

UNIVERSIDAD NACIONAL DE CÓRDOBA
FACULTAD DE MATEMÁTICA,
ASTRONOMÍA Y FÍSICA

TRABAJO FINAL DE GRADO

Software para la detección de variables hidrodinámicas en laboratorio mediante técnicas de video de alta definición.

Autor: Pérez Paladini, Agustín Daniel.
Director: Ing. Mg. Paolo Gyssels.
Profesor Representante: Dr. Oscar Bustos.

30 de Marzo de 2012

Resumen

En laboratorio se utilizan distintos tipos de instrumentos para medir la altura de ola (y otras variables hidrodinámicas), los cuales en general tienen un alto costo económico y requieren de un gran conocimiento por parte del usuario para su correcta utilización como así también para evitar generar daños físicos a las personas o el instrumento en si.

En este trabajo se plantea el desarrollo de un sistema utilizado para capturar la altura de ola en tiempo real y por medio de una webcam como método alternativo y de bajo costo con respecto a los utilizados actualmente en laboratorio.

Clasificación: I.4.0 - Image processing and computer vision.

Palabras claves: Algoritmos de procesamiento de imágenes, *real time processing*, GPU, CPU, altura superficie libre.

Agradecimientos

Quisiera agradecer en primer lugar a mi queridísima madre por su apoyo y afecto incondicional a lo largo de toda la carrera y la vida. Este trabajo y lo que simboliza va dedicado a ella.

A mi hermana Julieta y mi cuñado Sebastián por las largas charlas y buenos consejos en esta ultima etapa.

A mi novia Cony, por soportar mis quejas y a mi ser en general, por sus aportes anímicos y gráficos a este trabajo.

A mis hermanos de amistad: Gornete, Fedelo, Gringo, ElHacha, Wil que siempre estuvieron al lado mio durante toda mi vida en los buenos y malos momentos.

A mis amigos Mingo, Liber, Fabio, Oso, Hae, Chino, Carlox, Adrian, Ruleman, Mariano, Ramox y demás, por tantas horas y buenos momentos compartidos.

A mi director de tesis Paolo Gyssels por la paciencia y la confianza en mi. Por estar siempre ahí en todo momento cuando lo necesite.

A Oscar Bustos por su continuo positivismo, y su ayuda constante, a Nicolás Wolovick por su apoyo a lo largo de toda la carrera y a Daniel Fridlender por su contribución en este trabajo.

A todos los profesores de la carrera que hicieron de la misma una exelente experiencia.

A toda la gente de la facu por su buena predispcion y ayuda.

Y a toda la gente del laboratorio de hidráulica, en especial a Mariana y Antoine, por su buena onda y por darme la posibilidad de desarrollar el trabajo ahí.

Índice

1. Introducción	6
1.1. Motivaciones	6
1.2. Objetivos	7
2. Fundamentos de hidrodinámica marina	8
2.1. Introducción	8
2.2. Teoría lineal de ondas	10
2.2.1. Oleaje irregular	12
2.3. Análisis del oleaje	14
2.4. Descripción espectral del oleaje	15
2.4.1. Método para la obtención del espectro	15
2.4.2. Parámetros espetrales	17
2.5. Propiedades espetrales del oleaje: espectro de JONSWAP . .	18
2.6. Aplicaciones en laboratorio	19
3. Detección de la superficie libre en laboratorio	21
3.1. Tipo de sensores	21
3.1.1. Instrumentos intrusivos	21
3.1.2. Instrumentos no intrusivos	22
3.2. Técnicas basadas en procesamiento de imágenes	23
3.2.1. Rectificación	25
3.2.2. Detección de contornos	27
3.2.3. Análisis y procesamiento	28
3.3. Técnicas utilizadas	29
3.3.1. Rectificador	31
3.3.2. Recortador	32
3.3.3. Detector de bordes	33
3.3.4. Analizador	34
3.3.5. Proceso completo	34
4. Algoritmos involucrados	36
4.1. Introducción	36
4.2. Perspective transform	36
4.3. Gaussian Blur	37
4.4. Canny Edge Detector	39
4.5. Wave height analyzer	44

5. Implementación y funcionamiento del sistema	46
5.1. Requerimientos principales	46
5.2. Arquitectura	48
5.2.1. A nivel general	48
5.2.2. A nivel de IPS	49
5.3. Diseño	50
5.3.1. Preliminares	50
5.3.2. Nivel estructural o de framework	51
5.3.3. IPS Core	52
5.3.4. IPS GUI	54
5.4. WHIPS	55
5.4.1. Introducción	55
5.4.2. Diseño	56
5.4.3. Funcionamiento WHIPS Core	58
5.4.4. Funcionamiento WHIPS GUI	59
6. Aplicación en un canal 2D	62
6.1. Canal de laboratorio	62
6.2. Preparación de las mediciones	63
6.3. Ensayos propuestos	69
6.4. Mediciones y análisis de los resultados	72
6.4.1. Que medimos?	72
6.4.2. EN1: Oleaje regular	73
6.4.3. EN2: Oleaje irregular de 5 minutos	75
6.4.4. EN3: Oleaje irregular de 30 minutos	77
6.4.5. EN4: Oleaje irregular de 5 minutos	80
6.4.6. EN5: Oleaje irregular de 30 minutos	82
6.5. Consideraciones generales	85
6.5.1. Ángulo y posición de captura	85
6.5.2. Velocidad de captura de la cámara	88
6.5.3. Superficie libre	93
7. Conclusiones	95
8. Trabajos futuros	97

1. Introducción

1.1. Motivaciones

En la modelación física de laboratorio se emplean en muchos casos instrumentos costosos para la medición de variables hidrodinámicas tales como velocidades, variaciones de niveles, presiones, trayectorias de las partículas, turbulencia, o variables de dinámica sedimentaria, como erosiones de playas, erosiones en las pilas de puentes.

Todos los instrumentos modernos de medición de laboratorio requieren una buena experiencia por parte del usuario, como en el caso del PIV¹ (Particle Image Velocimeter), donde se utilizan sensores muy delicados como el láser, que inclusive pueden causar daños a las personas que lo utilizan sin las adecuadas precauciones. En otros casos, los instrumentos que se suelen utilizar son de tipo intrusivos, como en el caso del ADV² (Acoustic Doppler Velocimeter), o de los sensores de nivel resistivos, que se sumergen directamente en el fluido; asimismo, tales instrumentos se estropean con el uso. Además, todos los sensores son específicos para la medición de las variables en un cierto rango de escala espacio-temporal, en algunos casos se utilizan solamente en ensayos bidimensionales, determinan un parámetro en un cierto volumen de fluido, no pueden describir los procesos cerca de los contornos, dependen de la siembra de partículas en el fluido, se corroen si el agua es salada, etc.

En este trabajo se desarrolla un sistema de medición y análisis de variables hidrodinámicas en laboratorio mediante técnicas de vídeo con cámaras web de alta definición. Estas técnicas tienen diferentes ventajas con respecto a los métodos e instrumentos tradicionales: son técnicas no intrusivas, económicas, son fácilmente utilizables por usuarios con poca experiencia en trabajos de laboratorio, y son muy flexibles porque se pueden aplicar a diferentes tipos de mediciones y a diferentes escalas espacio-temporales de los procesos.

La ventaja principal de los instrumentos tradicionales es la elevada precisión de la medición que tienen que realizar. La exactitud de las mediciones que se pueden realizar con vídeo cámaras depende de la definición de las mismas, con lo cual un aumento en la precisión requiere un costo económico mayor en la adquisición de la misma. Sin embargo en aplicaciones ingenieriles de laboratorio, una webcam de alta definición estándar puede ser suficiente para el nivel de detalle requerido. Este aspecto será analizado en este

¹http://en.wikipedia.org/wiki/Particle_image_velocimetry

²http://en.wikipedia.org/wiki/Acoustic_Doppler_velocimetry

proyecto, en particular para realizar mediciones de niveles con la suficiente precisión.

En mediciones de campo, las muestras con vídeo cámaras de alta definición ofrecen ya muchas aplicaciones prácticas y muy funcionales, por ejemplo para la detección de la erosión costera y en la determinación de parámetros de oleaje ([1]).

Las aplicaciones de estas técnicas en laboratorio son múltiples en los campos de la mecánica de fluidos y de la dinámica sedimentaria, entre otros. En principio, se implementará una técnica para la adquisición de datos de altura de ola en tiempo real y su posterior análisis, pero la metodología servirá como punto de partida para las mediciones de otros parámetros hidrodinámicos (como rotura de oleaje, dispersión de contaminante) y sedimento-lógicos de interés (erosión de un perfil de playa).

1.2. Objetivos

El objetivo principal de este trabajo es optimizar la toma de datos de la variable “altura de la superficie libre” (altura de ola) mediante cámaras de alta definición³, durante la realización de ensayos de laboratorio con un canal bidimensional de oleaje. La adquisición de datos de altura de ola en tiempo real se realiza en la actualidad mediante el empleo de sensores de niveles intrusivos (se sumergen en el fluido) de tipo resistivos que brindan una frecuencia de muestreo de hasta 100 Hz. En la mayoría de los trabajos que se realizan en el campo de la ingeniería marítima o de costas, es suficiente un muestreo de 20-25 Hz para medir y procesar la altura de ola que se genera en laboratorio.

Las técnicas de vídeo a implementar deberán en primer lugar rectificar las imágenes a procesar y realizar muestreos de las variables a medir con tomas de imágenes de alta definición con frecuencias igual o superiores a los 20 Hz.

Durante los ensayos estas tomas deberán permitir la visualización en tiempo real de la variación de la variable a medir de manera tal que el funcionamiento de esta metodología pueda ser equivalente (dentro de una precisión admitida) a las mediciones realizadas con los instrumentos tradicionales para estudios en el campo de la ingeniería.

³http://es.wikipedia.org/wiki/Alta_definici%C3%B3n

2. Fundamentos de hidrodinámica marina

2.1. Introducción

Las olas del océano son movimientos ondulatorios que ponen de manifiesto la propagación de energía mecánica entre la superficie marina y la atmósfera. Éstas son generadas por distintos factores naturales (viento, perturbaciones meteorológicas, terremotos, etc). Las olas son perturbaciones que se propagan en la superficie del mar y cuyo movimiento en mar abierto viene amortiguado principalmente por la gravedad.

De manera esquemática, en la *Figura 1* se puede observar la energía de las ondas de superficie asociadas a cada frecuencia. En este trabajo se estudiarán las olas de viento pertenecientes a las frecuencias con períodos entre 1 y 30 segundos (olas de viento cuya fuerza estabilizadora es la gravedad).

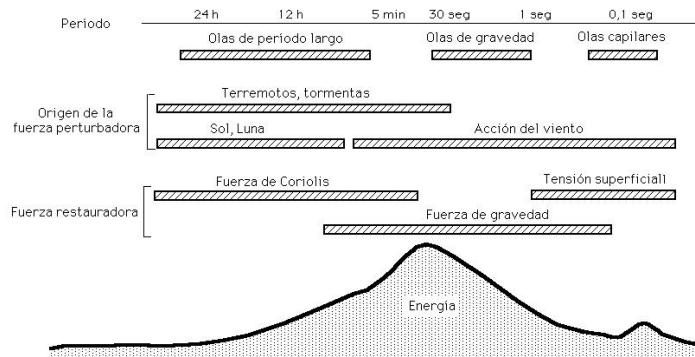


Figura 1: Distribución energética de las ondas de superficie.

Matemáticamente se puede describir a las olas, en forma general, como oscilaciones uniformes y periódicas de la superficie del agua, distinguiendo dos tipos: ondas progresivas (se propagan en una dirección) y ondas estacionarias (aquellas que no se propagan, como por ejemplo las oscilaciones de niveles en cuerpos de agua cerrados o en puertos); en este trabajo se analizan las primeras. Existen distintas magnitudes que caracterizan a las ondas y algunas definiciones útiles para representarlas, estas son (*Figura 2*):

- η : Desplazamiento de la superficie libre (desplazamiento vertical de la superficie del fluido con respecto a un nivel medio de referencia NM (MWL, *Mean Water Level*)).
- T : Período de la onda (tiempo que transcurre entre dos pasos de un punto idéntico de la ola).

- L : Longitud de la onda (distancia entre dos puntos idénticos de la ola).
- *cresta*: Punto en el que la superficie libre alcanza un máximo.
- *seno*: Punto en el que la superficie libre alcanza un mínimo.
- *paso por cero*: Puntos en el que la superficie libre interseca al nivel medio de referencia.
- H : Altura de onda (distancia vertical total entre seno y cresta).
- A_c : Amplitud de la cresta (distancia vertical máxima entre nivel de reposo y cresta).
- A_s : Amplitud del seno (distancia vertical máxima entre nivel de reposo y seno).
- *nivel en reposo*: NR (SWL, *Still Water Level*) nivel correspondiente en ausencia de olas.
- h : profundidad o calado (distancia entre el fondo y el NR, en caso de presencia de ondas el nivel medio no es NR ya que este viene modificado usándose en cambio a NM como referencia).

