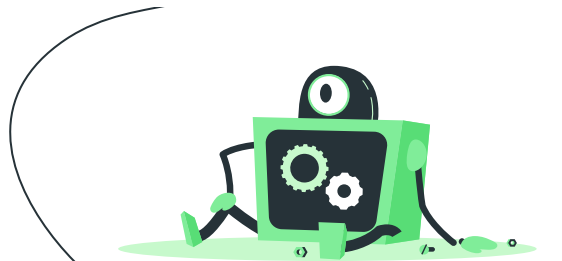
| | | |
|----|-----|----------------------|
| θ1 | 1 | <input type="text"/> |
| θ2 | 90 | <input type="text"/> |
| θ3 | 90 | <input type="text"/> |
| θ4 | 180 | <input type="text"/> |

Trayectoria I

Creando las trayectorias:

```
function P=createDownwardTrajectory(obj,xyCoords,step,downward)
    xi=xyCoords(1);
    yi=xyCoords(2);
    if(downward)
        z=linspace(obj.table_height + 0.05,obj.table_height,step);
    else
        z=linspace(obj.table_height,obj.table_height + 0.05,step);
    end
    x=z*0+xi;
    y=z*0+yi;
    P=[x;y;z];
end
```

```
function T=createLineTrajectory(InitialPosition,FinalPosition,step)
    xi=InitialPosition(1);
    yi=InitialPosition(2);
    xo=FinalPosition(1);
    yo=FinalPosition(2);
    if(xi ~= xo)
        b=(yo-yi)/(xo-xi);
        m=yo-xo*b;
        x=linspace(xi,xo,step);
        y=x*b+m;
    else
        y = linspace(yi,yo,step);
        x = xo+y*0;
    end
    T=[x;y];
end
```



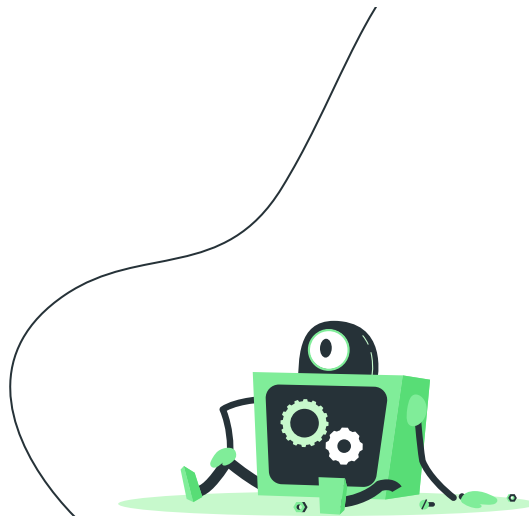
Trayectoria II

Moviendo al modelo:

