

Un algoritmo para la detección de horizonte en dispositivos voladores radiocontrolados

Pablo Odorico¹ y Claudio Delrieux²

¹ Departamento de Cs. e Ing. de la Computación

² Departamento de Ing. Eléctrica y Computadoras
Universidad Nacional del Sur - claudio@acm.org

Parcialmente financiado por la SECyT-UNS

Abstract

Presentamos un algoritmo para la detección no supervisada en tiempo real del horizonte en imágenes de video. El mismo está diseñado para ser implementado en dispositivos autónomos de bajo costo y capacidad de cómputo (microcontroladores), como para ser incorporados en vehículos voladores no tripulados (UAVs). Esto permitirá la obtención de información indispensable para la orientación, navegación, georeferenciamiento y la elaboración de mosaicos de imágenes.

El algoritmo se basa en una función de costo para los pixels de un submuestreo de la imagen en las coordenadas YI del espacio cromático YIQ. Con una baja cantidad de evaluaciones el algoritmo determina la orientación más probable del horizonte, y luego refina la búsqueda con un método adaptativo. La implementación de un prototipo en una PC muestra que el algoritmo es lo suficientemente robusto como para determinar la posición del horizonte con muy bajo costo computacional.

Palabras Clave: PROCESAMIENTO DE IMÁGENES — PROCESAMIENTO DE VIDEO — SENSADO REMOTO

1. Introducción

La obtención de imágenes para sensado remoto (aéreas, satelitales, etc.) es siempre un recurso costoso y de disponibilidad limitada. Sin embargo, en zonas geográficas de accesibilidad limitada, estas imágenes representan el único medio primario de obtención de información. Las imágenes satelitales son caras, están disponibles para fechas y condiciones climáticas no negociables. La resolución es aceptable, aunque imágenes satelitales de resolución comparable con las obtenibles con imágenes aéreas (p. ej., imágenes IKONOS) son de un costo prohibitivo.

Las imágenes aéreas, por su parte, se pueden generar para fechas y condiciones climáticas flexibles, pero el costo operativo es muy alto, requieren un post-procesamiento intensivo para georeferenciarlas, así como corregir la distorsión por perspectiva, y ecualizar las luminancias.

Una solución para mejorar el costo operativo consiste en utilizar dispositivos voladores teledirigidos, montando en ellos una cámara con un microcontrolador para realizar la adquisición. El georeferenciamiento se puede conseguir agregando un GPS a la electrónica anterior, lo cual es posible actualmente dado el bajo costo y la alta performance de estos dispositivos. La corrección de la distorsión por perspectiva, por su parte, requiere determinar en tiempo real la orientación de la cámara. Esto se determina por medio de la localización del horizonte.

Los vehículos voladores no tripulados (UAVs) se caracterizan por su bajo momento de inercia, por lo cual son susceptibles a rápidos cambios de orientación y velocidad. Por lo tanto, para garantizar la estabilidad del vuelo, y así mismo del *tracking* de la toma de video, la determinación de la posición del horizonte es esencial [6]. Sin embargo, este es un proceso que puede resultar computacionalmente costoso, teniendo en cuenta las limitadas capacidades del hardware a ser incorporado en el dispositivo volador.

En este trabajo proponemos un algoritmo rápido y preciso para la determinación no supervisada del horizonte. El mismo se basa en evaluar un subconjunto submuestreado de la imagen original. Para cada una de las muestras o pixels examinados, se determina la posición de los mismos en un subespacio cromático YI, evaluando una función de similitud (determinada experimentalmente) con las coordenadas del cielo en dicho espacio. Un algoritmo de tracking recorre las muestras en cuatro posibles direcciones, determinando la orientación óptima de una frontera dinámica entre pixels clasificados como probable cielo y pixels clasificados como improbable cielo. Por último, un ajuste lineal de la frontera encontrada produce el horizonte detectado.

El algoritmo fue implementado en PC, y testeado con videos de prueba tomados con una cámara en movimiento sobre un lugar elevado, simulando condiciones similares a las que se encontrará en el dispositivo volador. Los resultados obtenidos son satisfactorios, y el tiempo de cómputo estimado está completamente dentro de los límites de un microcontrolador moderno, por lo que se estima que la implementación de un prototipo permitirá reproducir estos resultados. Se presentan también mejoras e ideas que permitirán reducir aún más el tiempo de cómputo así como aumentar la precisión de los resultados.