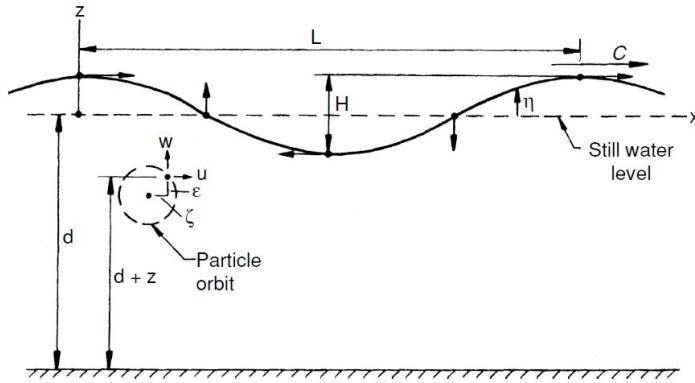


Figura 2: Parámetros asociados a la onda (extraída de [2])

En general, las teorías de ondas, tienen como objetivo calcular la *celeridad o velocidad de fase* (cuando las magnitudes H , L o h son conocidas), que es la velocidad de propagación C , definida como

$$C = \frac{L}{T} [\text{m/s}] \quad o \quad C = \frac{\omega}{\kappa} [\text{m/s}] \quad (1)$$

donde κ es el *numero de onda* definida como

$$\kappa = \frac{2\pi}{L} [\text{rad/m}] \quad (2)$$

y ω la *frecuencia angular* definida como

$$\omega = \frac{2\pi}{T} [\text{rad/s}] \quad (3)$$

Utilizando estos parámetros se puede caracterizar (usando la teoría lineal) la superficie libre de las ondas de tipo estacionarias como

$$\eta(x, t) = A \cos \kappa x \cos \omega t \quad (4)$$

las cuales se caracterizan por tener su evolución espacial y temporal desacoplada. Este tipo de ondas no progresan en el espacio oscilando verticalmente entre puntos fijos llamados nodos.

Por otro lado se encuentran las ondas progresivas, donde su movimiento temporal y espacial están acoplados, dando lugar a la siguiente formula, la cual caracteriza su superficie libre

$$\eta(x, t) = A \cos(\kappa x - \omega t) \quad (5)$$

propagándose y manteniendo su forma en la dirección positiva del eje x a una velocidad C .

2.2. Teoría lineal de ondas

La teoría lineal de ondas es una simplificación matemática al problema de la descripción del movimiento y transformación de las ondas en el mar. Tiene ventajas en el sentido que (i) es una teoría sencilla, (ii) posibilita la obtención de soluciones analíticas, (iii) puede usarse como base para obtener otras soluciones por superposición.

En esta teoría existe un eje horizontal de simetría que es el nivel de reposo donde se cumple que $A_c = A_s$ lo que nos da $H = A_c + A_s$ donde $A_c = A_s = A$ quedando $H = 2A$, donde A se define como amplitud.

El problema se plantea definiendo las ecuaciones que gobiernan el movimiento del fluido, la ecuación de continuidad y las ecuaciones de momentum (que deriva directamente de la 2da ley de Newton). Con este planteamiento se tienen 4 variables: las tres componentes de la velocidad u, v, w y la presión

p. Bajo ciertas hipótesis simplificativas (fluido incomprimible, gravedad como única fuerza externa, fluido no viscoso) las ecuaciones llegan a tener la siguiente forma [3]:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \quad (6)$$

$$\begin{aligned} \frac{Du}{Dt} &= \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} = -\frac{1}{\rho} \frac{\partial p}{\partial x} \\ \frac{Dv}{Dt} &= \frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + w \frac{\partial v}{\partial z} = -\frac{1}{\rho} \frac{\partial p}{\partial y} \\ \frac{Dw}{Dt} &= \frac{\partial w}{\partial t} + u \frac{\partial w}{\partial x} + v \frac{\partial w}{\partial y} + w \frac{\partial w}{\partial z} = -\frac{1}{\rho} \frac{\partial p}{\partial z} - g \end{aligned} \quad (7)$$

La solución de estas ecuaciones es complicada, pero introduciendo la hipótesis de flujo irrotacional se puede introducir una función escalar de la cual dependen las variables a determinar u , v y w . El problema se reduce a uno de dos variables: Φ y p , dando como resultado dos ecuaciones, la de continuidad y la ecuación de Bernoulli.

$$\nabla^2 \Phi = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial z^2} = 0 \quad (8)$$

$$-\frac{\partial \Phi}{\partial t} + \frac{1}{2} \left[\left(\frac{\partial \Phi}{\partial x} \right)^2 + \left(\frac{\partial \Phi}{\partial y} \right)^2 + \left(\frac{\partial \Phi}{\partial z} \right)^2 \right] + \frac{p}{\rho} + gz = C(t) \quad (9)$$

El problema sigue siendo no-lineal. La hipótesis de linealidad simplifica aun más el problema y permite llegar a una formulación analítica de la solución del potencial.

$$\Phi(x, z, t) = -\frac{Ag}{w} \frac{\cosh k(h+z)}{\cosh kh} \sin(kx - wt) \quad (10)$$

Sea $\Phi(x, z, t)$ el *potencial de velocidades*, se lo puede expresar de diferentes formas, pero la estructura es siempre la misma, donde se pueden identificar tres partes diferentes: (1) el módulo o magnitud del potencial dado $g \frac{A}{\omega}$ (siendo g la gravedad, A amplitud, ω la frecuencia angular); (2) una función de profundidad $\cosh k(h+z)/\cosh kh$ (donde h es la profundidad o calado y k el numero de onda); (3) una función $(kx - wt)$ que relaciona x con t .

A este potencial le corresponde una superficie libre en $z = 0$ dada por ([3]):

$$\eta(x, t) = \frac{1}{g} \left(\frac{\partial \Phi}{\partial t} \right) = A \cos(\kappa x - \omega t) = \frac{H}{2} \cos(\kappa x - \omega t) \quad (11)$$

Esta ultima expresión corresponde a una onda que se propaga con una celeridad $C = \omega/\kappa = L/T$ en el sentido positivo del eje x . En la *Figura 3* se

puede observar una representación de la elevación de la superficie libre dada por η en función del tiempo t .

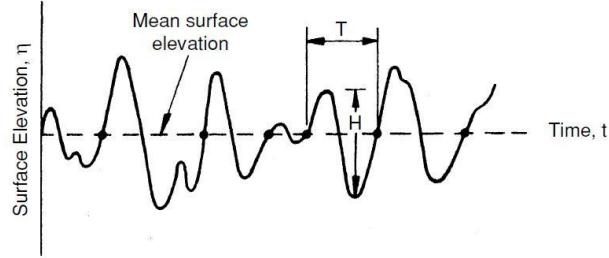


Figura 3: Altura de la superficie libre en función del tiempo (extraída de [4]).

2.2.1. Oleaje irregular

En el mar, un registro de una señal de oleaje, dado por ejemplo por un boyo, es un registro de forma caótica y se denomina Oleaje Irregular (*Figura 4*). El oleaje irregular puede ser considerado como la superposición de un gran numero de componentes correspondientes a ondas con diferentes periodos, alturas y direcciones (*Figura 5*). Este conjunto de ondas forman un *grupo de ondas* el cual se mueve con una velocidad equivalente a la mitad del promedio de las velocidades de sus componentes. Análogamente, la energía espectral asociada al oleaje (explicado en el siguiente capítulo), no se mueve a la velocidad de la ola en si sino a la velocidad del grupo de olas. En la *Figura 6* se muestra como 2 “trenes” de olas (lineas punteadas y finas) interfieren formando un *grupo de ondas* de mayor amplitud (linea gruesa).

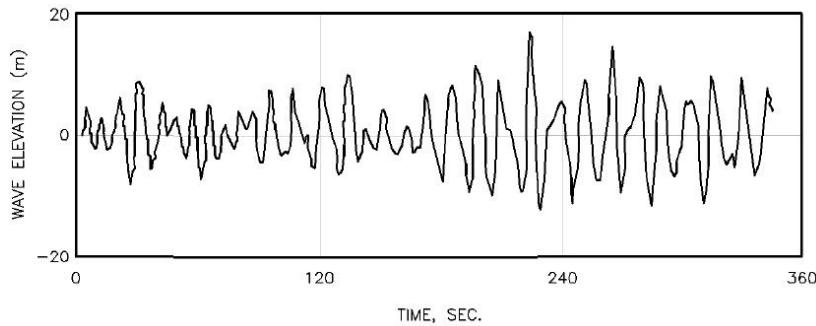


Figura 4: Señal de un oleaje irregular.

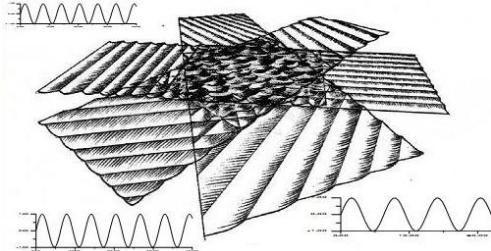


Figura 5: Resultado de la suma (o superposición) de distintos oleajes.

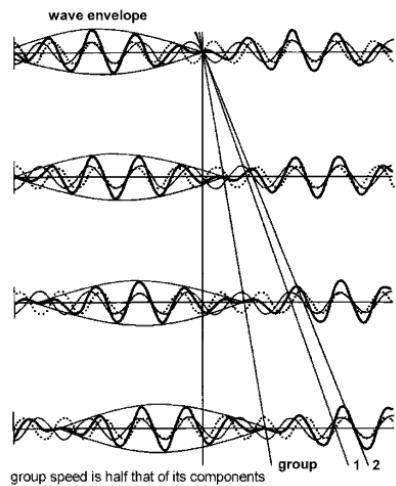


Figura 6: Solapamiento y grupos de trenes de ondas (extraída de [18])

Matemáticamente la representación de la superficie libre de este tipo de oleaje puede expresarse como ([4])

$$\eta(x, y, t) = \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} A_{mn} \cos(\kappa_m \cos \theta_n x + \kappa_m \sin \theta_n y - 2\pi f_m t + \epsilon_{mn}) \quad (12)$$

Donde A_{mn} y ϵ_{mn} representan la amplitud y la fase correspondiente a cada una de las componentes cuyas frecuencias varían en el rango f_m a $f_m + \Delta f_m$ y el ángulo de incidencia en el rango α_m a $\alpha_m + \Delta \alpha_m$. Las variables A_{mn} y ϵ_{mn} son aleatorias y κ_m es el numero de onda asociado a la m -ésima onda.

Esta expresión sirve como base para la definición del *espectro de energía*, el cual se explica en el siguiente capítulo.

2.3. Análisis del oleaje

Genéricamente a los movimientos correspondientes a la superficie del mar, que son ondas de gravedad, generadas por el viento y que se propagan desde el área de generación perdiendo lentamente energía por fricción con la atmósfera, viscosidad molecular, y por la gravedad, (hasta alcanzar las costas) se lo conoce como Oleaje. Una manera simple de modelizar el comportamiento del oleaje es su descomposición en una suma de trenes de ondas regulares (*Figura 6*) de amplitudes, periodos y direcciones variables.

La cantidad máxima de energía que puede existir en cada tren de ondas componentes del oleaje esta regulada por los procesos entre las mismas y los de interacción de rotura. Para el caso de las olas generadas por acción del viento, se las puede clasificar como oleaje totalmente desarrollado (estado de equilibrio logrado bajo la acción de un viento constante sobre una superficie ilimitada) o parcialmente desarrollado (oleaje en crecimiento, sucede cuando el tiempo mínimo o distancia mínima para llegar al estado desarrollado no es alcanzada).

El análisis del oleaje se puede realizar mediante 2 aproximaciones diferentes: a) en el tiempo; b) en frecuencia o análisis espectral.

En este trabajo se utiliza el análisis en frecuencia (b), el cual tiene como objetivo la obtención de la *función de densidad espectral* o *espectro del oleaje* (*Figura 7*), la cual representa la energía total asociada a cada una de los trenes de ondas en que se puede descomponer un registro de oleaje en función de la frecuencia y de la dirección de propagación.

El espectro del oleaje contiene gran cantidad de información que puede ser representada mediante la introducción de una serie de parámetros que sirven para conocer las características principales de dicho espectro, los cuales se mencionaran en el siguiente capítulo.

