

# **DIMENSIONAMIENTO DE UNA OFICINA MEDIANTE UN SISTEMA DE VISIÓN UTILIZANDO LOS PATRONES CUADRADOS UBICADOS EN TECHOS DE TABLA ROCA**

***Blanca Arlette Serrato Espino***

Tecnológico Nacional de México en Celaya  
13031064@itcelaya.edu.mx

***María del Carmen Tinajero Campos***

Tecnológico Nacional de México en Celaya  
13031063@itcelaya.edu.mx

***Salvador Manuel Malagon Soldara***

Tecnológico Nacional de México en Celaya  
salvador.malagon@itcelaya.edu.mx

## **Resumen**

Este trabajo propone una alternativa para el proceso de desplazamiento en una oficina mediante un sistema de visión, siendo éste el primer paso para un posterior dimensionamiento del área de trabajo.

Muchas oficinas cuentan con un techo de tabla roca cuyos marcos son de aluminio en una forma particularmente cuadrada. Es importante resaltar que se trata de un ambiente controlado, donde los factores como luz, sombra, ángulos de inclinación de la toma de imágenes, entre otros; tienen una influencia destacable en los resultados obtenidos.

Para el desarrollo de este trabajo se utilizó una *Raspberry Pi 3*, una cámara web, un monitor, un teclado, un mouse, un editor de programación (Ninja) y el compilador Python con las librerías *OpenCV* y *NumPy*.

Obteniendo resultados favorables, ya que se logró la correcta detección de las líneas presentes para el cálculo de los cuadros que dimensionan el área de trabajo.

Para la realización de este trabajo se siguió el método de investigación científico exploratorio, ya que se pretende dar una visión general de tipo aproximado respecto a una determinada realidad.

**Palabra(s) Clave:** Líneas, OpenCV, Python, Visión.

## **Abstract**

*This work puts forward an alternative for the process of displacement in an office through a vision system, being this the first step for the dimensioning of the work area.*

*Many offices have a sheetrock ceiling whose frames are made of aluminum in a particularly square shape. It is important to emphasize that it is a controlled environment, where factors such as light, shadow, inclination angles of the image taken, and so on; they have an outstanding influence on the results obtained.*

*For the development of this work we used a Raspberry Pi 3, a webcam, a monitor, a keyboard, a mouse, a programming editor (Ninja) and the Python compiler with the OpenCV libraries and NumPy.*

*Obtaining favorable results, since the correct detection of the present lines was obtained for the calculation of the tables that dimension the work area.*

*For the accomplishment of this work the method of exploratory scientific investigation was followed, since it is intended to give an approximate general view of a certain reality.*

**Keywords:** *Lines, OpenCV, Python, Vision.*

## **1. Introducción**

Las imágenes de los techos de las oficinas están segmentadas con líneas rectas horizontales y verticales que forman cuadrados, lo cual nos lleva a sugerir el utilizar comandos establecidos mediante un sistema de visión para lograr el dimensionamiento del área de trabajo; debido a que ya se había logrado implementar satisfactoriamente el uso de sistemas de visión para la detección de líneas por medio de comandos de programación ya aprendidos con anterioridad en experiencias pasadas.

“El procesamiento de imágenes es la primera etapa a la que se someten las imágenes de la escena analizada con objeto de intentar determinar de forma automática su contenido. En este sentido hay que considerar el procesamiento de imágenes como el conjunto de técnicas que permiten transformar una imagen digital, con el fin de facilitar las etapas posteriores de segmentación y reconocimiento” [7].

“La transformación de imágenes se refiere a todos aquellos procesos que permiten obtener una nueva imagen  $g(x, y)$  a partir de la imagen original  $f(x, y)$ . Donde el objetivo es conseguir que la imagen de salida mejore en alguna de sus características” [6].

Por otra parte, para comenzar el procesamiento, es esencial conocer algunas de las características más importantes en la constitución de una imagen. Un borde es una frontera entre dos regiones con propiedades de nivel de gris o tono relativamente distintas, es necesario que las regiones en cuestión sean suficientemente homogéneas para que la transición entre dos de ellas se pueda determinar solamente sobre la base de estas discontinuidades [1].