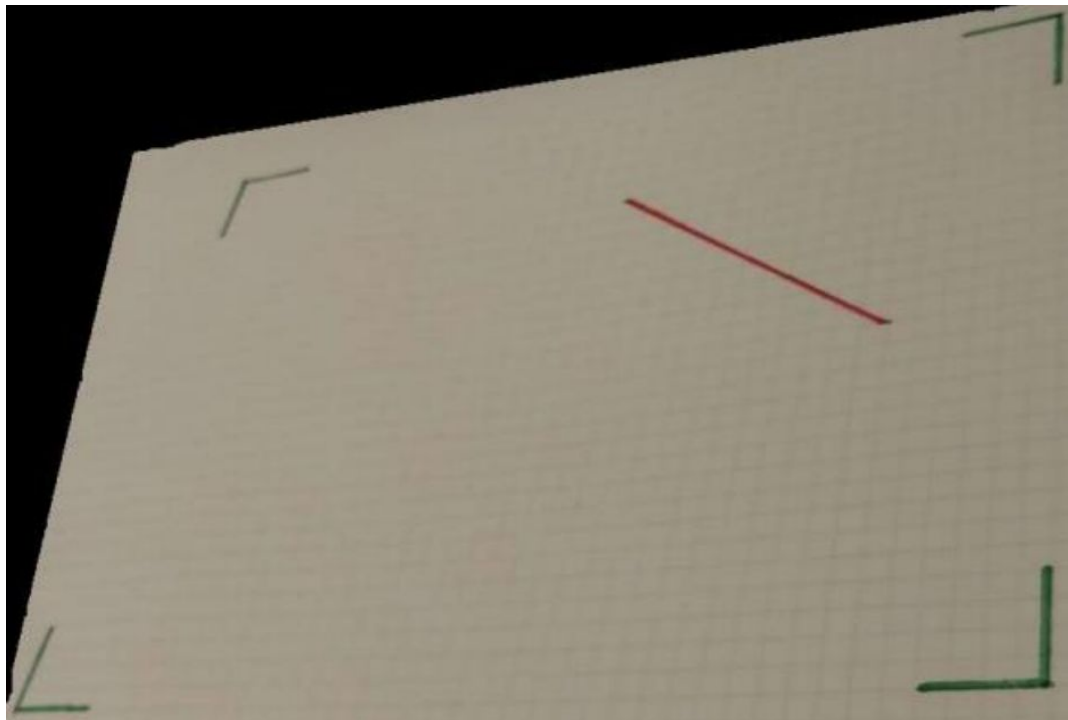
```
function moveWidow(obj,T)
    obj.Qtarget = obj.Widow.ikine(T, 'mask', [1 1 1 1 0 1]);
    obj.Widow.plot(obj.Qtarget);
    xlim([-obj.Lp-obj.L4*2-0.2, obj.Lp+obj.L4*2+0.2]);
    ylim([-obj.Lp-obj.L4*2-0.2, obj.Lp+obj.L4*2+0.2]);
    zlim([-0.6, obj.Lp+obj.L4+obj.L1]);
    obj.currPos = T(1:3,4,end);
end

function moveWidowXY(obj,pos)
    obj.currPos(1) = pos(1);
    obj.currPos(2) = pos(2);
    T = [obj.rotation, obj.currPos; 0, 0, 0, 1];
    obj.Qtarget = obj.Widow.ikine(T, 'mask', [1 1 1 1 0 1]);
    obj.Widow.plot(obj.Qtarget);
    xlim([-obj.Lp-obj.L4*2-0.2, obj.Lp+obj.L4*2+0.2]);
    ylim([-obj.Lp-obj.L4*2-0.2, obj.Lp+obj.L4*2+0.2]);
    zlim([-0.6, obj.Lp+obj.L4+obj.L1]);
    obj.currPos = T(1:3,4,end);
end
```

```
function getWidowInPosition(obj,down)|
    if (down)
        obj.currPos(3) = obj.table_height;
    else
        obj.currPos(3) = obj.table_height + 0.05;
    end
end
```



Visión I



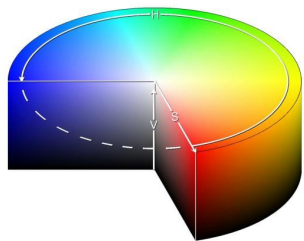
Visión II



Filtrado y limpiado

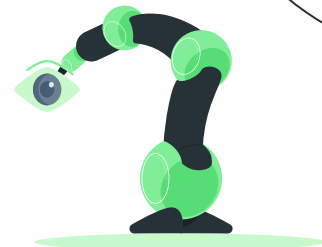
```
%0 grados hue = rojo
hsv_redhue_hi = (-7.5+27.5)/360; %27.5 grados adelante de -7.5 grados
hsv_redhue_lo = ((-7.5+360)-27.5)/360; %27.5 grados atras de -7.5 grados
%~120 grados hue = verde
hsv_greenhue_hi = (110+80)/360; %80 grados arriba de 110 grados
hsv_greenhue_lo = (110-75)/360; %75 grados abajo de 110 grados

hsv_sat_lo = 0.14;
hsv_val_hi = 0.55;
hsv_val_lo = 0.25;
```

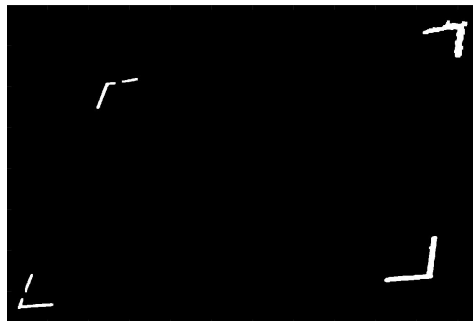


Espacio HSV

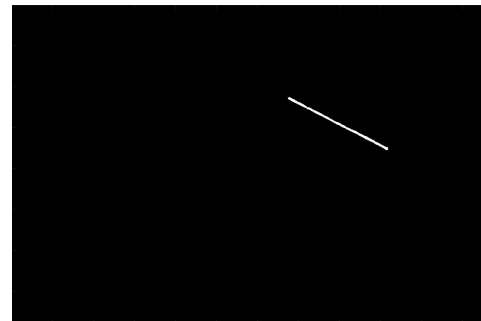
```
green_filter_c1 = iclose(green_filter, ones(5));
green_filter_c12 = iclose(green_filter, ones(7));
green_filter_1 = iopen(green_filter_c1, ones(3));
green_filter_12 = iopen(green_filter_c1, ones(3), 2);
green_filter_13 = iopen(green_filter_c1, ones(3), 3);
green_filter_14 = iopen(green_filter_c12, ones(11));
red_filter_1 = iclose(red_filter, ones(7));
red_filter_1 = iopen(red_filter_1, ones(3));
```