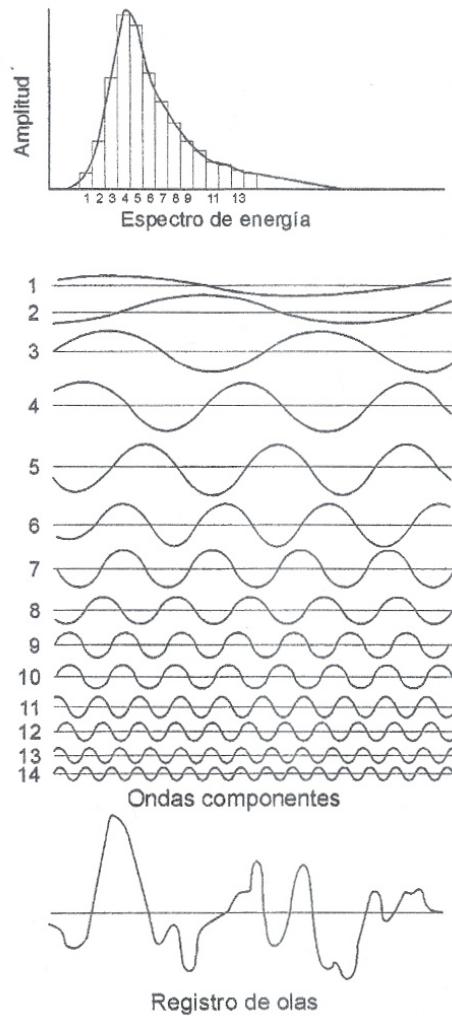


Figura 7: Representación esquemática de las componentes de un registro de olas (extraído de [5]).

2.4. Descripción espectral del oleaje

2.4.1. Método para la obtención del espectro

En nuestro caso, es de interés obtener el espectro dado un registro del parámetro “superficie libre” del oleaje. Este registro del oleaje viene dado por la función $\eta(t)$, donde nos da la altura de la superficie libre dado el tiempo t . Si $\eta(t)$ es una función periódica, con periodo T (lo cual es una hipótesis

bastante fuerte en el caso de una serie de oleaje), puede representarse como una *Serie de Fourier* ([4]):

$$\eta(t) = a_0 + 2 \sum_{k=1}^{\infty} \left(a_k \cos \frac{2\pi k t}{T} + b_k \sin \frac{2\pi k t}{T} \right) \quad (13)$$

donde $k = 1 \dots \infty$ es el numero de armónico considerado, y

$$a_k = \frac{1}{T} \int_0^T \eta(t) \cos \frac{2\pi k t}{T} dt \quad k \geq 0 \quad (14)$$

$$b_k = \frac{1}{T} \int_0^T \eta(t) \sin \frac{2\pi k t}{T} dt \quad k \geq 1 \quad (15)$$

son los *Coeficientes de Fourier*, funciones continuas de la frecuencia y denominadas las *Transformadas de Fourier*. En el análisis del oleaje $a_0 = \text{valor medio}$ se lo considera nulo por comodidad, con lo que la *ecuación 14* pasa a ser valida por $k \geq 1$, por otro lado T corresponde a la duración del registro entero debiendo ser el *m.c.m* de los periodos existentes.

Para realizar un análisis espectral del oleaje se debe realizar el calculo de la *Transformada Discreta de Fourier*, donde los a_k y b_k (de las *ecuaciones 14* y *15* respectivamente) se los puede concentrar en una expresión compleja como:

$$X_k = a_k - i \cdot b_k \quad (16)$$

con lo cual la Transformada de Fourier pasa a tener la forma

$$X_k = \frac{1}{T} \int_0^T \eta(t) e^{-i \frac{2\pi k t}{T}} dt \quad k \geq 0 \quad (17)$$

Si ahora la serie temporal es dada por una muestra discreta de datos uniformemente espaciados:

$$\{x_r\} : r = 0, 1, 2, \dots, (N - 1) \quad (18)$$

con $t = r\Delta$, $\Delta = T/N$, y T periodo *m.c.m* que corresponde con la duración del registro, N tamaño del registro, podemos expresar la *Transformada Discreta de Fourier (DFT)* de la *ecuación 17* como:

$$X_k = \frac{1}{T} \sum_{r=0}^{N-1} x_r e^{-i \frac{2\pi k r \Delta}{T}} \Delta; \quad k \geq 0 \quad o \quad X_k = \frac{1}{N} \sum_{r=0}^{N-1} x_r e^{-i \frac{2\pi k r}{N}}; \quad k \geq 0 \quad (19)$$

Podemos entonces calcular el espectro de energía asociado a un registro de datos por medio de ([4]):

$$S(f) = \frac{A^2}{2 \cdot df} \quad (20)$$

donde

$$A = \frac{2\sqrt{Re(X_k)^2 + Im(X_k)^2}}{N+1} \quad (21)$$

con $a_k = Re(X_k)$, $b_k = Im(X_k)$.

2.4.2. Parámetros espectrales

Existen ciertos parámetros que dan a conocer las características principales del espectro del oleaje. Algunos de estos parámetros aparecen frecuentemente en las funciones de distribución asociadas al oleaje. A continuación mencionaremos los mas relevantes para el caso de estudio presente.

Se define *momento de orden n* de la función de densidad espectral como ([4]):

$$m_n = \int_0^\infty \omega^n S(\omega) d\omega; \quad n = 0, 1, 2, \dots \quad (22)$$

Otro parámetro de utilidad es

$$H_s = 4,004\eta_{rms} = 4,004\sqrt{m_0} \quad (23)$$

donde H_s es la altura de ola significante, el cual es el parámetro más importante en ingeniería marítima a la hora de clasificar un estado de mar (junto con el período de pico $T_p = 1/f_p$ donde f_p es la frecuencia asociada al pico de máxima energía del espectro). H_s se define como la altura media de las $N/3$ mayores olas de un registro de superficie libre compuesto de N olas; m_0 representa el momento de orden cero y es la integral del espectro. Por lo tanto, H_s se puede obtener simplemente calculando el espectro de la señal.

Estos parámetros serán utilizados en los siguientes puntos.

2.5. Propiedades espectrales del oleaje: espectro de JONSWAP

La energía espectral alcanza su máximo en la *frecuencia de pico*, $\omega = \omega_p$ y decrece tanto hacia las altas como hacia las bajas frecuencias (*Figura 8*).

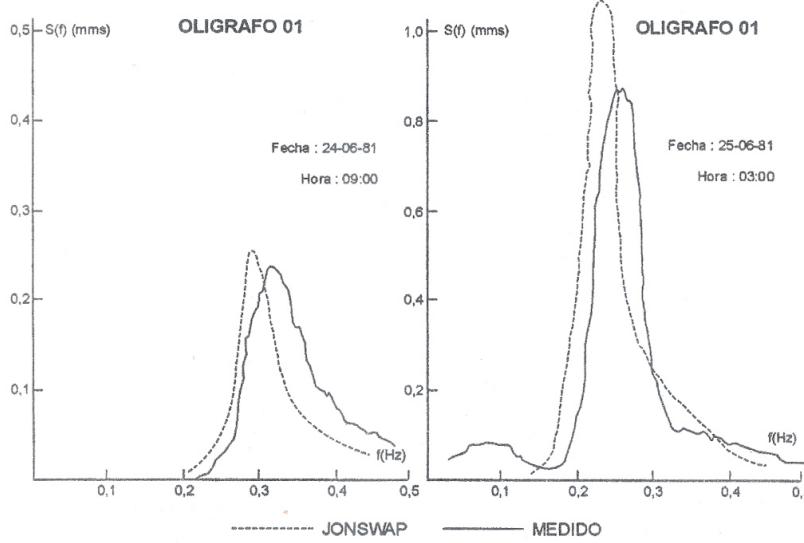


Figura 8: Espectro de JONSWAP calculados y medidos (extraído de [5])

En ingeniería marítima se suele representar el contenido energético de la señal de oleaje mediante el espectro perimétrico de *JONSWAP* (JOint North Sea WAve Project). Este espectro esta basado en una extensa campaña de medidas llevada a cabo en el Mar del Norte entre 1968 y 1969, el cual recibió una aprobación casi inmediata por parte de la comunidad científica una vez publicado, y es quizás el mas utilizado. La forma espectral propuesta es ([4]):

$$S(f) = \alpha H_s^2 T_p^{-2} f^{-5} e^{-1,25(T_p f)^{-4}} \gamma e^{-\frac{(T_p f - 1)^2}{2\sigma^2}} \quad (24)$$

con

$$\alpha \approx \frac{0,0624}{0,230 + 0,0336\gamma - 0,185(1,9 + \gamma)^{-1}}$$

$$\sigma = \begin{cases} \sigma_a; & f \leq f_p \\ \sigma_b; & f > f_p \end{cases} \quad \text{con } \sigma_a = 0,07, \sigma_b = 0,09$$

$$\gamma = 1 \text{ a } 7 \text{ (media de 3,3)}$$

Como se observa, el espectro de JONSWAP depende de los parámetros H_s y T_p , los cuales son objeto de análisis en este trabajo como se lo detalla en el siguiente capítulo.

2.6. Aplicaciones en laboratorio

Todo el marco teórico anteriormente mencionado se aplicó en este trabajo para la detección de la altura de ola en laboratorio y su posterior análisis espectral.

Los oleajes simulados en el canal del laboratorio se generan por medio de una pala mecánica la cual toma como parámetros valores de H_s y T_p para formar una serie temporal de oleaje irregular con el espectro de JONSWAP $S(f)$ asociado a la *ecuación 24* (utilizando $\gamma = 3,3$ constante).

En la *Figura 9* se puede observar por un lado el espectro “teórico” generado por la pala mecánica ($S(f)$) por medio de los parámetros introducidos por computadora (H_s y T_p), y por el otro lado el sensor, el cual mide la altura de ola $\eta(t)$ a lo largo del tiempo. En la parte derecha de la figura se observa el espectro de JONSWAP $S'(f)$ calculado por el sensor (en base a los datos obtenidos $\eta(t)$), que en general difiere al espectro “teórico” debido a varios factores como el caso del desnivel, fricción de la ola con los laterales del canal, etc. Una vez calculado el espectro, se pueden obtener los parámetros H_s y T_s , que son el objeto de estudio a la hora de clasificar el estado del oleaje.

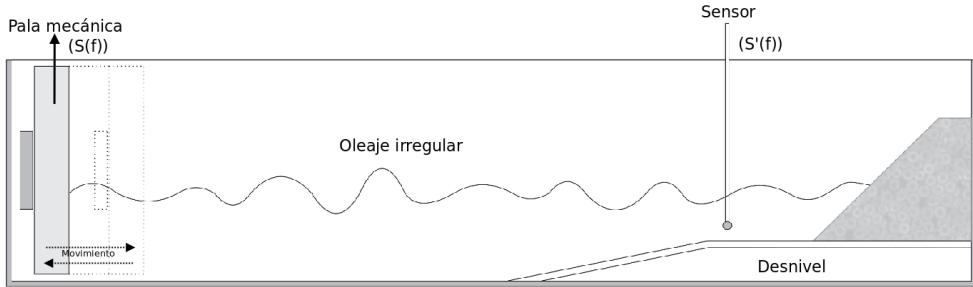


Figura 9: Representación esquemática de la generación del oleaje irregular y medición del espectro de JONSWAP en el canal de laboratorio.

En el *capítulo 6.1. Canal de laboratorio* se detallaran las características

del canal, la pala mecánica y los instrumentos utilizados para realizar las mediciones.

3. Detección de la superficie libre en laboratorio

3.1. Tipo de sensores

Existen una gran cantidad de instrumentos utilizados para medir la altura de la superficie libre del oleaje en laboratorio, cada uno con ventajas y desventajas asociadas, donde todos requieren de una buena experiencia por parte del usuario para su correcta utilización y en general sus costos son elevados. Estos se los pueden clasificar como intrusivos o no intrusivos.

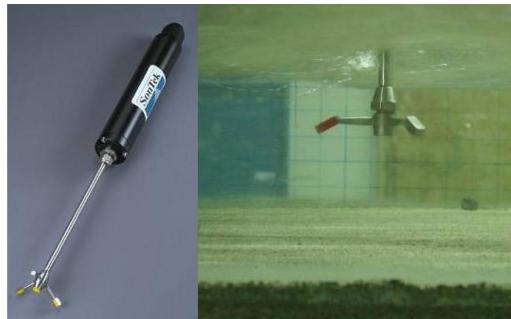
3.1.1. Instrumentos intrusivos

Son aquellos que se introducen dentro de la zona que contiene el fluido a medir, en contacto con el mismo. Éstos tienen como desventaja que con el paso del tiempo, al estar en contacto con el medio, su durabilidad es menor, y en algunos casos limitan los tipos de fluidos que se pueden medir (por ejemplo, instrumentos se estropean en agua salada). Además, estos sensores son específicos para la medición de variables en un cierto rango de escala espacio-temporal, generalmente no pueden medir los procesos cerca de los contornos, dependen de la siembra de partículas en el fluido, entre otras.

Por otra parte, estos instrumentos pueden brindar una alta frecuencia de muestreos con alta precisión.

Como ejemplo, podemos mencionar el *ADV (Acoustic Doppler Velocimeter, Figura 10a)* que mide velocidades en un volumen de fluido al igual que el *Tubo Pitot⁴* (*Figura 10b*), o el sensor de nivel resistivo (*Figura 10c*) para medir niveles.

⁴http://es.wikipedia.org/wiki/Tubo_Pitot



(a) ADV (Acoustic Doppler Velocimeter).



(b) Tubo Pitot



(c) Sensor de nivel resistivo.

Figura 10: Instrumentos intrusivos.