“Llamamos línea recta al lugar geométrico de los puntos tales que tomando dos puntos diferentes cualesquiera  $P_1(x_1, y_1)$  y  $P_2(x_2, y_2)$  del lugar, el valor de la pendiente  $m$  resulta siempre constante” [5]. Esta definición no puede aplicarse a una recta paralela al eje  $y$ , porque no tiene pendiente, tiende al infinito. La recta también se puede definir, analíticamente, como “una ecuación lineal o de primer grado con dos variables” [4].

## **2. Métodos**

Para detectar las líneas horizontales y verticales, mostradas en la imagen en perspectiva, se utilizó el método de la Transformada Estándar de Hough.

“La transformada de Hough es un método de extracción de características patentado por Paul Hough en 1962, para la detección de bordes en imágenes parametrizadas, es decir, los objetos que se desean detectar se representan por medio de una ecuación conocida. La idea es transformar la imagen del espacio cartesiano  $xy$  a un espacio de parámetros y a través de un proceso de votos

identificar los puntos que describen una línea recta u otra formulación de bordes” [2].

“Los valores usados para los parámetros de la función *HoughLines* de *OpenCV* son: 1 para la resolución de  $\rho$ ,  $\pi/360$  radianes ( $0,5^\circ$ ) para la resolución de  $\theta$ , y  $193\pm 5$ , para el umbral del acumulador. El método *HoughLines* devuelve una lista de la líneas detectadas en la imagen que se está procesando y cada una definida en coordenadas polares  $(\rho, \theta)$ ” [6].

Del conjunto de líneas obtenidas se etiquetan aquellas donde:

- $1.66 > \theta < 1.48\text{rad}$ , como horizontales, paralelas al eje polar de referencia.
- $-0.09\text{rad} > \theta < 0.09\text{rad}$ , como verticales.
- Se trata de un rango cercano al tipo de recta que define la etiqueta.
- Los parámetros para el método *Canny* de *OpenCV* utilizados, según López Correa, fueron:

- ✓ Umbral inferior para la histéresis = 50
- ✓ Umbral superior para la histéresis = 150
- ✓ Apertura para el operador de Sobel = 3
- ✓ Los materiales utilizados fueron:
- ✓ *Raspberry Pi 3*
- ✓ Cámara web
- ✓ Pantalla
- ✓ Teclado
- ✓ Mouse
- ✓ Ambiente controlado

El proceso general se presenta en la figura 1. Éste se compone de varios pasos para lograr el objetivo que se plantea. Para generar el código primeramente fue necesario importar las librerías *cv2* y *NumPy*. Para realizar el código se utilizaron comandos especiales que se irán detallando poco a poco según su función en el programa [3].

Se propuso una función interna llamada *onMouse* para que cuando se oprimiera el clic derecho del mouse se realizara un evento y la condición cambiara a

verdadera. Luego se procedió a capturar el video utilizando el comando `cv2.VideoCapture()`, así como a nombrar la ventana para ser usada como un marcador de posición para imágenes y barras de seguimiento, donde fue referenciada por el nombre donde éste es el identificador utilizando el comando `cv2.namedWindow('MyWindowLAlala')`. Posteriormente se utilizó el comando `cv2.setMouseCallback('MyWindowLAlala', onMouse)` que llamaba a la función interna declarada `onMouse` donde se estableció como controlador al mouse para una ventana especificada.

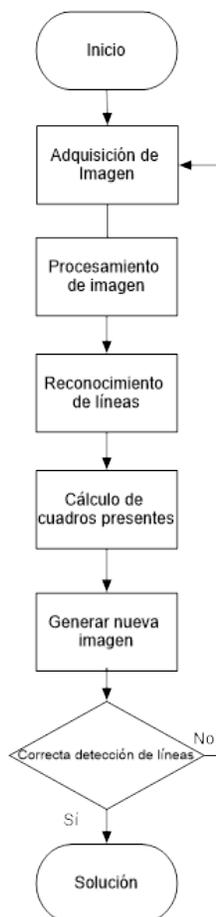


Figura 1 Proceso General.