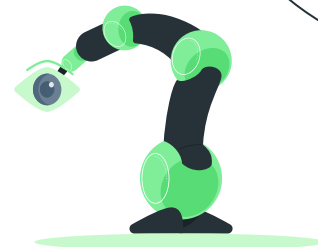
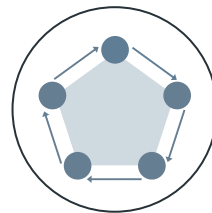
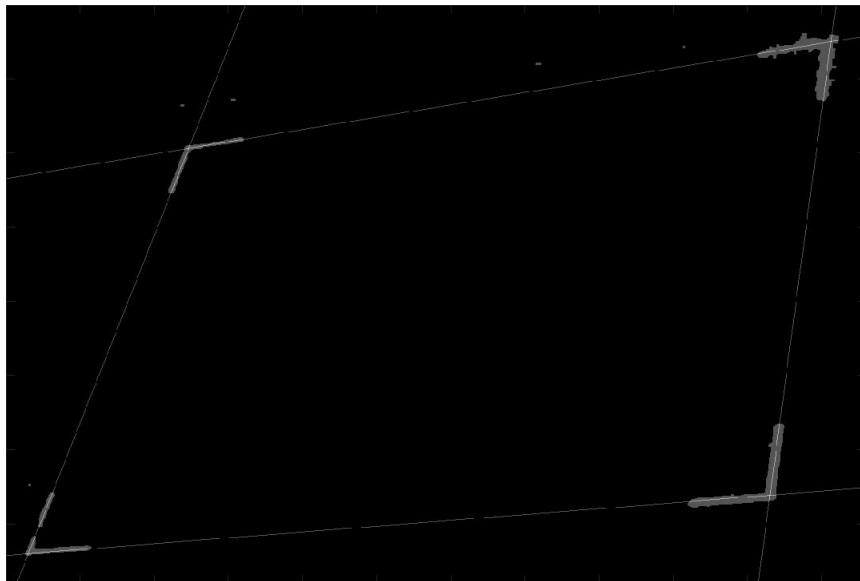
Verde



Rojo



Visión III



Se utiliza el algoritmo de Hough para encontrar las líneas generadas por las esquinas verdes.

Visión IV

Identificación de esquinas

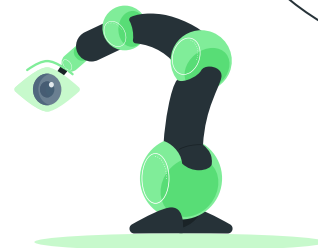
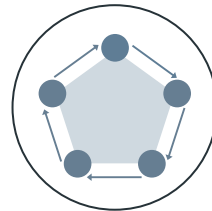
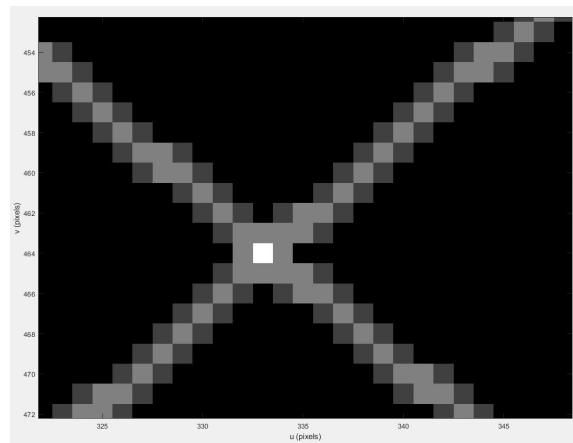
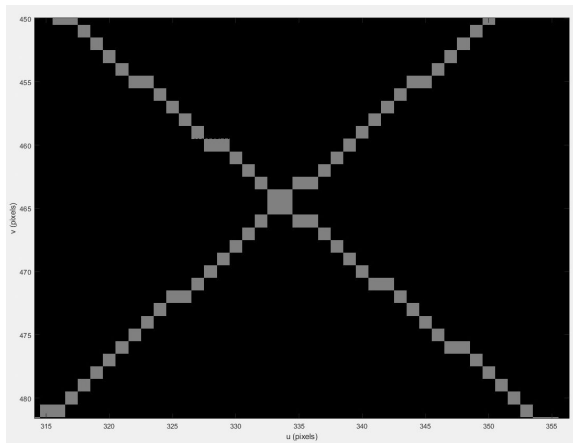
```
% Donde se superponen vale 2
```

```
sum_bordes = imlinea1+imlinea2+imlinea3+imlinea4;
```

```
% Hay casos en los que hay interseccion pero no hay superposicion
```