2. Trabajo Previo

La navegación teleasistida de vehículos voladores no tripulados (UAVs) es un tema de creciente interés. En [3, 4] se presenta un algoritmo de estabilización y navegación autónoma guiado por visión computacional, el cual se basa en un método robusto para determinar la posición del horizonte. Este método de detección del horizonte se fundamenta en resolver el siguiente problema de optimización. Sea una recta $y = ax + b$ en coordenadas del viewport, se superpone esa recta a la imagen y se clasifican las muestras del color en el espacio cromático RGB de los

pixels por encima o por debajo de la recta. Para cada grupo de muestras, se computa la varianza 3D en las tres coordenadas del espacio cromático. El método, entonces, busca los valores de pendiente a y ordenada al origen b que minimiza la suma de las varianzas de ambos grupos de muestras.

Si bien la hipótesis de partida de este método es valiosa (el horizonte es una línea que separa dos zonas homogéneas en un frame o cuadro del video), es también evidente que es computacionalmente costoso. Contando con un enlace de radio es posible transmitir el *stream* de video y operar la navegación autónoma con una computadora en tierra. Sin embargo, nuestro propósito consiste en resolver la navegación con un microcontrolador a bordo, por lo cual la complejidad del este método lo hace inadecuado.

En el otro extremo del espectro, es posible pensar en chips dedicados a la detección del horizonte, como la propuesta presentada en [2]. Este integrado provee un sensor de 12×12 fotorreceptores, e implementa un algoritmo de separación lineal basado en la luminancia de los receptores. Si bien los tiempos y los consumos de potencia del integrado son sobresalientes, el algoritmo es rígido respecto de los parámetros dentro del cual opera. Por ejemplo, sería imposible realizar vuelos nocturnos.

Nuestra propuesta, en cambio, busca un equilibrio entre ambas posibilidades (computación en un chip vs. computación en tierra), utilizando dispositivos programables más flexibles y algoritmos de bajo costo. Eventualmente, en caso de ser necesario, los algoritmos pueden ser migrados a dispositivos más eficientes como los FPGA.

3. Descripción del método

Como ya mencionáramos, el método consiste en trabajar con un subsamplado de la imagen, de tamaño determinado dinámicamente de acuerdo con las restricciones que encuentre el algoritmo en tiempo de ejecución. Una mayor cantidad de muestras permite una determinación más precisa del horizonte, con un costo mayor, aunque es más sensato trabajar con una cantidad baja de pixels y luego refinar adaptativamente en las áreas donde se requiere mayor precisión. Para los ejemplos desarrollados en este trabajo, las imágenes originales de 640×480 se submuestrearon a $\downarrow 32$ por coordenada, resultando una resolución de 20×15 para cada cuadro del submuestreo, lo cual resultó ser lo más adecuado en los casos que fueron analizados.

Los pixels del submuestreo (o muestras) se pre-clasifican utilizando como función de costo a una “distancia” d en espacio cromático YIQ [1],

$$d(I, Y) = (2I - Y + 1, 2)^2,$$

la cual resulta de evaluar la distancia Euclídea al cuadrado entre la posición del color del pixel y la recta $2I - Y + 1, 2$ en el espacio YIQ¹. Una distancia menor que 0.2 determina que el contenido del pixel es de un color típicamente cercano al cielo. Es importante destacar que como

¹El valor lineal de la distancia hubiera sido igualmente eficiente para determinar una clasificación, pero el uso de la distancia al cuadrado permite ahorrar una operación de punto flotante por cada muestra considerada.