3.1.2. Instrumentos no intrusivos

Son aquellos que no necesitan estar sumergidos ni en contacto con el fluido. Algunos de estos instrumentos (como es el caso del *PIV - Particle*

Image Velocimeter - (*Figura 11*), *LDV* - *Laser Doppler Velocimetry*⁵ -), funcionan por medio de sensores muy delicados como el láser, que en general, sin las precauciones adecuadas, puede generar daños a los seres humanos o dañar el sensor en si. A su vez, dentro de esta rama de instrumentos se encuentra el presentado en este trabajo, el cual utiliza técnicas de procesamiento de imágenes para detectar la superficie libre.



Figura 11: PIV (Particle Image Velocimeter).

3.2. Técnicas basadas en procesamiento de imágenes

A diferencia de los instrumentos tradicionales, los cuales son muy costosos pero que brindan una elevada precisión de la medición que realizan, las técnicas basadas en procesamiento de imágenes (por medio de video cámaras) brindan una opción alternativa de bajo costo, que requieren de poca experiencia por parte del usuario, no son peligrosos (ya que no pueden generar ningún daño físico a personas ni al instrumentos en si), tienen una mayor flexibilidad (aplicables a otros tipos de mediciones y a diferentes escalas espacio-temporales de los procesos) y en general, su precisión es limitada por la definición de la cámara web utilizada, lo que a su vez, a mayor precisión, mayor costo. Sin embargo, con una cámara de alta definición estándar puede ser suficiente para la aplicaciones ingenieriles de laboratorio, como lo es el fin de este trabajo (medir la altura de ola).

El conjunto de técnicas investigadas en este trabajo como así también el trabajo desarrollado se basan en la utilización de cámaras que siguen el modelo *pinhole* o *pinhole model* [6], el cual supone que cada punto en el espacio y su correspondiente proyección en la imagen están sobre una linea recta que pasa por el centro óptico de la imagen (*Figura 12*).

⁵http://en.wikipedia.org/wiki/Laser_Doppler_velocimetry

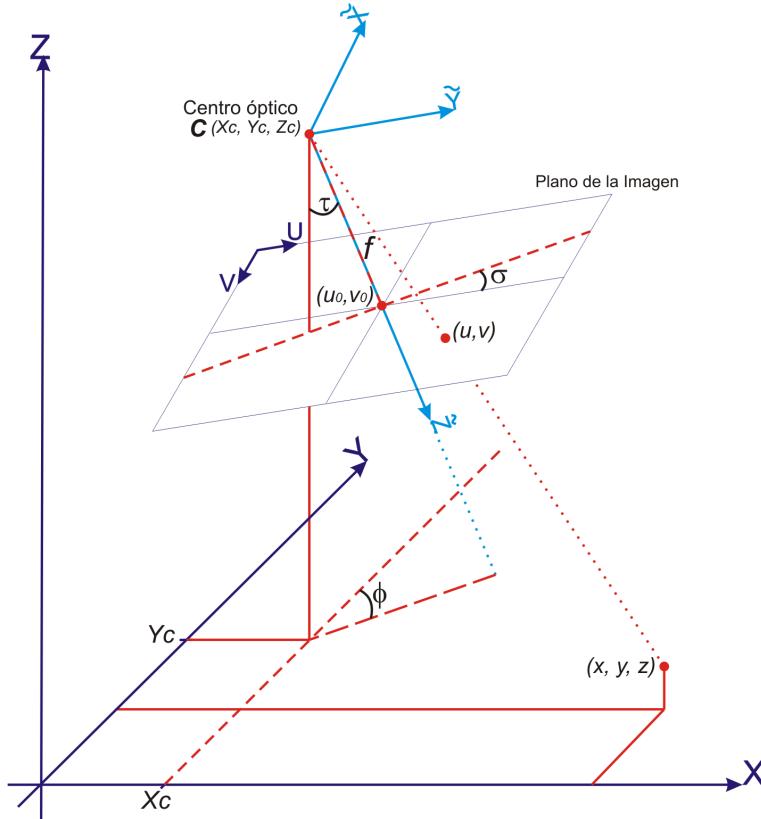


Figura 12: Representación esquemática del modelo *pinhole* (extraída de [6]).

Para lograr obtener resultados adecuados es necesario capturar imágenes con la mayor calidad posible, además de una buena resolución, es importante tener en cuenta varios factores del entorno donde se realizaran las muestras de laboratorio, como por ejemplo, luz ambiental, contraste de la superficie libre con respecto al fluido, suciedades en los vidrios laterales del canal, entre otras. Siempre tendiendo a minimizar la cantidad de ruido (zonas que pueden generar contornos que se confundan con la superficie libre, o áreas que puedan evitar la correcta captura de la misma).

Dentro del conjunto de técnicas basadas en procesamiento de imágenes se pueden diferenciar aquellas que son en tiempo real y las que no, pero ambas, a grandes rasgos siguen un mismo proceso. Este proceso, que tiene como entrada una imagen y como salida la detección de alguna variable determinada (altura de la superficie libre) podemos describirlo, de forma

general, como un conjunto de pasos (secuenciales) o sub-procesos:

- Rectificación.
- Detección de contornos.
- Análisis y procesamiento.

Cada uno de estos pasos pueden ser llevados a cabo de distintas maneras y utilizando diferentes algoritmos o modelos matemáticos.

Algunas de las técnicas analizadas en este trabajo se pueden encontrar en [7], [8], [11].

A continuación se explican en mayor detalle cada uno de los pasos mencionados anteriormente.

3.2.1. Rectificación

Para lograr una correcta medición de la altura de ola (u otra variable hidrodinámica de interés) es necesario trabajar sobre una imagen libre (o al menos con la menor cantidad posible) de errores. Podemos diferenciar 2 tipos de errores o problemas: a) aquellos introducidos por el dispositivo de captura (cámara de vídeo); b) aquellos relacionados con la posición y orientación de la cámara.

Con respecto a los primeros (a), también llamados distorsiones, podemos clasificarlos como radiales (relacionado con la forma del lente de la cámara) y tangenciales (provenientes del proceso de ensamblaje de la cámara en su totalidad), entre otros ([6], [13]).

Por el otro lado (b), el problema de mayor importancia es el de la diferencia que existe entre el sistema de coordenadas entre la cámara y el plano sobre el cual se quiere realizar las mediciones. En la *Figura 13* se muestra como el plano de la zona a analizar (en la figura *object plane*, que en este trabajo representaría la pared del canal donde se realiza la captura de imagen para detectar la altura de la ola), se representa dentro de la imagen capturada por el dispositivo (en la figura *image plane*) con cierta perspectiva. Esto es un problema debido a que, como el objeto de estudio es la altura de la superficie libre (aunque se aplique en general a cualquier otra variable a medir), la altura real de la misma que se observa en el canal se encuentra proyectada en la imagen y con una perspectiva diferente a la real, esto es, con un sistema de coordenadas diferentes. Es por esto que se debe encontrar la relación entre estos 2 sistemas de coordenadas (por medio de lo que se denomina “transformación de la perspectiva”, como se explicara mas adelante)

para poder determinar la correspondencia de cada punto de la imagen a su respectivo punto en la pared del canal.

En un principio, los problemas introducidos por (a) son de menor importancia (debido a que son despreciables en las cámaras nuevas, teniendo en cuenta que no se requiere de una altísima precisión) con respecto a los de (b). Además, estos últimos, son causados principalmente por la posición en la que se coloca la cámara frente a la zona que se quiere medir.

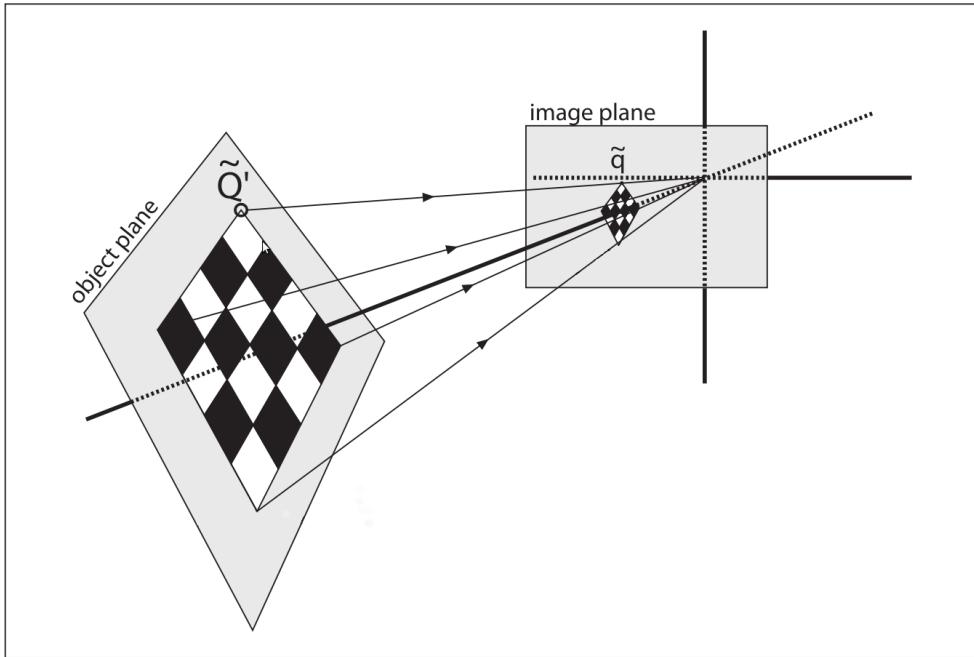


Figura 13: Diferencia de perspectiva entre la imagen en la cámara (image plane) y el plano real (object plane), extraída de [13].

Para llevar a cabo la rectificación, se utilizan diferentes métodos, entre ellos el modelo de Transformada Lineal Directa (DTL) ([6]), solución por mínimos cuadrados ([6]) y métodos alternativos ([9]).

En este trabajo se planteo utilizar un método de simple implementación (y configuración) el cual se basa en transformar la perspectiva de la imagen de entrada alineándola a un sistema de coordenadas cartesianas como se lo detallara en el *capítulo 3.3.1. Rectificador*.

3.2.2. Detección de contornos

Esta etapa tiene como fin detectar los bordes asociados a la superficie libre que se encuentran en la imagen de entrada, minimizando la detección de bordes falsos (aquellos que pertenecen al entorno y no a la superficie libre en si).

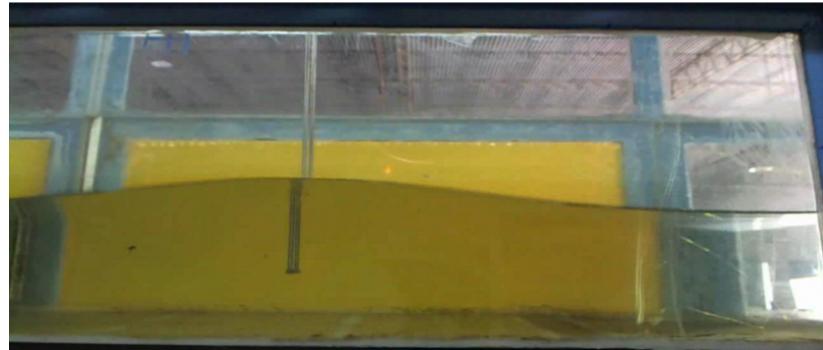
En una primera instancia, se procede a disminuir la complejidad de la imagen reduciendo la cantidad de canales (colores) de la misma, en general, convirtiéndola a una imagen monocromática.

En segundo lugar, se aplican una serie de filtros (o máscaras) con el fin de delimitar la zona que se quiere analizar, evitando procesar zonas que no son de interés y a su vez, eliminando posibles errores causados por falsos valores (esto es, los bordes falsos mencionados anteriormente).

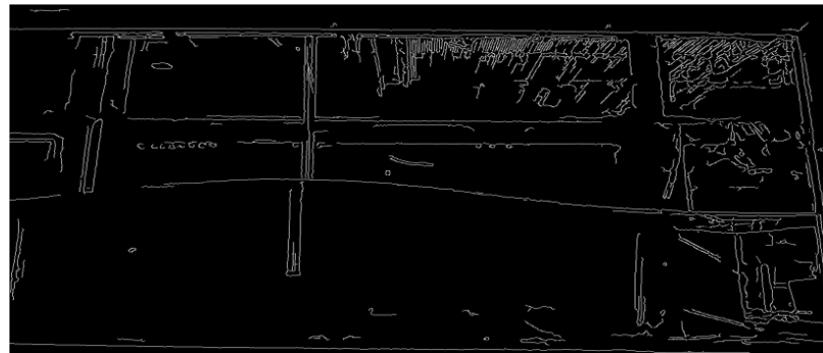
Por último se realiza la detección de contornos sobre la imagen filtrada por medio de algún algoritmo. Existen varios métodos utilizados para detectar los bordes, alguno de estos se pueden ver en [10]. En este trabajo se utilizó el algoritmo denominado *Canny Edge Detector*⁶ (o simplemente *Canny*), el cual es uno de los algoritmos más utilizados y que da mejores resultados da a la hora de encontrar bordes en una imagen, éste se explicará en mayor detalle en el *capítulo 4.4. Canny edge detector*.