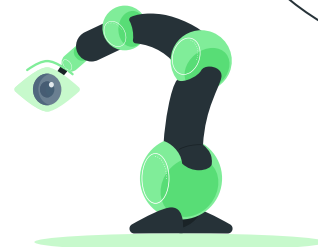
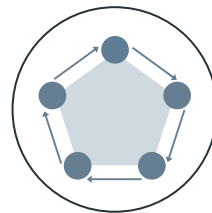
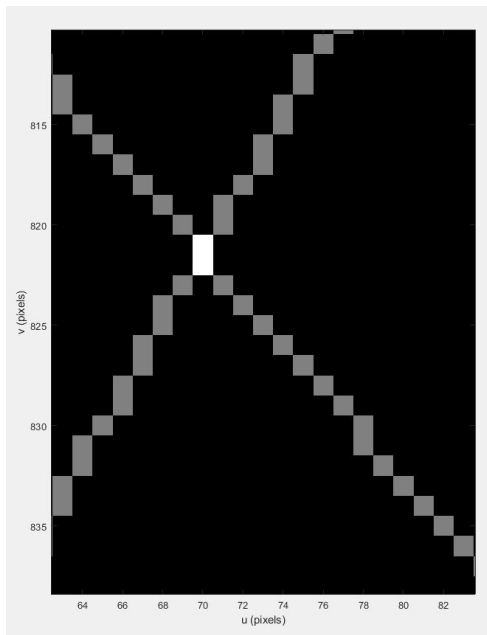
```
csum_bordes = conv2(sum_bordes,ones(2,2),'valid');
```

```
csum_bordes = csum_bordes/max(max(csum_bordes));
```



Visión V

Identificación de esquinas



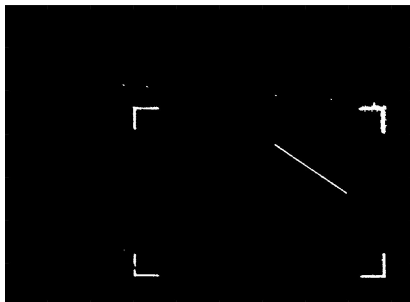
La superposición de líneas puede dar múltiples coordenadas para una misma esquina.
La convolución puede dar como resultado esquinas que ya se habían computado con la superposición.

Visión VI

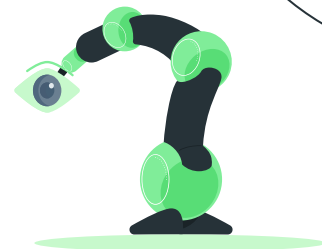
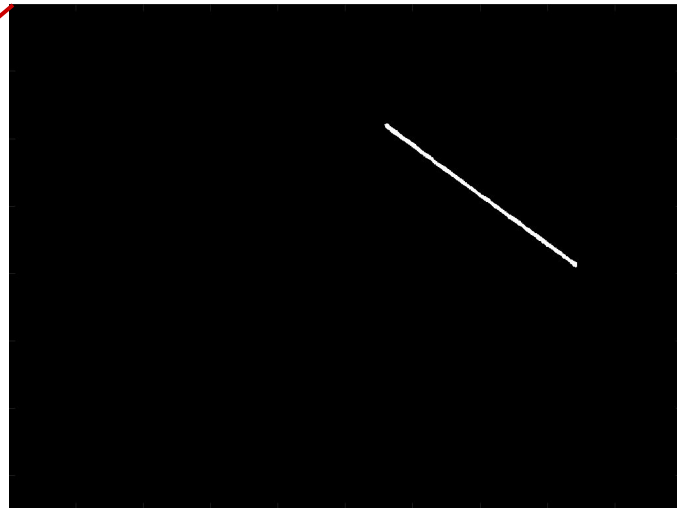


Ajustes de borde

Luego del warping se ajustan los bordes para quitar las esquinas verdes y conseguir un buen sistema de referencia para la recta.