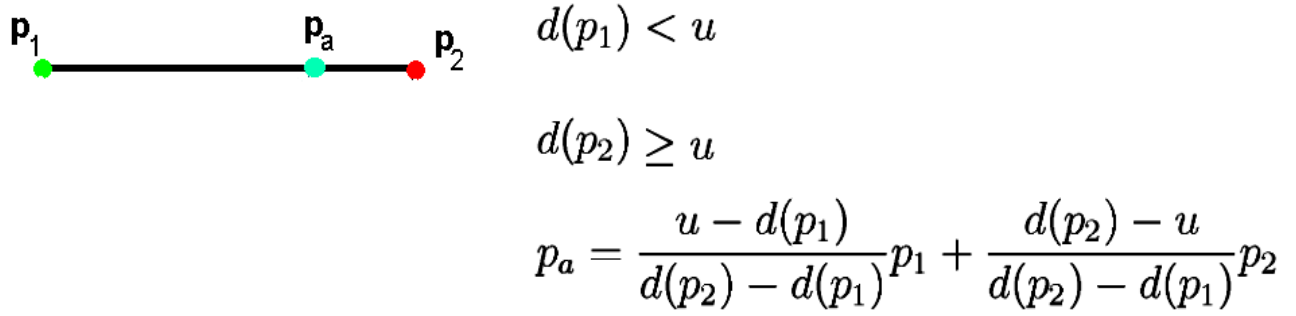


Figura 1: Determinación de un punto de frontera entre dos muestras.

el algoritmo es incremental, no se requiere convertir todos los pixels del cuadro submuestreado a YIQ y calcular el costo, sino solo a un subconjunto dinámicamente determinado de ellos, lo cual reduce el costo computacional sensiblemente. La función de costo, y el umbral mencionado más arriba fueron determinados experimentalmente como los valores que permiten una mejor separación entre las muestras que pertenecen al cielo y los que no.

En una primer etapa, se realiza una estimación de la orientación más probable del horizonte, dividida en cuatro cuadrantes que van de -45 a 45 grados respecto de la horizontal, de 45 a 135, de 135 a -135, y de -135 a -45. Para determinar el cuadrante más probable se hace el cálculo para los cuatro casos, y se elige el caso en el cual el horizonte queda mejor definido. Cada orientación gruesa determina un recorrido de las muestras, respectivamente descendente por columnas, por filas de izquierda a derecha, ascendente por columnas, y por filas de derecha a izquierda.

Por ejemplo, para testear si la orientación más probable del horizonte está entre -45 y 45 grados respecto de la horizontal, se recorren las muestras por columna en sentido descendente, asumiendo que el valor de d de las muestras inferiores están por encima del umbral (de no ser así, se descarta la columna por “anómala”). El recorrido en cada columna termina cuando se cruza el umbral 0.2 en el d , lo cual se considera que ocurre cuando un pixel cruza el horizonte que buscamos. La diferencia entre los d de los dos pixels sucesivos en la columna donde se produjo la transición da una idea de la “confiabilidad” de ese sitio como un punto probable del horizonte. Para cada una de las cuatro posibles orientaciones, la “confiabilidad global” es el acumulado de la confiabilidad de cada uno de los puntos de transición. Finalmente, la orientación más probable es la que obtuvo mejor confiabilidad global.

Los puntos de frontera se computan por interpolación lineal, utilizando la misma idea subyacente del algoritmo “marching squares” [7] (ver Fig. 1). Es importante notar que en un primer momento intentamos realizar el tracking del horizonte por medio de este algoritmo, obteniendo resultados muy pobres y problemáticos, lo cual derivó en la idea que finalmente presentamos aquí. También cabe resaltar que nuestro algoritmo procesa localmente la parte de la imagen submuestreada que es atravesada por el horizonte. En caso de ser necesaria mayor exactitud, el algoritmo puede recurrir a la imagen en un subsampleo más preciso del cuadro original, obteniendo una poligonal de mejor resolución.