A modo de ejemplo, observemos la *Figura 14a* representando a la imagen de entrada de este sub-proceso (ya rectificada) pero sin ningún filtro aplicado, mientras que la *Figura 14b* muestra la imagen de salida (de esta etapa) luego de haberla procesado con el algoritmo de detección de contornos (cabe destacar que en este caso no se delimitó ninguna zona analizar si no que se aplicó el algoritmo de detección de contornos a toda la imagen de entrada).

⁶http://en.wikipedia.org/wiki/Canny_edge_detector



(a) Imagen rectificada sin filtros (imagen de entrada).



(b) Imagen rectificada luego de aplicar el algoritmo de detección de contornos (imagen de salida).

Figura 14: Resultado del proceso de detección de contornos.

Como se puede ver, dependiendo del entorno de trabajo (luz, contraste de la superficie libre con respecto al canal, etc), el algoritmo utilizado, y los filtros aplicados, la existencia de bordes falsos puede generar mucho ruido e introducir errores a la hora de medir la altura de ola. Por esto es necesario una etapa posterior que permita discernir los bordes que pertenecen a la superficie libre, la cual se menciona a continuación.

3.2.3. Análisis y procesamiento

Una vez detectado los bordes de la imagen y en base a los algoritmos utilizados en la etapa anterior, se aplica cierta heurística para diferenciar los bordes correspondientes a la altura de la superficie libre (*bordes activos*), de aquellos que no corresponden a la variable que se quiere analizar (lo que se denomina ruido o *bordes pasivos*) como claramente se puede observar en la

Figura 14b anterior. La forma en la que se detectan estos *bordes activos* no es única y esta fuertemente ligada en como fue modificada la imagen en la etapa anterior y/o en la forma de representar los contornos detectados. A modo de ejemplo podemos mencionar distintos métodos utilizados, como el caso de [7] que utiliza una función de interpolación y filtrado de contornos para eliminar puntos (o bordes) falsos. Para este trabajo se desarrollo un algoritmo simple con tal finalidad, el cual se describe en el *capítulo 4.5. Wave height analyzer*.

Luego de lograr la detección de los *bordes activos* se procede a obtener la altura en metros (o alguna otra unidad de longitud) por medio de la transformación de la relación píxeles/metros (o la unidad de longitud correspondiente).

Podemos decir, en base a los trabajos analizados, que para esta etapa (Análisis y procesamiento) no existe algún lineamiento específico de como llevarla a cabo, si no que es la que mas varia (de trabajo en trabajo) y esta sumamente relacionada con la forma en que fueron realizadas las etapas anteriores.

3.3. Técnicas utilizadas

En el presente trabajo podemos dividir el proceso general en cuatro subprocesos como lo muestra en el diagrama de la *Figura 15*.

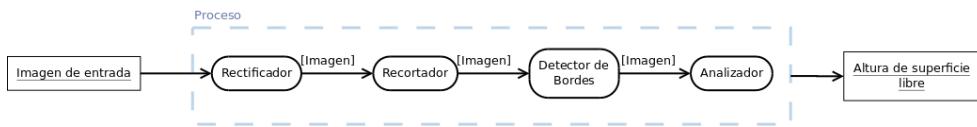
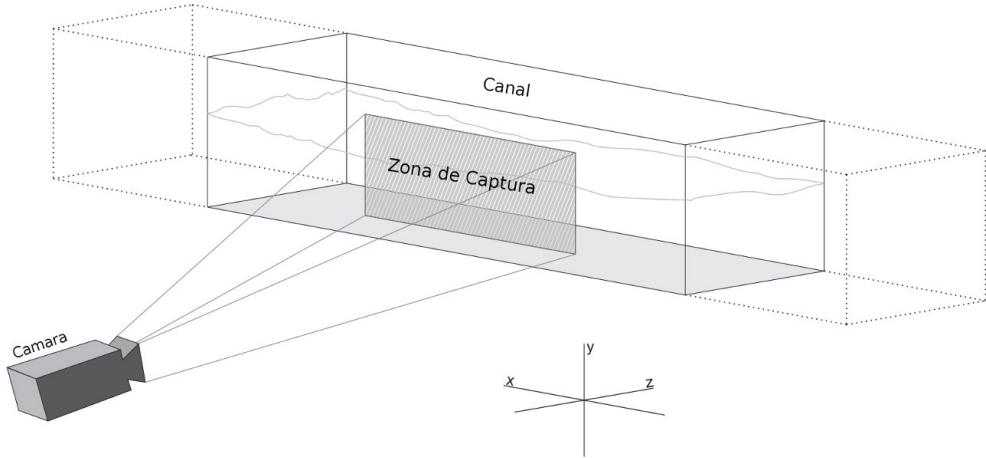
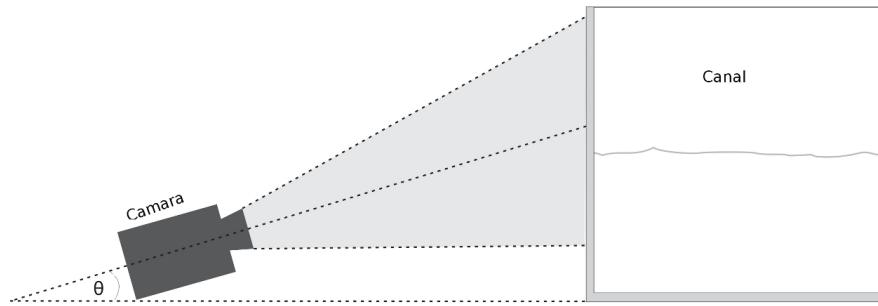


Figura 15: Diagrama general del proceso de la técnica utilizada.

Como se puede ver, el proceso (rectángulo azul) tiene como entrada una imagen que puede provenir de una cámara de video o de un archivo. Dentro del proceso general se pueden diferenciar los cuatro subprocesos que trabajan de manera secuencial, donde cada uno de estos tiene como entrada la imagen procesada por el subproceso inmediatamente anterior. Antes de continuar a explicar cada uno de los subprocesos realizaremos las siguientes definiciones para una mayor comprensión de lo que se desarrolle.



(a) Esquema 3D del canal y zona de captura.



(b) Esquema 2D del canal y ángulo de captura.

Figura 16: Esquema del canal y la zona de interés a analizar.

Basándonos en el esquema presentado en la *Figura 16* definiremos:

- *plano de captura*: como el plano formado por los ejes x , y .
- *zona de captura*: al rectángulo capturado por la cámara sobre la pared del canal que es paralelo al *plano de captura*.
- *ángulo de captura*: El ángulo de rotación de la cámara con respecto al *plano de captura* (en el diagrama de la *Figura 16b*, el ángulo θ).

Podemos observar que no se realizó distinción del ángulo de la cámara sobre el eje y con respecto a la zona de captura, el cual llamaremos ángulo β . Esto debido a que como se verá más adelante, no tiene la misma importancia que el ángulo de captura θ .

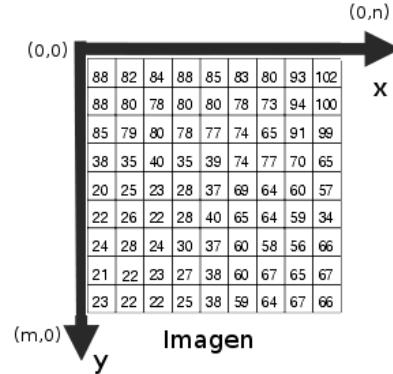


Figura 17: Matriz de píxeles representando una imagen donde cada píxel contiene un valor numérico indicando su color (o escala de grises).

Definiremos además a una imagen I_{mn} como una matriz de valores numéricos enteros tal que $a_{ij} \in I_{mn}$ es el píxel (valor numérico) que se encuentra en la i -ésima fila y j -ésima columna (*Figura 17*), tales que $0 \leq i < m$ y $0 \leq j < n$. Definiremos también a I_{*k} a la columna (k -ésima) de la imagen I_{mn} tal que $0 \leq k < n$ y análogamente para las filas de píxeles I_{l*} . Dada una imagen I_{mn} , $a_{00} \in I_{mn}$ corresponde al píxel superior izquierdo de la imagen mientras que el píxel $a_{mn} \in I_{mn}$ al inferior derecho.

A continuación mencionaremos las características principales y finalidad de cada uno de los subprocessos (mostrados en la *Figura 15*) utilizados en este trabajo para capturar la altura de la superficie libre.

3.3.1. Rectificador

Este subprocesso se basa en 2 grupos de 4 puntos (bidimensionales) correspondientes a los puntos de origen y de destino (de la imagen de entrada y salida respectivamente). Los primeros están asociados a la *zona de trabajo* que se quiere analizar (la cual debe ser un rectángulo contenido dentro de la *zona de captura* - *Figura 16a* - con base alineada al eje x). Mientras que el segundo grupo esta relacionado con el tamaño de la imagen capturada por el dispositivo en si (el tamaño del frame/imagen con el que se trabaja). El primer grupo de puntos necesita ser especificado por el usuario mientras que el segundo se asigna de manera automática.

Debido a que la cámara se encuentra con un *ángulo de captura* $\theta \neq 0$ (*Figura 16b*), la *zona de trabajo* se encuentra proyectada en la imagen toma-

da por el dispositivo con cierta perspectiva (como se explico en el *capítulo 3.2.1 Rectificación.*), esto es, el sistema de coordenadas de la imagen en el dispositivo se encuentra rotado $-\theta$ grados sobre el eje x . Por otro lado y como se menciono anteriormente, existe también el ángulo de rotación β , el cual produce el mismo efecto sobre el sistema de coordenadas de la imagen capturada.

Para solucionar esto, y gracias a que la *zona de captura* es paralela al *plano de captura*, se genera una matriz de transformación de perspectiva por medio de los 8 puntos mencionados anteriormente, la cual mapea cada píxel de la imagen origen a su nueva posición en la imagen destino (solventando el desfasaje de los distintos sistemas de coordenadas como se puede observar en la *Figura 13*).

De esta manera se soluciona el error de la perspectiva y al mismo tiempo se eliminan las zonas (a grandes rasgos) que no pertenecen al área de interés. Además de que es una método sencillo y fácil de configurar por parte del usuario.

En el *capítulo 4.2. Perspective transform* se realiza un desarrollo mas profundo sobre el algoritmo que permite realizar esta rectificación.

3.3.2. Recortador

Este proceso realiza un recorte aún mayor sobre la imagen de entrada descartando aquellas zonas que no son de interés, con el fin de agilizar el procesamiento de los posteriores sub-procesos. Se basa en los siguientes parámetros:

- 1) i : Es el numero de columna I_{*i} (columna de píxeles) de la imagen de entrada I_{mn} a la cual se quiere analizar la altura de la superficie libre ($0 \leq i < n$).
- 2) c : Cantidad de columnas a analizar, donde éstas son las I_{*j} con $j = i - \frac{c}{2}, \dots, i + \frac{c}{2}$, $c > 0$.

Dada la imagen de entrada I_{mn} este proceso la transforma a una imagen I'_{mc} (de salida) reduciéndola su numero de columnas a c .

La existencia de este simple proceso se debe a que en este trabajo necesitamos detectar la altura de la superficie libre (y por ende realizar todo el proceso) en tiempo real. Por otro lado, solo es necesario capturar la altura de la misma en un punto determinado (una columna de píxeles, dada por el parámetro i), aunque esta única columna no alcanza debido a que podría no detectarse el contorno de la superficie de forma correcta dando lugar a cálculos erróneos y un sistema muy propenso a fallas (por tal motivo se utiliza el parámetro $c > 0$). Como ultimo, simplifica las configuraciones de los

posteriores subprocessos, evitando tener que configurar en cada uno de estos la zona de análisis, tomando directamente toda la imagen ya recortada.

3.3.3. Detector de bordes

Su objetivo es detectar los bordes, eliminando en lo posible la mayor cantidad de ruido, pero no de discernir entre *bordes activos* y *bordes pasivos*. Podemos distinguir una secuencia de acciones realizadas por este proceso. Estas son:

1. Se convierte la imagen de entrada I_{mn} (en general color) en una imagen en blanco y negro, por medio de un simple filtro de imagen, el cual se aplica a cada $a_{ij} \in I_{mn}$ modificando solo su valor.
2. Se le aplica un segundo filtro denominado *Gaussian Blur*⁷ con el fin de “suavizar” la imagen suprimiendo la mayor cantidad posible de ruido (como puntos aislados debido a partículas sobre la pared del canal) para una mayor diferenciación de los bordes reales. El funcionamiento de este algoritmo se explicara en el *capítulo 4.3 Gaussian Blur*.
3. Por ultimo se corre el algoritmo conocido por el nombre *Canny Edge Detector* (detallado en el *capítulo 4.4*) utilizado para detectar los bordes en la imagen basado en ciertos parámetros previamente configurados por el usuario que serán mencionados posteriormente junto con los detalles del funcionamiento del algoritmo.