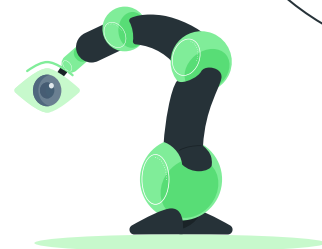
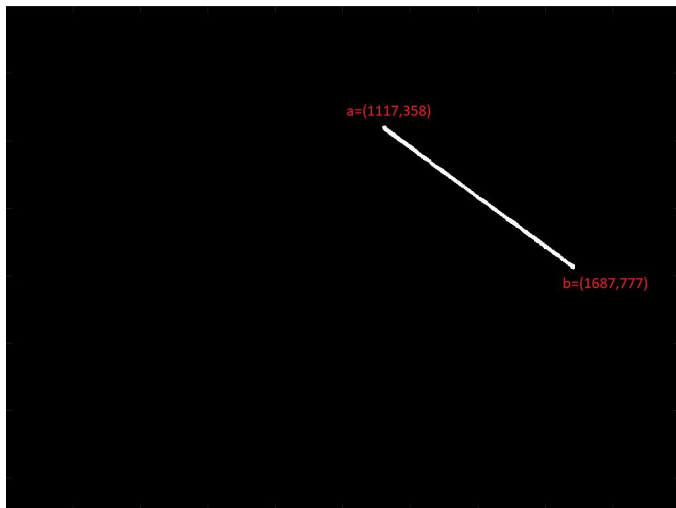


$$[x \ y] = [0 \ 0]$$



Visión VII

```
[row_fin,col_fin,~] = size(final_linea);  
fl_hough = Hough(final_linea,'suppress',30);  
imlinea5 = takeLine(fl_hough.lines.rho,fl_hough.lines.theta,col_fin,row_fin);  
fin_sup = imlinea5.*final_linea;  
  
[x_min, y_min] = find(fin_sup,1,'first');  
[x_max, y_max] = find(fin_sup,1,'last');
```



Para mayor inmunidad al ruido que pueda superar al filtrado, se toma 1 línea de Hough de la recta rotada. Luego se multiplica la recta por la imagen anterior mencionada. A este resultado es al cual se le buscan los extremos.



Extracción de Coordenadas

Integración I

Automacion_GUI

WidowXMKII

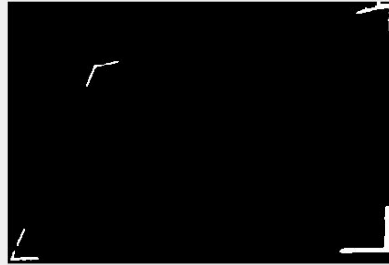
Cargar Imagen

Procesar Imagen

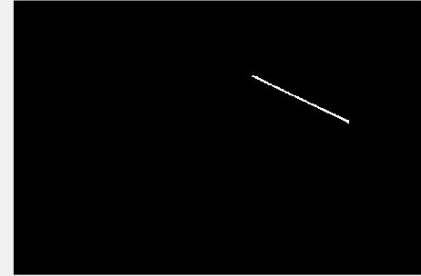
Mover manipulador

Modificar Filtros

Filtro verde procesado



Filtro rojo procesado



Modificar parámetros del Robot [mm]