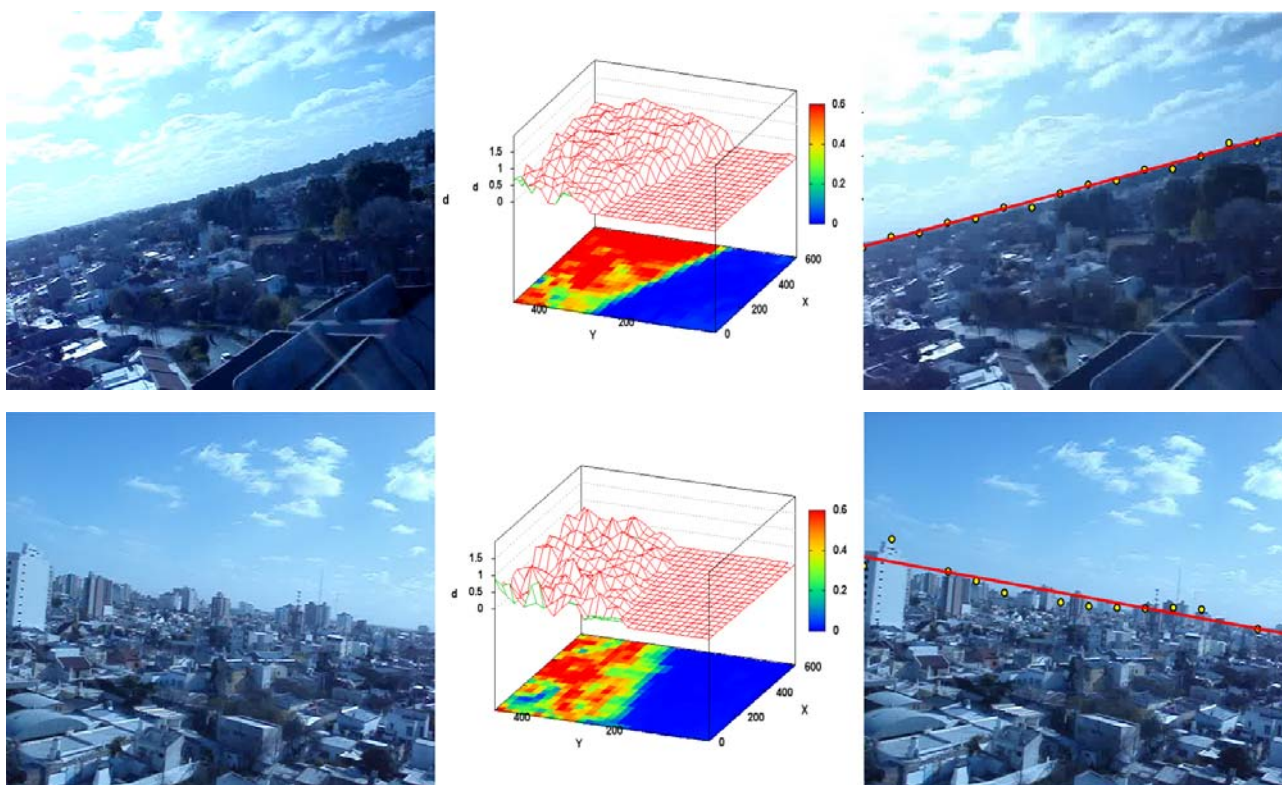


Figura 2: Ejemplos del horizonte encontrado con nuestro método.

En una segunda etapa, los puntos de frontera obtenidos pueden unirse por una poligonal, o bien puede computarse un ajuste lineal, lo cual determina un horizonte promedio que oblitera los obstáculos que pueden presentarse en la línea visual. También es posible deducir la inclinación de la cámara con respecto al plano teniendo en cuenta la elevación media de los puntos del horizonte. En la Fig. 2 es posible ver dos cuadros de un video, la función de costo en cada muestra de la imagen submuestreada, los puntos de frontera encontrados, y el ajuste de estos últimos por regresión lineal.

4. Resultados obtenidos

La performance del algoritmo resultó sobresaliente, tanto en tiempo de cómputo como en la calidad de los resultados. Implementamos un prototipo del método en PC, el *Horizon 1.0*, para poder evaluar los resultados y visualizar sus propiedades. En la Fig. 3 se muestran los resultados en un cuadro de prueba, obtenido de una toma de video con una cámara fotográfica comercial, sin corrección de la temperatura cromática [5], lo cual dificulta aún más la detección del horizonte en tomas diurnas.