A continuación, se utilizó el comando `cv2.cameraCapture.read()` que sirve para grabar, decodificar y regresar un cuadro del video (*frame*) cuyo argumento fue el nombre de la ventana donde se obtendrá el *frame*. La cámara fue tomando un video con un ciclo *while* que iba revisando cada milisegundo si ya se había dado

un *clic* del mouse y se detendría hasta que ocurra dicho *clic* del mouse. También se utilizó el comando `cv2.imshow('MyWindowLAlala',frame)` para mostrar la imagen obtenida en una ventana específica, donde sus parámetros eran el nombre de la ventana establecida y la imagen fue mostrada. Enseguida, se utilizó el comando `cv2.imwrite('MyWindowLAlala.jpg',frame)` para salvar la imagen en un archivo especificado con un nombre y el formato del archivo (*jpg* o cualquier otro). Para destruir la ventana generada se utilizó el comando `cv2.destroyWindow('MyWindowLAlala')`.

Después, se comenzó a procesar la imagen utilizando el comando `cv2.imread('MyWindowLAlala.jpg')` que cargaba una imagen de un archivo creado anteriormente. Para convertir una imagen de un espacio de color a otro se utilizó el comando `cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)` donde sus parámetros eran la entrada de la imagen, la salida de la imagen del mismo tamaño que el de la entrada y el código de conversión del espacio de color. Además, se utilizó el comando `cv2.Canny(gray,50,150,apertureSize=3)` para configurar los parámetros en el método *Canny* explicado anteriormente. Se prosiguió a encontrar las líneas en una imagen usando la transformación estándar de Hough, utilizando el comando `cv2.HoughLines(edges,1,np.pi/180*10,100)`, donde los parámetros eran la imagen que sacaba la función `cv2.canny(edges)`, *Rho* que era igual a 1 representa la resolución de distancia del acumulador de pixeles,  $np.pi/180*10$  representa *theta* que es la resolución angular del acumulador en radianes, el 100 es el umbral del acumulador, el cual hace que solo se devuelvan las líneas que obtengan suficientes votos ( $> threshold$ ).

Para realizar el cálculo del número de líneas se utilizaron dos ciclos anidados *for*, donde se generaban dos puntos y su pendiente para generar una línea con el comando `cv2.line(img,(x1,y1),(x2,y2),(0,0,255),2)` y así ir contando el número de líneas que fueron creadas.

Se debe de tomar en cuenta que al detectar dos líneas paralelas significa que es una arista de un cuadrado. Por lo cual, se procedió a realizar operaciones aritméticas para calcular el número de cuadrados presentes en la imagen.

Se obtiene el total dividiendo el número de líneas detectadas entre cuatro (ecuación 1), así como también se obtiene el residuo del número de líneas detectadas con la operación de módulo tomando en cuenta que cuatro líneas conforman un cuadrado (ecuación 2), y este residuo se multiplicó por diez para no trabajar con decimales y luego dividirlo entre cuatro (ecuación 3).

$$total = \frac{num}{4} \quad (1)$$

$$res = num \% 4 \quad (2)$$

$$residuo = \frac{res * 10}{4} \quad (3)$$

Según el residuo obtenido se procedió a realizar determinada operación, si el residuo era cero quería decir que el área total estaba conformada por el mismo número de líneas tanto horizontalmente como verticalmente, y como se mencionó que dos líneas representan un lado de un cuadrado es necesario restarle una unidad al *total* obtenido anteriormente (ecuación 4) y luego se elevaba al cuadrado (ecuación 5) obteniendo así el número total de cuadrados presentes.

$$total1 = total - 1 \quad (4)$$

$$fin = total1 * total1 \quad (5)$$

Si el residuo era cinco quería decir que el área total estaba conformada de tal manera que el número de líneas horizontales era mayor al número de cuadros verticales o viceversa, como se mencionó que dos líneas representan un lado de un cuadrado es necesario restarle una unidad al *total* obtenido anteriormente (ecuación 6) y luego se multiplicaba el *total* obtenido primeramente por el *total2* obtenido de la resta (ecuación 7) obteniendo así el número total de cuadrados presentes.

$$total2 = total - 1 \quad (6)$$

$$fin = total * total2 \quad (7)$$

Si el residuo era diferente a cualquier resultado de los mencionados anteriormente quería decir que había ocurrido alguna falla en la detección correcta de las líneas presentes.