Al finalizar esta secuencia de pasos, podemos asumir que la imagen I'_{mn} de salida, cumple lo siguiente:

$$\forall a_{ij} \in I'_{mn} : \begin{cases} a_{ij} = 1 & \text{si } a_{ij} \text{ pertenece a un contorno} \\ a_{ij} = 0 & \text{caso contrario.} \end{cases}$$

Dicho de otra manera, la imagen de salida I'_{mn} pasa a ser una matriz de valores booleanos donde los a_{ij} perteneciente a la misma contienen el valor verdadero si y solo si forman parte de un contorno o falso en el caso contrario.

Este subprocesso junto con la captura de imágenes desde el dispositivo son los que ocupan casi la totalidad del tiempo de procesamiento.

En este trabajo se opto por la utilización de este algoritmo (*Canny*) debido a que es uno de los mas utilizados en el área de “*computer vision*” y además la librería con la que se trabajo proveía de una implementación

⁷ http://en.wikipedia.org/wiki/Gaussian_blur

del mismo. Aún así, al ser el proceso general una secuencia de subprocesos independientes (cada uno con respecto a los otros), cambiar de algoritmo solo debería implicar la modificación de este único subproceso (o modulo).

3.3.4. Analizador

Dada la imagen I_{mn} (de entrada) preprocessada por el algoritmo de detección de contornos, este proceso tiene como fin decidir cuales de estos $a_{ij} \in I_{mn}$ marcados como verdaderos (recordando que son verdaderos si y solo si pertenecen a un contorno detectado) son *bordes activos* y cuales son *bordes pasivos*. Para lograrlo se basa en un simple algoritmo implementado en este trabajo el cual se explica en el *capítulo 4.5 Wave height analyzer*. y una serie de parámetros que sirven de base como configuración inicial (suministrados por el usuario).

Una vez detectado los *bordes activos*, se realiza una conversión de unidades, pasando de “altura de superficie libre en píxeles” a “altura de superficie libre en metros”, por medio de uno de los parámetros de configuración que establece la relación de equivalencia entre píxeles y metros.

A diferencia de las otras etapas, ésta tiene como salida un valor numérico (y no una imagen), que es la salida del proceso completo, esto es, la altura de la superficie libre detectada y convertida en metros.

3.3.5. Proceso completo

En la *Figura 18* se muestran las imágenes resultantes de cada uno de los procesos mencionados anteriormente (diagramados en la *Figura 15*).

En una primera instancia, tenemos la imagen de entrada sin ningún tipo de transformación (obtenida directamente de la cámara de video o un archivo), ésta es procesada por el *Rectificador* dando lugar a la segunda imagen, la cual solo contiene la zona seleccionada por el usuario (el rectángulo de trabajo mencionado en el *capítulo 3.3.1. Rectificador*). Ésta imagen resultante es la entrada del proceso denominado *Recortador*, el cual descarta fragmentos de la misma generando una con menor cantidad de columnas (en este caso en particular se uso el parámetro c anteriormente mencionado con un valor de 201 con el fin de poder obtener una imagen de mayor tamaño como resultado, lo que no es práctico y en general se utilizan valores entre $5 \leq c \leq 15$). El siguiente proceso involucrado es el *Detector de bordes*, encargado de localizar los contornos en la imagen de entrada (por medio del algoritmo *Canny*) dando lugar a una nueva imagen en la que solo se aprecian bordes. Se puede observar que existe gran cantidad de ruido (*bordes falsos*)

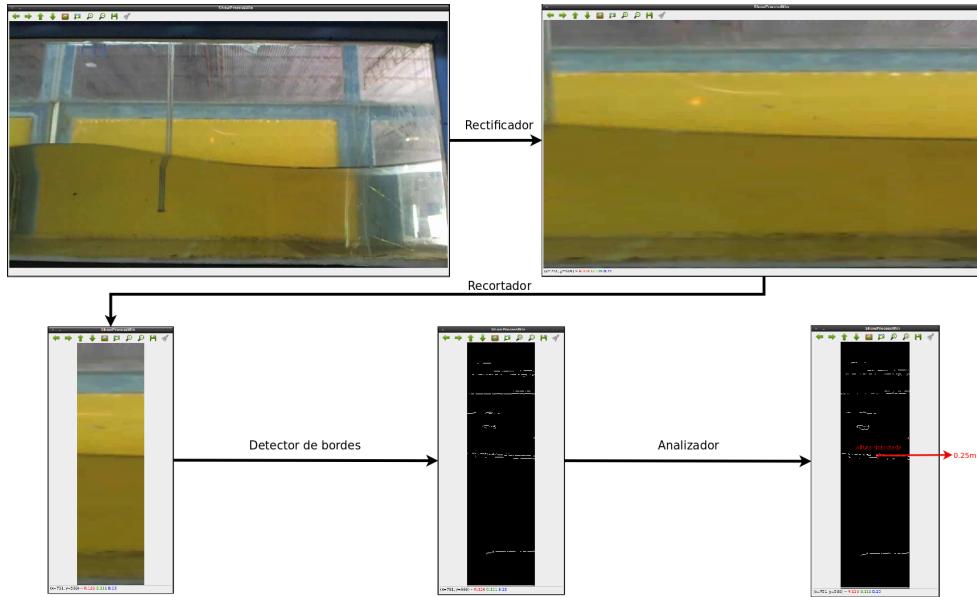


Figura 18: Imágenes resultantes en cada etapa del proceso de la técnica utilizada.

por encima y por debajo de la superficie libre, lo que varia mucho dependiendo de varios factores como la luz utilizada, suciedad en las paredes del canal, parámetros utilizados en el algoritmo de *Canny* (umbrales de “sensibilidad” a la detección de contornos), entre otros. Por ultimo se encuentra el *Analizador*, el cual tiene como finalidad detectar la altura de ola en la imagen de contornos procesada anteriormente. Como mencionamos en el capitulo anterior, este proceso tiene como salida un valor numérico el cual se lo representa por medio de la ultima imagen en la que se resalta (en rojo) la altura de ola detectada.

4. Algoritmos involucrados

4.1. Introducción

En esta sección describiremos las bases matemáticas de los algoritmos utilizados en este trabajo a lo largo del proceso de la detección de la superficie libre. Se mencionaran en el orden en el que son aplicados sobre la imagen de entrada I_{mn} del proceso, proveniente ya sea del dispositivo (webcam) o de un archivo de vídeo.

Los algoritmos utilizados para la rectificación de la imagen y el análisis de los contornos (*capítulos 4.2 Perspective transform.* y *4.5 Wave height analyzer* respectivamente) no poseen nombres concretos por lo que se decidió asignarles uno con el fin de facilitar su identificación en los capítulos siguientes. En particular, este ultimo (*Wave height analyzer*) fue completamente desarrollado para este trabajo.

4.2. Perspective transform

Este algoritmo es utilizado para la rectificación de la imagen al comienzo del proceso.

Se basa en una transformación de perspectiva (por medio de una matriz de transformación T [14]) sobre la imagen de entrada I_{mn} dando como resultado una imagen I'_{mn} donde cada píxel $a'_{i',j'} \in I'_{mn}$ puede ser expresado en términos del punto original $a_{ij} \in I_{mn}$, utilizando coordenadas homogéneas [12] de forma tal que las coordenadas origen $(i, j, 1)$ pasen a ser las coordenadas destinos (i', j', w) para algún w :

$$\begin{cases} i' = \frac{m_{00}i + m_{01}j + m_{02}}{m_{20}i + m_{21}j + m_{22}} \\ j' = \frac{m_{10}i + m_{11}j + m_{12}}{m_{20}i + m_{21}j + m_{22}} \end{cases}$$

donde

$$T = \begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{pmatrix}$$

es la matriz de transformación y con $m_{22} = 1$, entonces podemos expresar

$$\begin{pmatrix} i' \\ j' \\ w \end{pmatrix} = \begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{pmatrix} \begin{pmatrix} i \\ j \\ 1 \end{pmatrix} = \begin{pmatrix} m_{00}i + m_{01}j + m_{02} \\ m_{10}i + m_{11}j + m_{12} \\ m_{20}i + m_{21}j + m_{22} \end{pmatrix} \quad (25)$$

y cada uno de los m_{ab} se calculan en base a los 8 puntos bidimensionales de configuración, 4 de origen (llamémoslos $p_{o_k} : k = 1, \dots, 4$), con sus 4 puntos de destino correspondientes ($p_{d_k} : k = 1, \dots, 4$) resolviendo el siguiente sistema lineal de ecuaciones

$$\begin{pmatrix} p_{o_1}.x & p_{o_1}.y & 1 & 0 & 0 & 0 & -p_{o_1}.x * p_{d_1}.x & -p_{o_1}.y * p_{d_1}.x \\ p_{o_2}.x & p_{o_2}.y & 1 & 0 & 0 & 0 & -p_{o_2}.x * p_{d_2}.x & -p_{o_2}.y * p_{d_2}.x \\ p_{o_3}.x & p_{o_3}.y & 1 & 0 & 0 & 0 & -p_{o_3}.x * p_{d_3}.x & -p_{o_3}.y * p_{d_3}.x \\ p_{o_4}.x & p_{o_4}.y & 1 & 0 & 0 & 0 & -p_{o_4}.x * p_{d_4}.x & -p_{o_4}.y * p_{d_4}.x \\ 0 & 0 & 0 & p_{o_1}.x & p_{o_1}.y & 1 & -p_{o_1}.x * p_{d_1}.y & -p_{o_1}.y * p_{d_1}.y \\ 0 & 0 & 0 & p_{o_2}.x & p_{o_2}.y & 1 & -p_{o_2}.x * p_{d_2}.y & -p_{o_2}.y * p_{d_2}.y \\ 0 & 0 & 0 & p_{o_3}.x & p_{o_3}.y & 1 & -p_{o_3}.x * p_{d_3}.y & -p_{o_3}.y * p_{d_3}.y \\ 0 & 0 & 0 & p_{o_4}.x & p_{o_4}.y & 1 & -p_{o_4}.x * p_{d_4}.y & -p_{o_4}.y * p_{d_4}.y \end{pmatrix} * \begin{pmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{20} \\ m_{21} \end{pmatrix} = \begin{pmatrix} p_{d_1}.x \\ p_{d_2}.x \\ p_{d_3}.x \\ p_{d_4}.x \\ p_{d_1}.y \\ p_{d_2}.y \\ p_{d_3}.y \\ p_{d_4}.y \end{pmatrix}$$

Con todo esto, el algoritmo simplemente se puede describir como sigue:

1. Establecidos los 8 puntos de configuración ($p_{o_k}, p_{d_k} : k = 1, \dots, 4$), se construye la matriz de transformación de perspectiva T .
2. Para cada píxel $a_{ij} \in I_{mn}$ se obtiene su nueva posición en la imagen de salida I'_{mn} utilizando la matriz T como se muestra en la *ecuación 25* copiándose el valor del píxel ($a'_{i'j'} = a_{ij}$).

Notar que asumimos que los valores i', j' obtenidos luego de multiplicar la matriz de transformación sobre el punto de origen (valores i, j), caen dentro de una imagen de igual tamaño, esto es $0 \leq i, i' < m$ y $0 \leq j, j' < n$, en caso de no cumplirse esta condición, los píxeles son automáticamente descartados.

4.3. Gaussian Blur

Este algoritmo, también conocido como *Gaussian smoothing* [15], tiene como finalidad disminuir la amplitud de las variaciones de la imagen (ruidos), especialmente utilizado en la fase previa a los algoritmos que detectan bordes o contornos.

Para comprender mejor el funcionamiento del algoritmo expliquemos algunos conceptos utilizados:

- *Kernel* o *Mask*: Usualmente una matriz de números (a derecha en *Figura 19*) usada en lo que se denomina “*image convolution*”.
- *Convolution*: Es una simple operación matemática⁸ que provee una forma de multiplicar entre si 2 arrays de números, generalmente de diferentes tamaños pero de igual tipo numérico produciendo un tercer array

⁸<http://en.wikipedia.org/wiki/Convolution>

(resultado) de números del mismo tipo. En éste caso (procesamiento de imágenes), los arrays numéricos multiplicados son por un lado la imagen I_{mn} (que puede verse como un array numérico bidimensional, a izquierda en la *Figura 19*) y por el otro lado la matriz *kernel*, dando como resultado, la imagen con el suavizado aplicado.

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}	I_{58}	I_{59}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}	I_{68}	I_{69}

K_{11}	K_{12}	K_{13}
K_{21}	K_{22}	K_{23}

Figura 19: Representación de una imagen (a izquierda) y un *kernel* (a derecha). Extraída de [15].

La *convolución* es realizada deslizando el *kernel* (empezando generalmente por arriba a la izquierda) sobre todos los píxeles de la imagen de forma tal que la matriz *kernel* entre completamente dentro de los límites de la misma, devolviendo en cada posición (que se desliza el *kernel*) un único valor correspondiente. A modo de ejemplo, en el caso del ultimo píxel (abajo a la derecha) el resultado de la operación sería $O_{57} = I_{57}K_{11} + I_{58}K_{12} + I_{59}K_{13} + I_{67}K_{21} + I_{68}K_{22} + I_{69}K_{23}$, donde O_{57} es el resultado de esta operación. Notemos que si la imagen de entrada tiene M filas y N columnas, y el kernel tiene m filas y n columnas, entonces el tamaño de la imagen de salida tendrá $M - m + 1$ filas y $N - n + 1$ columnas.