En la interfaz del prototipo se muestra el cuadro original, al que se superponen los puntos de frontera obtenidos con el algoritmo. También se muestran la distribución del color de los pixels en el plano YI del espacio cromático YIQ, la superficie de costo obtenida interpolando la función d definida en la Sección anterior sobre todas las muestras en el cuadro, y la inclinación

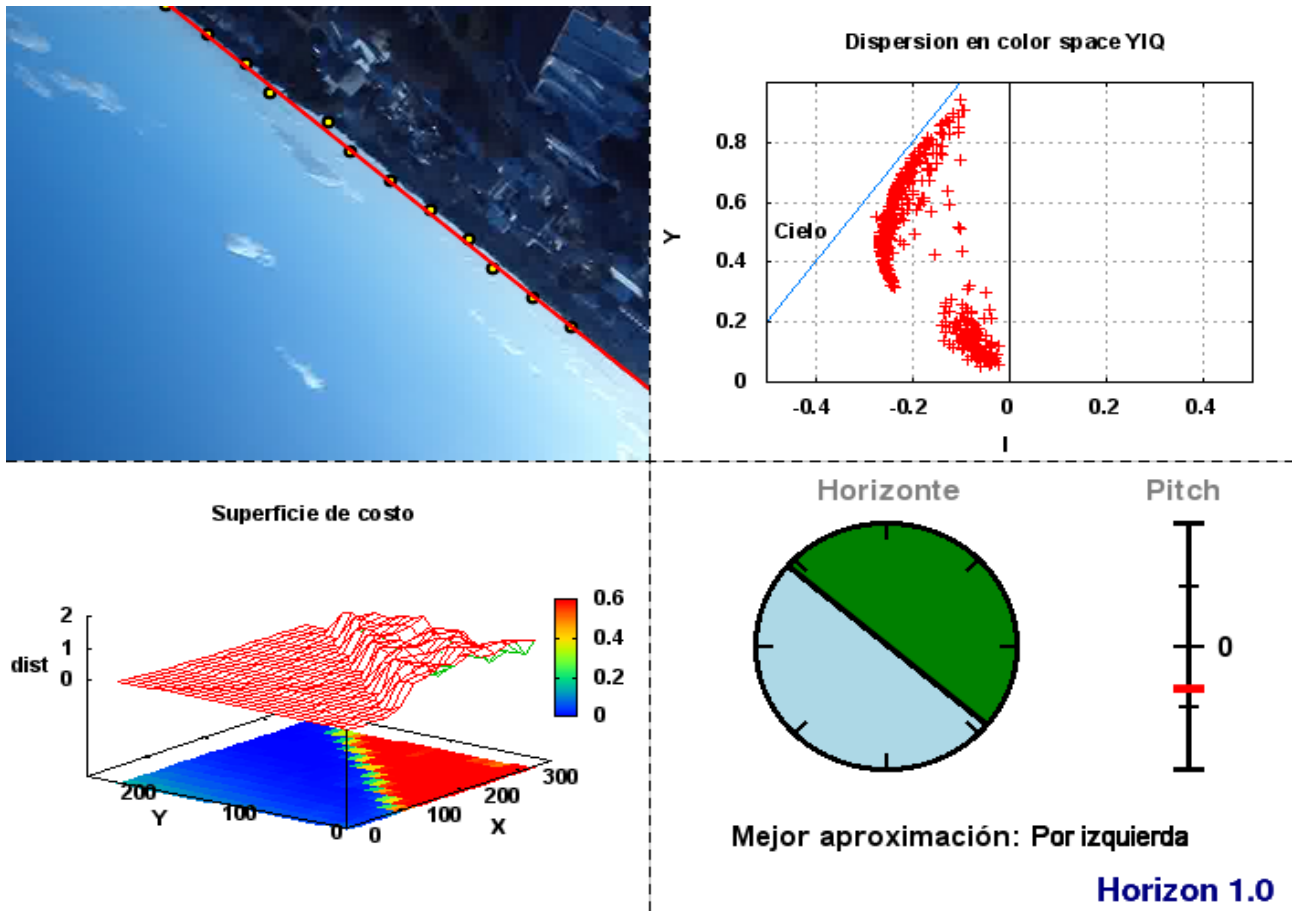


Figura 3: Horizon 1.0 y un cuadro de prueba.

y posición del horizonte virtual. En la Fig. 4 se aprecia cómo el algoritmo de búsqueda de fronteras reacciona bien frente a los obstáculos, en este caso edificios.