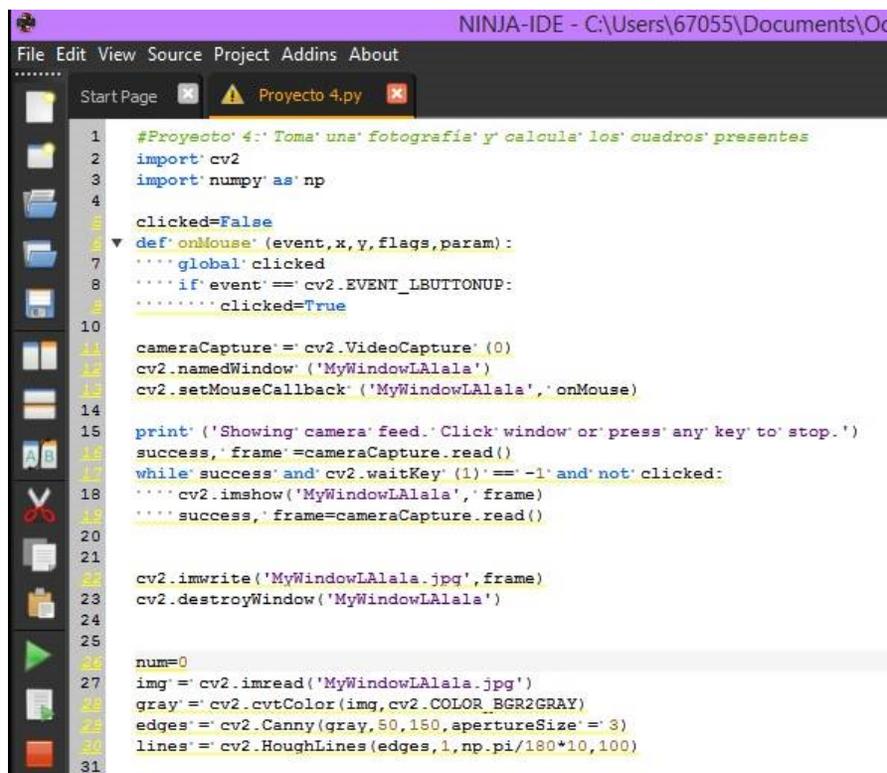
Al realizar el procedimiento, primeramente, se tomó la fotografía con una cámara conectada a la Raspberry Pi 3 una vez que se haya dado un clic sobre la pantalla,

enseguida se guardó la imagen que fue procesada posteriormente para obtener el número de líneas que tiene la imagen, para después realizar cálculos y así obtener el número de cuadros detectados, además de generar una nueva imagen donde se pudieran observar las líneas que el programa había reconocido.

En la figura 2 se muestra la parte del código utilizado donde se adquiere la imagen y se realiza el procesamiento de la misma, así como en la figura 3 hace referencia a la parte del código donde se hace el reconocimiento de las líneas, el cálculo de cuadros y la generación de la nueva imagen resultante.

### 3. Resultados

La fotografía tomada en la figura 4 se realizó con una cámara conectada a la Raspberry Pi 3 mostrada en la figura 5 bajo un ambiente controlado haciendo referencia en la figura 6, ya que los factores de luz, sombra y ángulo de inclinación de la fotografía tomada, entre otros, afectaban el resultado final.