Reset

L1 130

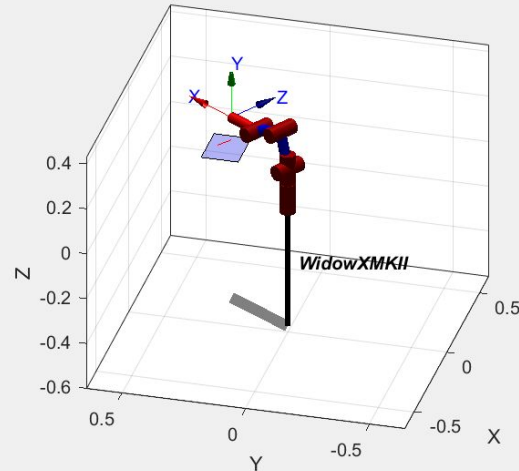
L2 145

L3 55

L4 144

L5 144

Confirmar



Mostrar espacio alcanzable

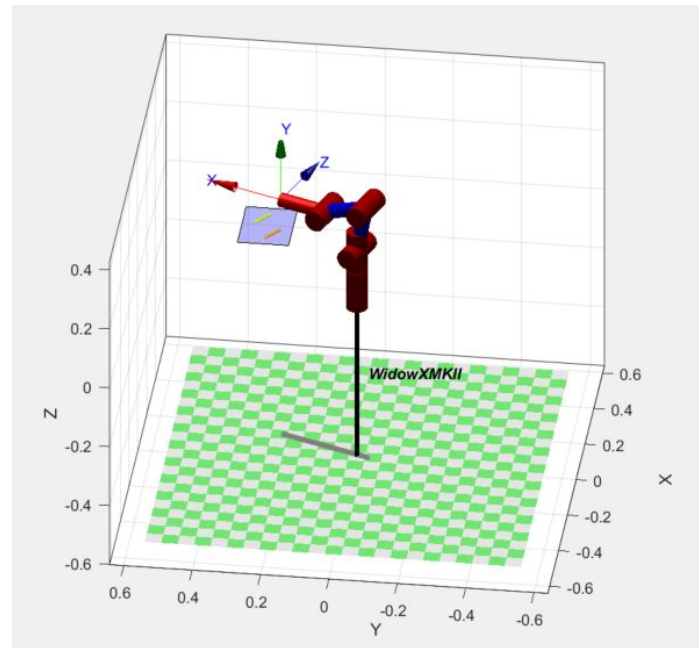
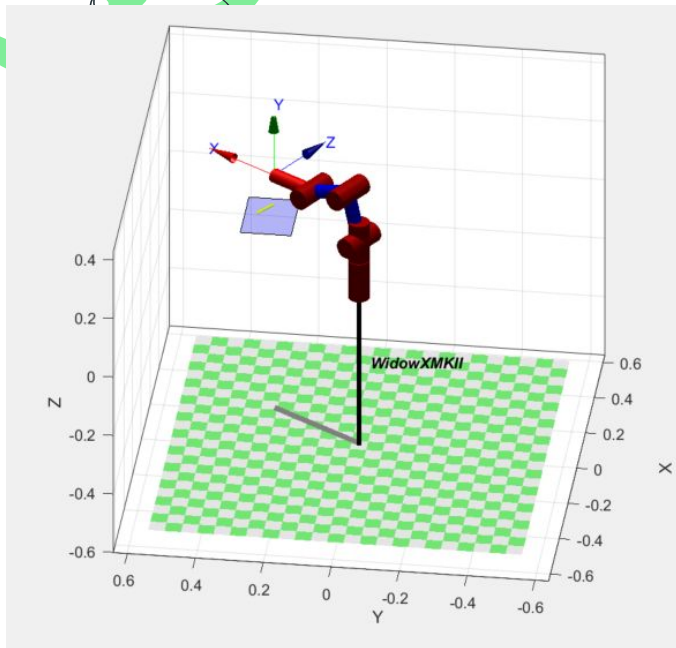
$|\theta 1|$ 130

$|\theta 2|$ 70

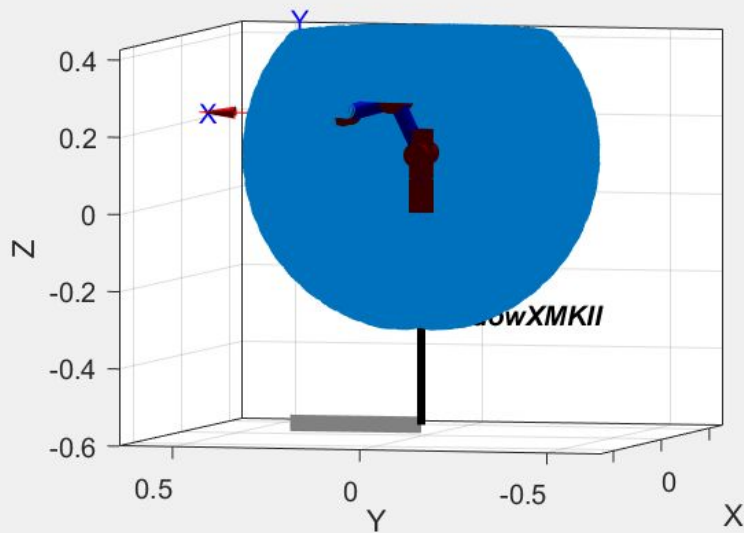
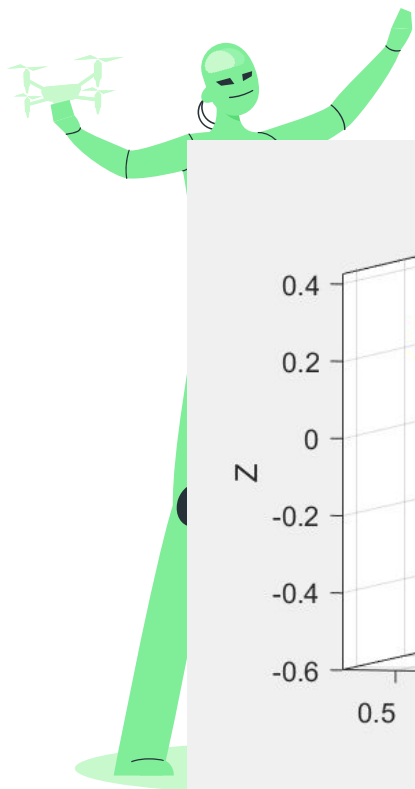
$|\theta 3|$ 70