El algoritmo está en condiciones de determinar cuándo no existe una línea de horizonte probable. En la Fig. 5 se tomó como cuadro de prueba una parte del video en la cual la cámara apunta hacia el cielo. La primer parte del algoritmo encuentra que no hay puntos que cruzan el valor umbral predeterminado en la función de costo, por lo que la salida del algoritmo es un horizonte nulo.

En la siguiente tabla se consignan las velocidades aproximadas de ejecución para los cuadros en los que la detección del horizonte resultó la más rápida o la más lenta. En los peores casos, con una PC antigua de bajo poder computacional, el método computa varios miles de cuadros por segundo, lo cual permite concluir que su implementación en microcontroladores modernos tendrá una performance más que suficiente para los objetivos requeridos.

CPU	Mejor caso	Peor caso
Pentium 2 MMX	25 Khz.	2 Khz.
Athlon64 3000+	1600 Khz.	16 Khz.

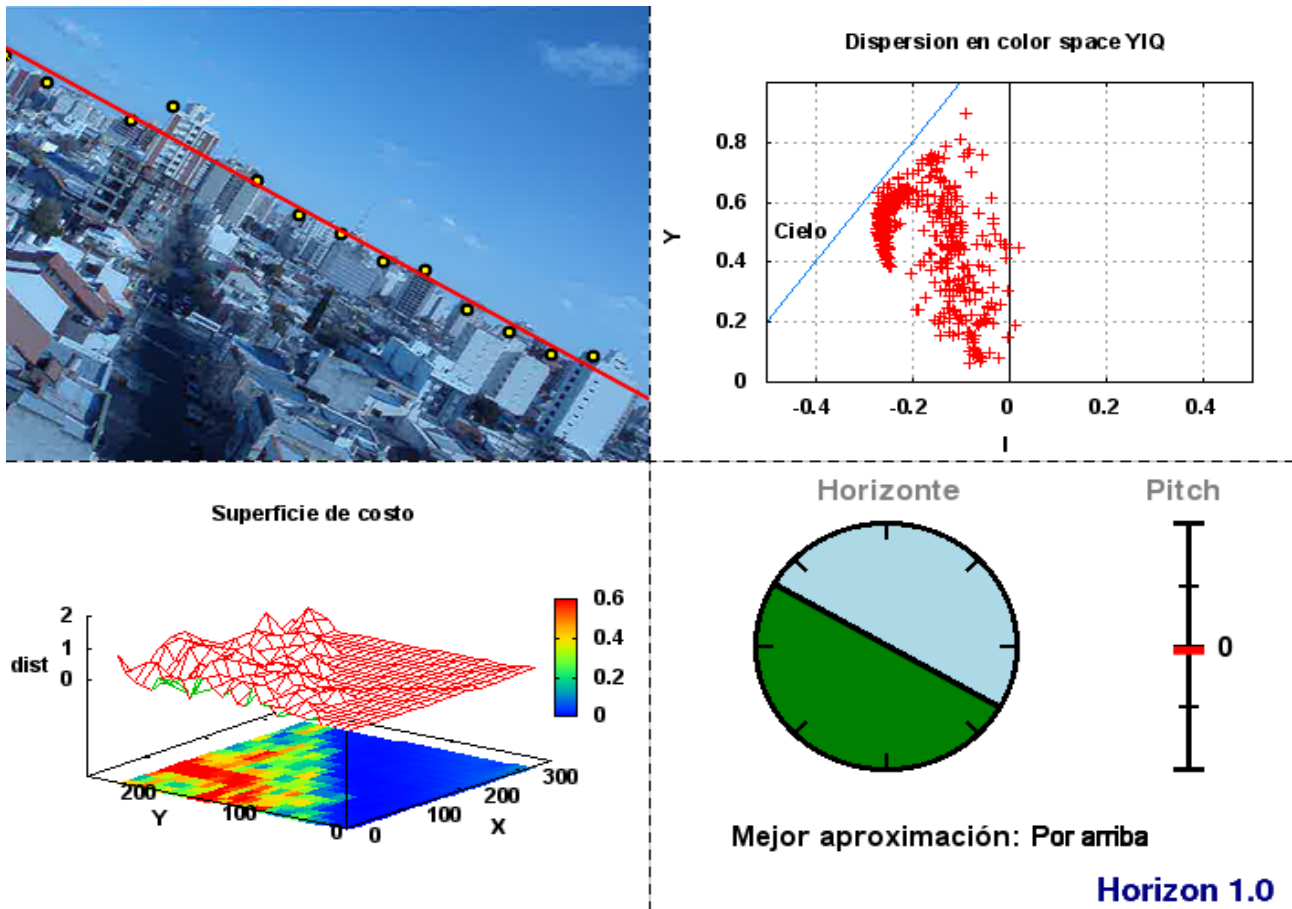


Figura 4: Horizonte con obstáculos.

En <http://www.lip.uns.edu.ar/horizonte> es posible ver un conjunto de videos completos con los resultados del método en tiempo real, tomados en diferentes condiciones y con distintos tipos de obstáculos.

5. Conclusiones y trabajo futuro

Se presentó un algoritmo robusto para la determinación en tiempo real del horizonte en imágenes de video. El mismo se basa en clasificar el color de los pixels por medio de una función de costo en el espacio cromático YIQ. En un primer paso, el método determina la orientación gruesa más probable del horizonte, por medio de una búsqueda del punto de cruce de un valor umbral predeterminado en la función de costo. Luego, el algoritmo realiza un *tracking* entre muestras sucesivas, buscando los puntos de frontera de acuerdo a la función de costo. Finalmente, los puntos de frontera del paso anterior son ajustados linealmente para determinar la posición del horizonte y su inclinación respecto del plano de tierra.

El algoritmo fue implementado en un prototipo en PC en una plataforma de programación de distribución libre, exhibiendo una performance muy buena, tanto la calidad del resultado como en tiempo de ejecución. Los resultados obtenidos en tomas de video muestran que el