Distribución gaussiana 1D:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \quad (26)$$

Distribución gaussiana 2D:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (27)$$

donde σ es la desviación estándar de la distribución gaussiana.

La idea es usar esta distribución 2D (*ecuación 27*) como una función del tipo “punto de propagación”, por medio de la *convolución*. Dicho en otras palabras, armar la matriz kernel utilizando la *distribución gaussiana 2D* para luego aplicar la *convolución* y obtener la imagen resultante.

Por temas de eficiencia en general, y gracias a que la *ecuación 27* puede separarse en las componentes x e y (dándonos como resultado *ecuación 26*), podemos aplicar la *convolución* primero en la dirección x y luego en la dirección y por medio de la *ecuación 26*, obteniendo el mismo resultado que habiendo aplicado *convolución* con la *ecuación 27*.

Este algoritmo da como resultado (en cada corrida de la *convolución*) un promedio ponderado de cada píxel vecino, con la media ponderada mas hacia el valor de los píxeles centrales. Utiliza además ciertos parámetros que determinan el tamaño de la *kernel* y el σ utilizado.

A modo de ejemplo, se muestran dos imágenes en la *Figura 20*, a izquierda la imagen original antes de aplicarse el filtro y a derecha luego de haber aplicado el *Gaussian blur*. Para este ejemplo se utilizo un *kernel* de tamaño 7×7 y un $\sigma = 2$.



Figura 20: Imagen comparativa de antes (a izquierda) y luego (a derecha) de aplicar el filtro *Gaussian Blur* con un tamaño de *kernel* 7×7 y $\sigma = 2$.

4.4. Canny Edge Detector

Este algoritmo fue desarrollado por *John F. Canny*⁹ en 1986, tiene como finalidad detectar los bordes en una imagen, filtrando todo aquello que no pertenezca a un contorno. El algoritmo se lo puede dividir en una serie de pasos:

1. **Smoothing:** Suavizar la imagen para remover el ruido utilizando el filtro gaussiano anteriormente mencionado con parámetro $\sigma = 1,4$ y un *kernel* de tamaño 5×5 .
2. **Finding gradients:** Los bordes deben ser marcados donde los gradientes de la imagen tengan magnitudes grandes por medio de alguno de

⁹ <http://www.eecs.berkeley.edu/Faculty/Homepages/canny.html>

los “operadores de gradiente” (como *Roberts*¹⁰, *Sobel*¹¹ - el cual es el utilizado en este trabajo -, *Prewitt*¹²). Estos operadores devuelven el valor de la derivada primera en la dirección horizontal G_x y vertical G_y , los cuales sirven para determinar la magnitud del gradiente G (o también conocidos como *edge strengths*) y dirección Θ asociados a cada uno de los píxeles de la imagen, donde

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\Theta = \arctan \left(\frac{G_y}{G_x} \right)$$

El ángulo Θ es “redondeado” a una de las 4 posibles “inclinaciones” dadas por los ángulos 0, 45, 90 y 135 (ya que 180 se lo considerara análogo a 0, 135 a 45, 270 a 90 y 315 a 135).

3. **Non-maximum suppression:** Dada la imagen de magnitudes de gradientes calculada en el paso anterior, donde cada píxel contiene el valor $|G|$ y el Θ redondeado, se procede a eliminar todo borde borroso dejando solo aquellos bien marcados (también denominados como *fuertes*), esto se logra por medio de la eliminación de todo máximo que no sea local en la imagen de gradientes y dejando solo aquellos que si lo sean. Se puede describir el procedimiento realizado sobre cada píxel p en 2 pasos y por medio de la *Figura 21* como: a) Se obtienen los píxeles vecinos (p_1 y p_2) asociados a p en base a la dirección Θ (en este caso $\Theta = 90$ por lo tanto se toman tanto el píxel vecino superior p_1 y el píxel vecino inferior p_2). b) Se compara la magnitud del gradiente G de p con el de los vecinos p_1 y p_2 . Si éste es mayor que el de sus vecinos se lo deja marcado, caso contrario se lo suprime.

¹⁰http://en.wikipedia.org/wiki/Roberts_Cross

¹¹http://en.wikipedia.org/wiki/Sobel_operator

¹²http://en.wikipedia.org/wiki/Prewitt_operator

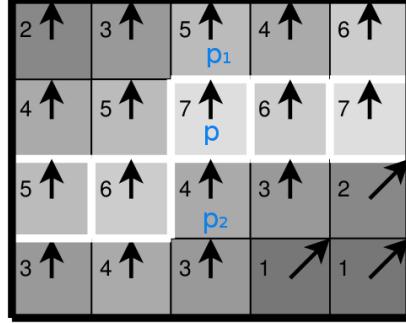


Figura 21: Non-maximum suppression. La magnitud del gradiente (edge strengths) se muestra como números (y escala de gris) al costado de la dirección de G , indicadas como flechas (Θ). Los píxeles pertenecientes a los bordes fuertes se resaltan con un borde blanco. (extraída y modificada de [16])

4. **Double thresholding:** La imagen resultante del paso anterior es procesada comparando cada uno de los píxeles (marcados como pertenecientes a un contorno) con un umbral alto y uno bajo. Aquellos píxeles que estén por debajo del “umbral bajo” son descartados, aquellos que estén por encima del “umbral alto” son marcados como *fuertes*, mientras que aquellos que se encuentren por encima del bajo y por debajo del alto son marcados como *débiles*.
5. **Edge tracking by hysteresis:** Dada la imagen anterior, todos los píxeles marcados como *fuertes* son automáticamente clasificados como contornos reales (y por ende incluidos a la imagen final resultante). Aquellos píxeles marcados como *débiles* son incluidos como contornos reales si y solo si están conectados a alguno de los contornos *fuertes*, caso contrario son descartados.

En [16] se describe en mayor detalle cada uno de los pasos mencionados anteriormente.

A modo de ejemplo podemos observar la *Figura 22* en la que se muestran 2 imágenes luego de ser aplicado el algoritmo de *Canny*, utilizando distintos parámetros de umbrales sobre la misma imagen de entrada (a izquierda en la *Figura 20*). En la primera (*Figura 22a*) podemos observar una gran cantidad de bordes detectados provenientes del ruido mismo de la imagen y debido a que se usaron valores bajos para ambos umbrales (thresholds). En la segunda (*Figura 22b*) se puede ver que la cantidad de contornos es semejante a lo que uno esperaría y prácticamente no se percibe ruido o bordes no deseados. Es

por esto, que es de suma importancia elegir valores adecuados para ambos umbrales a la hora de utilizar el algoritmo de *Canny*.



(a) *Canny* Thresholds: 2 y 15



(b) *Canny* Thresholds: 20 y 50

Figura 22: Imágenes resultantes luego de aplicar el algoritmo *Canny* con distintos parámetros de umbrales (thresholds).

4.5. Wave height analyzer

Este algoritmo tiene como finalidad determinar cual de todos los contornos detectados en la imagen corresponde a la altura de ola que se este analizando. Para lograrlo requiere de la siguiente información.

Sea I_{mn} la imagen de entrada, preprocesada de manera tal que $I_{mn} \ni a_{ij} = 1$ si y solo si a_{ij} pertenece a un contorno detectado (por simplicidad asumamos n impar, caso contrario se trabaja sobre $I_{m(n-1)}$).

Sea h_0 el valor de la altura de ola inicial en píxeles, esto es, el numero de fila en la imagen I_{mn} en la que se encuentra la altura inicial ($0 \leq h_0 < m$).

Sea V una constante numérica predefinida (previamente calculada) por el usuario representando la máxima variación de altura de ola aceptable en píxeles (esto es la cantidad máxima de píxeles aceptables que puede cambiar la ola de un paso al otro).

Se asume además que el movimiento de la ola se realiza sobre el eje y (*Figura 17*), esto es, dada la columna k -ésima (I_{*k}) de la imagen, podemos observar la diferencia de altura a medida que nos movemos verticalmente sobre las diferentes filas de I sobre la columna k .

Teniendo en cuenta que como objetivo principal, este algoritmo debe obtener la altura de ola en un punto determinado de la imagen (en una columna determinada), se asume que este punto se encuentra sobre la columna $\frac{n}{2}$.

Podemos describir entonces el funcionamiento del algoritmo como:

- Para cada columna $I_{*k} : 0 \leq k < n$, sea \bar{w}_k la ultima altura calculada en la k -ésima columna (donde \bar{w}_k es el índice de una fila), buscamos la nueva altura w_k de la siguiente forma: Partiendo de $a_{\bar{w}_k k} \in I_{mn}$, buscamos el i tal que se cumpla

$$a_{ik} = 1 \wedge |i - \bar{w}_k| \leq |j - \bar{w}_k| \forall j = 0..m : a_{jk} = 1$$

asignando entonces $w_k = i$. En caso de que tal i no exista asignamos $w_k = \bar{w}_k$. En palabras, esto no es mas que encontrar el píxel marcado como verdadero que se encuentre mas cercano (en posición) a la ultima altura calculada (\bar{w}_k).

- Sea $k = \frac{n}{2}$ la columna de mayor importancia (la que se esta analizando), se calcula la varianza $v = |w_k - \bar{w}_k|$. Si $v \leq V$ se termina el algoritmo y se devuelve v , caso contrario se continua con el siguiente paso. En palabras, calculamos la distancia entre la altura anterior y la altura actual y verificamos si esta altura es menor que la constante V , de ser así se la devuelve, caso contrario se sigue con 3.

3. Sea $k = \frac{n}{2}$, se calculan las $v_i = |w_{k-i} - \bar{w}_{k-i}| + |w_{k+i} - \bar{w}_{k+i}|$, $i = 1, \dots, \frac{n-1}{2}$ y se elige el i tale que $v_i \leq \min\{v_j\} : j = 1, \dots, \frac{n-1}{2}$. Una vez encontrado este i se calcula la altura promedio entre estas 2 ultimas alturas de las columnas correspondientes, esto es, $h = \frac{w_{k-i} + w_{k+i}}{2}$ y devolvemos h como altura. En caso de $h > V$ se devuelve \bar{w}_k . En palabras, esto no es mas que buscar las alturas correspondientes a los bordes del punto que se esta analizando (que son las columnas de píxeles a los alrededores de la que se analiza) que presente la menor variación, si estas variaciones son mayores que V se devuelve la ultima altura detectada en la columna k .

Cada vez que finaliza el algoritmo se actualizan las ultimas alturas encontradas para evitar acumulación de error.

Las \bar{w}_k de cada una de las columnas son inicializadas por defecto con el valor h_0 (altura inicial de ola).

Puede parecer que el paso 3 agrega complejidad innecesaria, pero debido a que en ciertas situaciones (por problemas de luz o de entorno), la detección de contornos no se realiza correctamente en ciertos puntos (y por ende la altura de ola no es detectada), es necesario una forma alternativa de encontrar la altura de la superficie libre a los alrededores del punto que se esta analizando, evitando de esta forma posibles mediciones erróneas como tomar un contorno marcado no perteneciente a la altura de ola y asumiendo que lo es. De esta forma se logra que el algoritmo sea mas robusto y por ende un sistema menos propenso a fallas.

5. Implementación y funcionamiento del sistema

5.1. Requerimientos principales

Como objetivo principal, el sistema debía capturar la altura de la superficie libre del fluido a medir desde una cámara (web/usb) y lograrlo en tiempo real. A continuación se listan los requerimientos mas importantes que el sistema debía cumplir:

- **REQ1:** Dada una cámara web usb, poder capturar y procesar las imágenes a una velocidad no inferior a 20 fps (o 20 capturas por segundo) y una resolución que permita una detección de altura con no mas de 0.5 cm de error.
- **REQ2:** El sistema debía funcionar en una maquina estándar (que no se requiera de una supercomputadora para lograr la velocidad de procesamiento), ya que, como se menciono anteriormente, una de las ventajas del sistema era su bajo costo.
- **REQ3:** Ser fácil de utilizar y configurar por cualquier persona (que no posea conocimientos previos) que use el sistema.
- **REQ4:** Tener una estructura (arquitectura) flexible que posibilite el desarrollo de futuras aplicaciones (que midan otras variables hidrodinámicas) o ser modificado por otros investigadores en el futuro.
- **REQ5:** Contar con una interfaz gráfica que permitiese mostrar los datos procesados y además la posibilidad de compararlos con los obtenidos por el otro sistema de medición actualmente utilizado (sensor de nivel resistivo).

Debido a que no existía ninguna restricción sobre el lenguaje de programación a utilizar, entorno sobre el cual desarrollar, ni tipos de librerías a emplear, se tomaron ciertas decisiones sobre como sería la implementación del sistema, y teniendo en cuenta los requerimientos antes mencionados, se opto por lo siguiente:

- **Entorno:** *Linux*¹³.
- **Lenguaje de programación:** *C++*¹⁴.