The image shows a screenshot of a code editor window titled 'NINJA-IDE - C:\Users\67055\Documents\Oc'. The editor has a menu bar with 'File', 'Edit', 'View', 'Source', 'Project', 'Addins', and 'About'. Below the menu bar, there are two tabs: 'Start Page' and 'Proyecto 4.py'. The code in the editor is as follows:

```
1 #Proyecto 4: Toma una fotografía y calcula los cuadros presentes
2 import cv2
3 import numpy as np
4
5 clicked=False
6
7 def onMouse(event,x,y,flags,param):
8     '''global clicked
9     '''
10    if event==cv2.EVENT_LBUTTONDOWN:
11        clicked=True
12
13
14 cameraCapture=cv2.VideoCapture(0)
15 cv2.namedWindow('MyWindowLAlala')
16 cv2.setMouseCallback('MyWindowLAlala',onMouse)
17
18 print('Showing camera feed. Click window or press any key to stop.')
19 success,frame=cameraCapture.read()
20 while success and cv2.waitKey(1)!='-1' and not clicked:
21     '''cv2.imshow('MyWindowLAlala',frame)
22     '''
23     success,frame=cameraCapture.read()
24
25
26
27 cv2.imwrite('MyWindowLAlala.jpg',frame)
28 cv2.destroyAllWindows()
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Figura 2 Primera parte del código.

```
NINJA-IDE - C:\Users\6
File Edit View Source Project Addins About
.....
Start Page [x] Proyecto 4.py [x]
34 ..... for rho,theta in lines[x]:
35 .....     num=num+1
36 .....     a= np.cos(theta)
37 .....     b= np.sin(theta)
38 .....     x0= a*rho
39 .....     y0= b*rho
40 .....     x1= int(x0 + 1000*(-b))
41 .....     y1= int(y0 + 1000*(a))
42 .....     x2= int(x0 - 1000*(-b))
43 .....     y2= int(y0 - 1000*(a))
44 .....     cv2.line(img, (x1, y1), (x2, y2), (0,0,255),2)
45
46 print('EL NUMERO DE LINEAS SON:')
47 print(num)
48
49 total=num/4
50
51 res=num%4
52
53 residuo=res*10/4
54
55
56 if residuo==0:
57     total1=total-1
58     fin=total1*total1
59 elif residuo==5:
60     total2=total-1
61     fin=total*total2
62 else:
63     print("fallo")
64     fin=0
65
66 print('EL TOTAL DE CUADROS SON:')
67 print(fin)
```

Figura 3 Segunda parte del código.



Figura 4 Fotografía tomada.

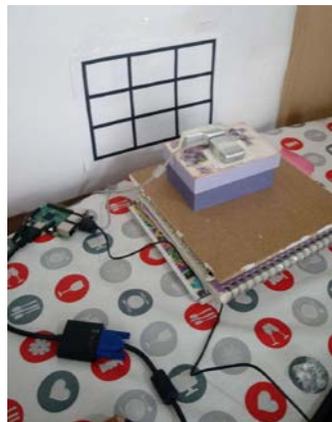


Figura 5 Cámara conectada a la Raspberry Pi 3.



Figura 6 Ambiente controlado.

El resultado mostrado en la figura 7 es el obtenido del programa al ejecutarse en Ninja fue que encontró ocho líneas por lo tanto es un cuadro.