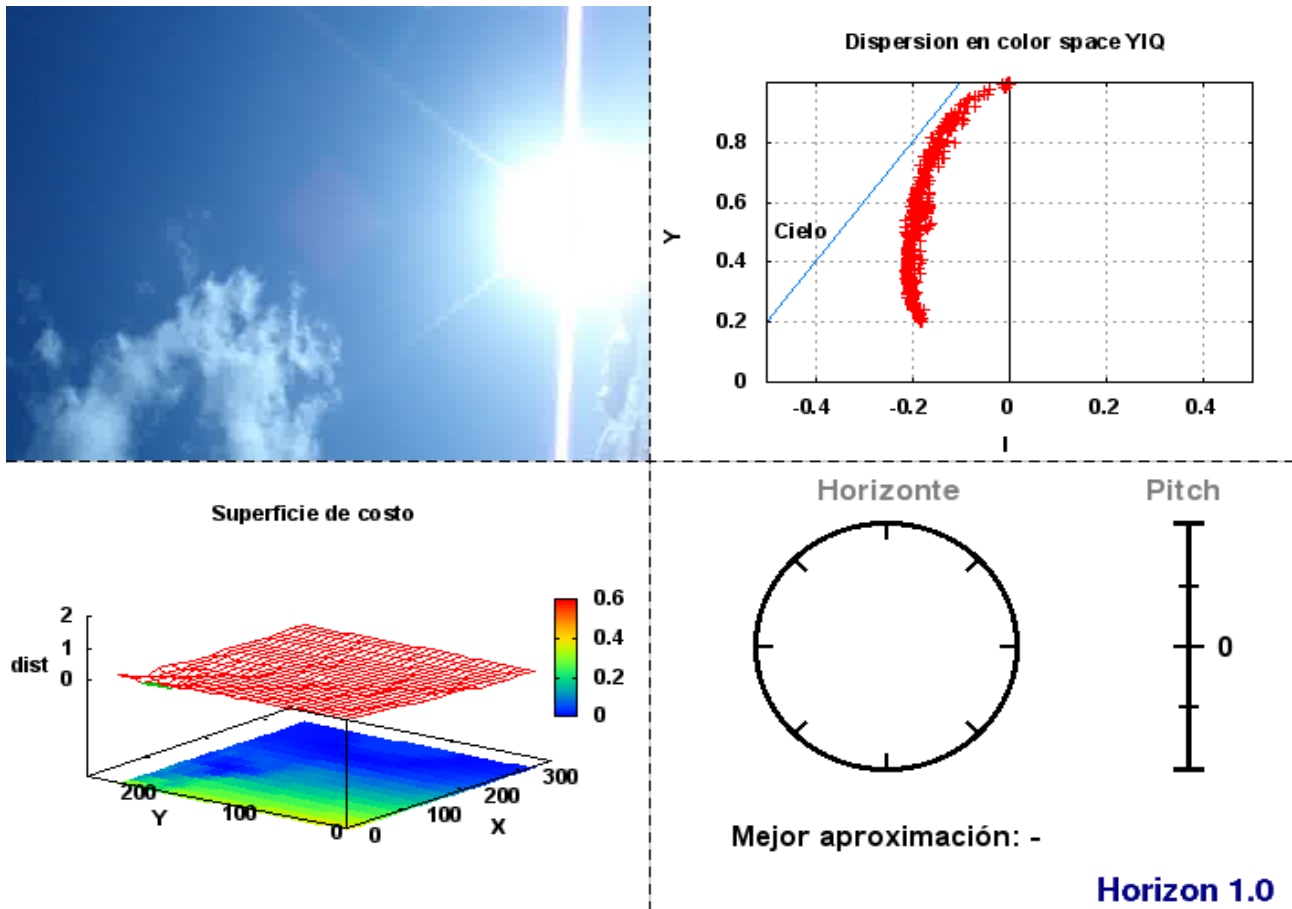


Figura 5: Cuadro de prueba sin horizonte.

algoritmo tiene la capacidad de determinar la posición del horizonte en tiempo real en forma robusta, así como determinar fronteras no lineales que rodean a obstáculos, o decidir si no existe línea de horizonte en un cuadro determinado.

Entre las ventajas determinantes de nuestro algoritmo, podemos mencionar que su implementación computacional es sencilla, por lo que la migración a dispositivos microcontroladores es inmediata. Además, es lo suficientemente flexible como para poder adaptarse a diferentes contextos de uso. En particular, claramente es posible determinar en forma dinámica una relación de compromiso entre la cantidad de cuadros por segundo procesados y la resolución o calidad de la frontera encontrada. Estas dos ventajas lo destacan respecto de los demás algoritmos publicados hasta ahora.

Como trabajos futuros a realizar, podemos contar la migración del algoritmo a un microcontrolador para determinar si las velocidades de cómputo son adecuadas, y su inclusión dentro de un sistema de navegación asistido. También es de gran importancia poder determinar si el algoritmo es factible de ser utilizado con imágenes infrarrojas, dado que en este caso sería necesario encontrar una