$|\theta 4|$ 70

Integración II



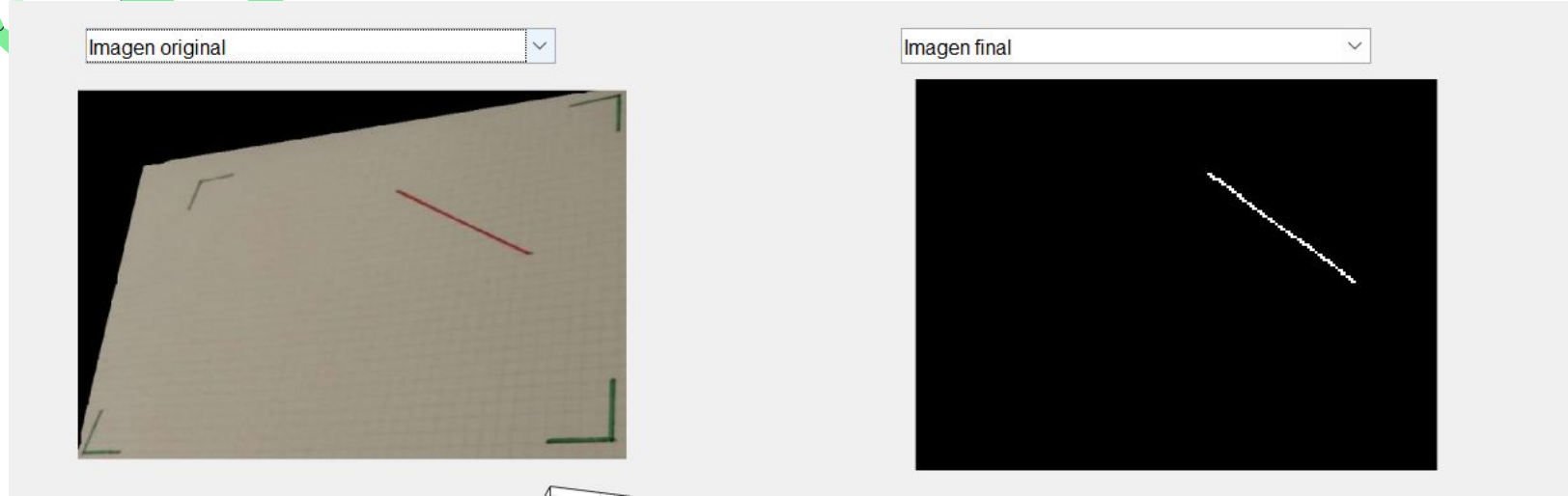
Integración III



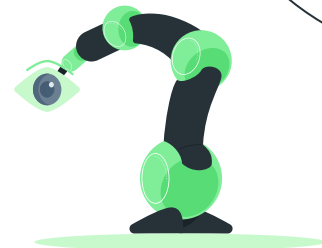
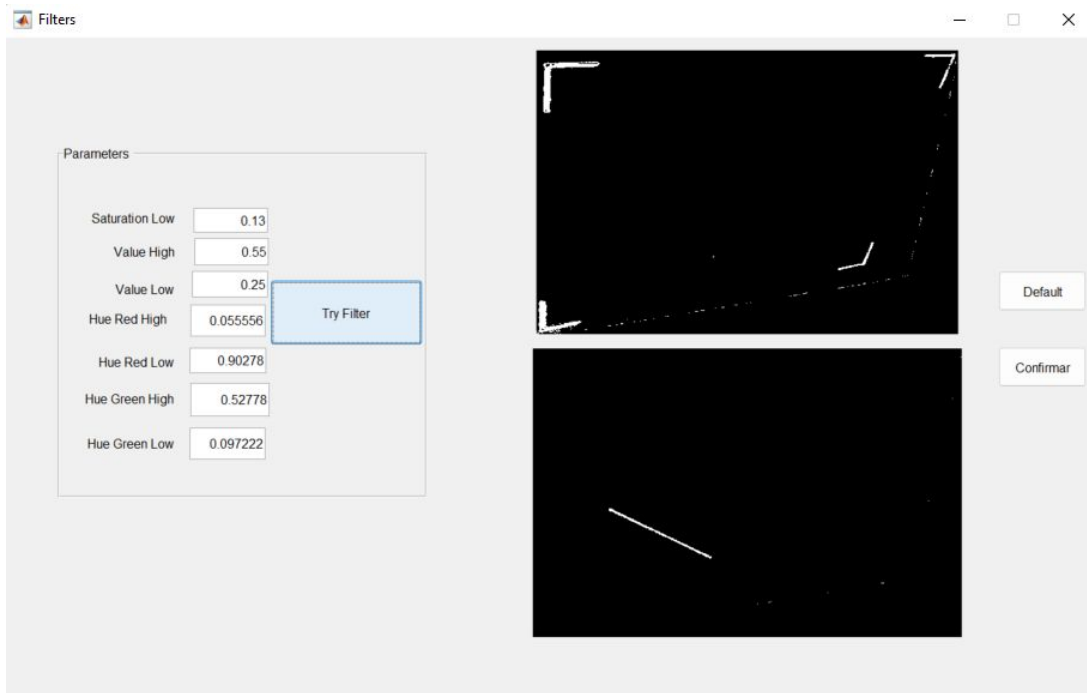
Mostrar espacio alcanzable