```
File Edit View Source Project Addins About
Start Page x Calcula los cuadros en una imagen.py x
36 if residuo==0:
37     total1=total-1
38     fin=total1*total1
39 elif residuo==5:
40     total2=total-1
41     fin=total*total2
42 else:
43     print("fallo")
44     fin=0
45 print("EL TOTAL DE CUADROS SON: ")
46 print(fin)
47
48 cv2.imwrite('houghlines3.jpg',img)
49
50
Running: C:\Users\67055\Documents\Octavo semestre\Sistemas Inteligentes\L
EL NUMERO DE LINEAS SON:
8
EL TOTAL DE CUADROS SON:
1.0
Execution Successful!
```

Figura 7 Resultados obtenidos.

Por lo cual se observó un correcto resultado, pero fue necesario generar la imagen con las líneas detectadas para saber que éstas fueron correctamente calculadas, así que con un comando especial se generó la imagen mostrada en la figura 8. Con lo cual se logró observar que el programa detectó correctamente cada línea presente en la imagen.

Se realizó el mismo procedimiento para otros casos con mayor número de líneas, tomando la fotografía mostrada en la figura 9, en la figura 10 se obtuvieron los resultados de Python y se corroboraron dichos resultados con la imagen generada en la figura 11.

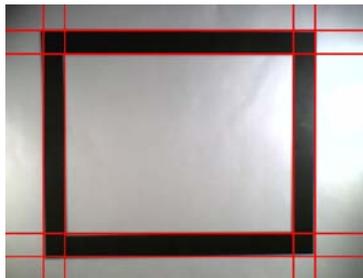


Figura 8 Imagen generada.



Figura 9 Fotografía tomada.

#### **4. Discusión**

Se obtuvieron los resultados deseados usando la técnica de transformación de línea de Hough aplicándola con la función *cv2.HoughLines*. Esta técnica se aplicó a imágenes de cuadrados impresos en tinta negra, aunque no es una fotografía de los patrones cuadrados ubicados en techos de tabla roca en oficinas, esta investigación presenta los principios básicos para la futura aplicación en imágenes tomadas en oficinas.

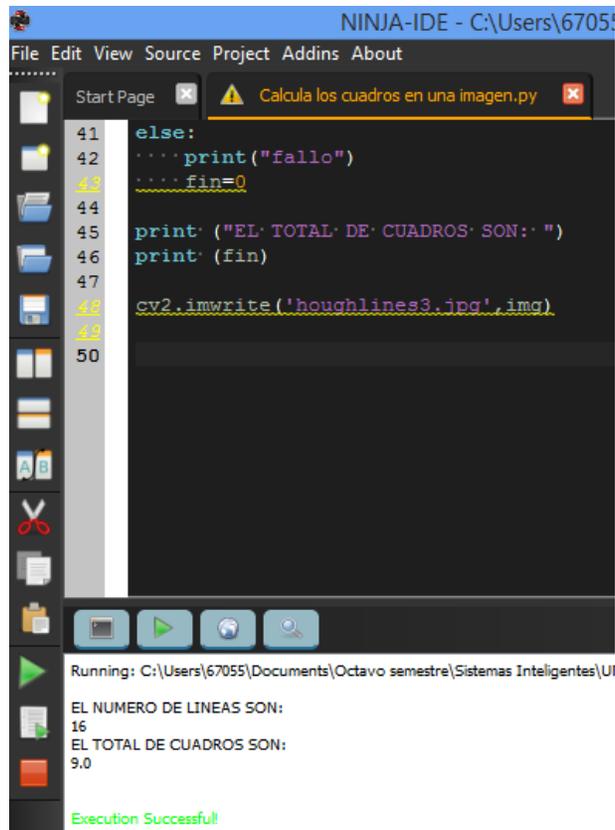


Figura 10 Resultados obtenidos.



Figura 11 Imagen generada.

## 5. Bibliografía y Referencias

- [1] González, R. & Woods, R. (2002). *Digital Image Processing* (2a. ed.). USA: Prentice Hall. ISBN 9780201180756.
- [2] Hough, P. (18 de Dic de 1962). *US Patente n° 3069654*.
- [3] Howse, J. (2003). *OpenCV Computer Vision with Python*. USA: Packt Publishing.

- [4] Kindle, J. (1989). *Teoría y problemas de geometría analítica plana y del espacio*. Compendios Schaum. McGraw-Hill. ISBN 9789684229488.
- [5] Lehmann, C. (1989). *Geometría Analítica*. Limusa. ISBN 9681811763.
- [6] López Correa, Ma. (2015). *Ayuda a la localización de un robot móvil autónomo en los pasillos de un supermercado por medio de métodos de visión. Tesis para optar al título de Maestra en Inteligencia Artificial*. México: Universidad Veracruzana.
- [7] Sánchez, J. (2002). *Avances en robótica y visión por computador. Ciencia y técnica*. España: Ediciones de la Universidad de Castilla-La Mancha: <http://books.google.com.mx/books?id=VeXwzEIngQC>.