función de costo adecuada. Por último, debería ser posible optimizar la búsqueda de la frontera tomando en cuenta la información del último cuadro procesado. Esto permitiría saltar el primer paso del algoritmo y de esa manera aprovechar la capacidad de

cómputo para obtener más cuadros por segundo, o bien mejorar la resolución de la frontera obtenida.

Referencias

- [1] Rafael González and Richard Woods. *Digital Image Processing (Second Edition)*. Addison-Wesley, Wilmington, USA, 2002.
- [2] T. Horiuchi. A Low-Power Visual Horizon Estimation Chip. In *Proceedings of the International Symposium on Circuits and Systems*, pages 4755–4758, Kobe, Japan, 2005. International Circuits and Systems Society, ISCAS.
- [3] S. M. Ettinger. *Design and Implementation of Autonomous Vision-Guided Micro Air Vehicles*. PhD thesis, University of Florida, 2001.
- [4] S. M. Ettinger, M. C. Nechyba, P. G. Ifju and M. Waszak. Vision-guided flight stability and control for micro air vehicles. *Advanced Robotics*, 17(1):617–640, 2003.
- [5] WIKIPEDIA. Color Temperature. http://en.wikipedia.org/wiki/Color_temperature, 20-7-2007.
- [6] WIKIPEDIA. Horizon. <http://en.wikipedia.org/wiki/Horizon>, 20-7-2007.
- [7] WIKIPEDIA. Marching Squares. http://en.wikipedia.org/wiki/Marching_squares, 20-7-2007.