| | | | |
|--------------|-----|----------------------|----------------------|
| $ \theta_1 $ | 130 | <input type="text"/> | <input type="text"/> |
| $ \theta_2 $ | 70 | <input type="text"/> | <input type="text"/> |
| $ \theta_3 $ | 70 | <input type="text"/> | <input type="text"/> |
| $ \theta_4 $ | 70 | <input type="text"/> | <input type="text"/> |

Integración IV

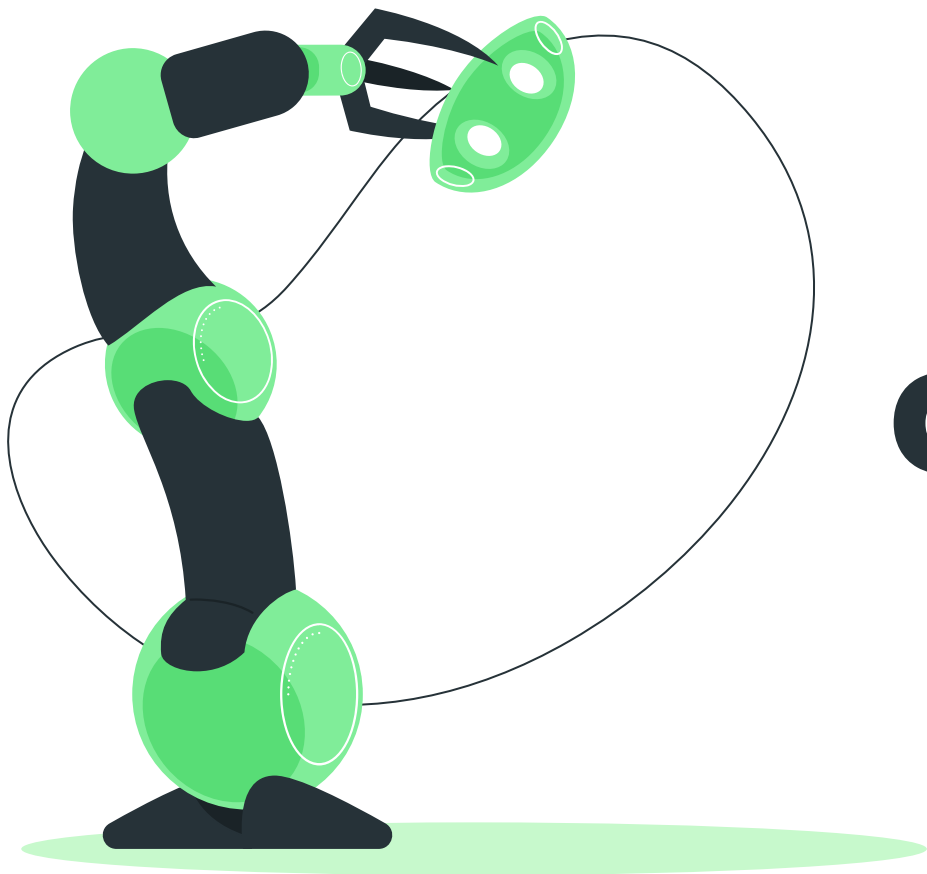


Integración V



Configuración de los filtros empleados en la parte de visión. Esto permite acomodar el filtrado a las condiciones particulares en las que se tomarán las fotos para optimizar el filtrado.

Filtros personalizados



Gracias!