¹³www.linux.com

¹⁴www.cplusplus.com

- **Librerías de procesamiento gráfico:** *OpenCV*¹⁵.
- **Librerías de interfaz gráfica (Graphical User Interface o GUI):** *Qt*¹⁶.
- **Librerías adicionales:** *Tinyxml*¹⁷. *Qwt*¹⁸. *v4l-utils*¹⁹, *FFTW++*²⁰.

En los siguientes párrafos se dará una breve explicación del porque de éstas decisiones.

El sistema fue desarrollado en el entorno Linux principalmente por comodidad y gusto, aún así, las librerías utilizadas son multiplataforma por lo que debería poder ser compilado en otras entornos (exceptuando la aplicación externa utilizada para configurar la cámara web, que en el caso de windows, la aplicación proveída por el manufacturador de la cámara debería alcanzar).

El lenguaje de programación *C++* fue elegido debido a que es un lenguaje de programación con mayor performance en comparación con otros, lo que era necesario para cumplir con el procesamiento en tiempo real (**REQ1**, **REQ2**). Por otro lado ya se poseían conocimientos previos del mismo (como así también preferencia), lo que agilizaría el desarrollo.

Dentro de un gran conjunto de librerías de procesamiento gráfico, se optó por *OpenCV* ya que ésta está principalmente orientada a sistemas que requieren de manipulación de imágenes en tiempo real (*real time computer vision systems*) como la robótica. Además de que tiene un alto grado de maduración, está actualmente activa y existe mucha documentación.

En cuanto al **REQ3**, para poder cumplirlo, necesariamente se requería de una interfaz gráfica para el usuario, por lo cual se decidió elegir *Qt* ya que posee una *API*²¹ para *C++*, cubría con todas las necesidades que el sistema planteaba, era de fácil integración con *OpenCV* y se poseía cierto grado de conocimiento previo sobre la misma.

Con respecto a las librerías adicionales, las de mayor impacto sobre el sistema es *Qwt* ya que **REQ5** se basa en ésta. Fue elegida debido a que proveía la funcionalidad deseada de una manera muy sencilla y de fácil integración con las demás librerías (es una extensión de *Qt* - utilizada para la *GUI* del sistema -). Además *FFTW++* se utilizó para el cálculo y análisis de los datos obtenidos.

¹⁵ <http://opencv.willowgarage.com/wiki/>

¹⁶ <http://qt.nokia.com/>

¹⁷ <http://www.grinninglizard.com/tinyxml/>

¹⁸ <http://qwt.sourceforge.net/>

¹⁹ <http://freecode.com/projects/libv4l>

²⁰ <http://fftwpp.sourceforge.net/>

²¹ *Application Programming Interface*

El código fuente del sistema implementado se lo puede encontrar en [19].

5.2. Arquitectura

5.2.1. A nivel general

A un nivel “macro”, la arquitectura del sistema se la puede simplificar como se lo muestra en la *Figura 23*. Esta decisión fue tomada para poder cubrir el **REQ4** (flexibilidad para la fácil integración de posibles futuras aplicaciones).

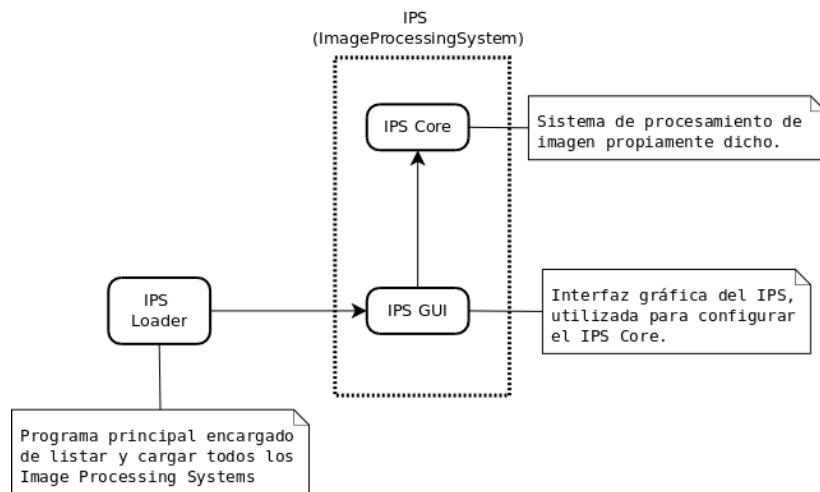


Figura 23: Arquitectura general del sistema.

Como se puede ver en el diagrama de la *Figura 23*, el sistema consta de un modulo principal (*Image Processing System Loader* o *IPS Loader*) el cual tiene como finalidad listar y mostrar todos los módulos *IPS* que se encuentren dando la posibilidad al usuario de elegir y cargar uno específico. Estos *IPS* tienen como objetivo encapsular una aplicación con una finalidad específica, que en el caso de este trabajo, es detectar la altura de la superficie libre de un fluido.

Cada *IPS* esta conformado por 2 partes, una gráfica (*IPS GUI*) y una de procesamiento (*IPS Core*). La primera, tiene como fin brindarle al usuario una forma fácil y ágil de configurar a la segunda, la cual en general, requiere de muchos pasos y parámetros para poder ser utilizada correctamente. En esta ultima (*IPS Core*) se encuentra toda la lógica y algoritmos para el procesamiento de imágenes, que no depende en absoluto de la parte gráfica para su

funcionamiento, lo que permite, si se desea, desarrollar nuevas aplicaciones sin la necesidad de implementar una interfaz gráfica o aún más, modificar de manera sencilla el sistema para soportar otro tipo de librerías para la *GUI*.

Debido a que las aplicaciones que pueden ser desarrolladas requieren en general de una interfaz gráfica, y que además, la configuración se realiza en una fase previa al procesamiento de imágenes (antes de empezar a capturar y procesar las imágenes el *IPS Core* debe estar configurado), se optó por ejecutar la interfaz gráfica en un principio y que esta sea la encargada de ejecutar la otra parte (*IPS Core*). Esto se lo puede ver en el diagrama de la *Figura 23* a través de las flechas (*IPS Loader* → *IPS GUI*, *IPS GUI* → *IPS Core*).

5.2.2. A nivel de IPS

En este trabajo se creó una estructura general (una especie de *Framework*²²) para todos los *Image Processing System (IPS)* que brindará la suficiente flexibilidad para las futuras aplicaciones a desarrollar. Esta estructura se aplica principalmente a la parte de procesamiento (*Core*) ya que la interfaz gráfica puede variar mucho de aplicación en aplicación.

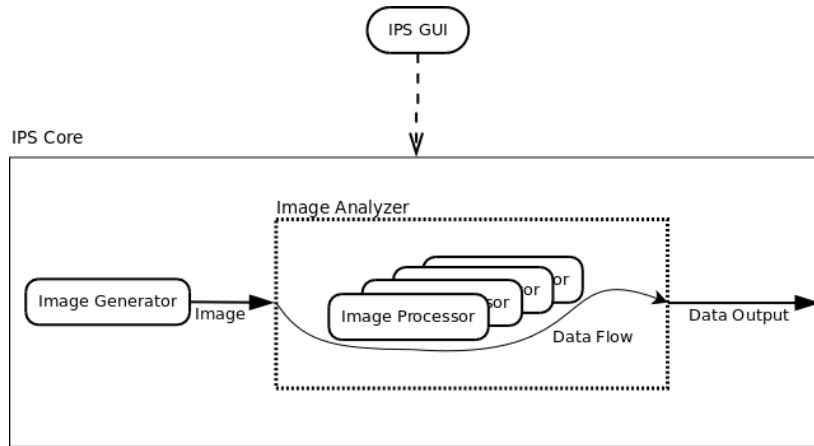


Figura 24: Arquitectura de los *Image Processing System (IPS)*

A grandes rasgos, se puede observar (*Figura 24*) que el *IPS Core* consta de dos módulos, por un lado el *ImageGenerator*, el cual, como su nombre lo indica, tiene como fin generar imágenes para ser procesadas, estas pueden

²²<http://es.wikipedia.org/wiki/Framework>

provenir de una cámara o bien de un archivo de vídeo. Por otro lado se encuentra el *ImageAnalyzer*, el cual se rige de sus unidades básicas funcionales llamadas *ImageProcessor*, las cuales realizan el verdadero procesamiento de imágenes, cada uno con una funcionalidad determinada. Dada una imagen I obtenida del *ImageGenerator*, el *ImageAnalyzer* tiene como finalidad procesarla (a través de los distintos *ImageProcessors* que contiene) y devolver como salida el resultado del procesamiento completo.

En la parte superior del diagrama se puede observar la interfaz gráfica (*IPS GUI*), la cual se comunica con el *IPS Core* con el fin de poder configurar cada uno de estos *ImageProcessors*.

Gracias a esta arquitectura, el sistema es muy flexible en cuanto a la actividad que deba realizar (y por ende la aplicación que se quiera desarrollar), ya que simplemente se basa en lo que los *ImageProcessor* estén configurados para hacer.

A modo de ejemplo, en el caso particular de este trabajo se desarrollo un *IPS* con el fin de capturar la altura de superficie libre de un fluido, al cual se lo llamo **WaveHeightIPS** (*WHIPS*). Para lograr la funcionalidad deseada se desarrollaron cuatro *ImageProcessor* distintos los cuales se muestran en la *Figura 25*.

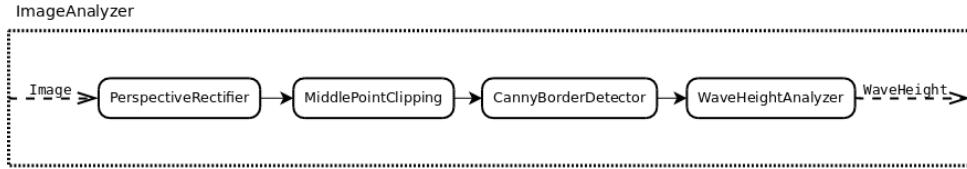


Figura 25: *ImagePorcessor's* del *WaveHeightIPS*.

En los siguientes capítulos se explicara el funcionamiento detallado del *WHIPS* y de cada uno de los *ImageProcessor* que lo conforman.

5.3. Diseño

5.3.1. Preliminares

Comenzaremos describiendo el diseño del sistema al nivel estructural sin entrar en detalles sobre el *IPS* desarrollado para este trabajo, osea, se mostrará en un principio, como es el diseño de la maquinaria (*framework*) que permite la creación de estos *IPS*.

5.3.2. Nivel estructural o de framework

Como se menciono anteriormente, el sistema debía facilitar el desarrollo de futuras aplicaciones (**REQ4**), para lograr esto, se modelo y encapsulo a cada posible aplicación como un *Image Processing System (IPS)*. Esto llevo a un diseño al nivel de *framework* como se muestra en la *Figura 26*.

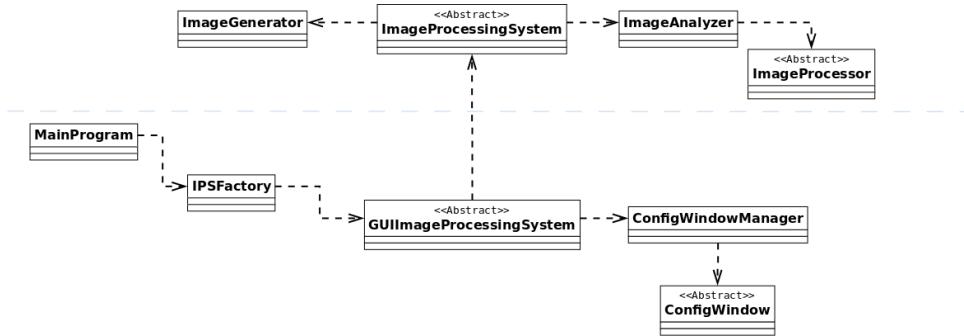


Figura 26: Diseño de alto nivel (al nivel estructural de los *IPS*).

Se observa el diseño a grandes rasgos del sistema completo sin ningún detalle de la interfaz de cada una de las clases, como así también, se obviaron varias clases auxiliares para evitar complejidad en el esquema. Listaremos a continuación, en forma general, las funcionalidades principales de cada una de estas clases.

- **MainProgram**: Programa principal encargado de cargar los *IPS* y ejecutarlos según corresponda.
- **IPSFactory**: Clase auxiliar encargada de buscar y listar todos los *IPS* disponibles.
- **GUIMainImageProcessingSystem**: Clase abstracta que brinda la interfaz de como deben ser los *IPS GUI* a ser implementados. Esta es la clase que se ejecuta en una primera instancia cuando el usuario desea ejecutar un *IPS* determinado.
- **Config WindowManager**: Manejador de ventanas, encargado de organizarlas y mostrarlas según corresponda.
- **ConfigWindow**: Abstracción de como debe ser toda ventana de configuración.

- **ImageGenerator:** Clase encargada de abstraer la generación de imágenes, ya sea de archivo o de una cámara de vídeo.
- **ImageProcessingSystem:** Abstracción de como deben ser los *IPS Core* a ser implementados.
- **ImageAnalyzer:** Clase encargada de contener y ejecutar en orden a los *ImageProcessor* que contenga. Representa el modulo de procesamiento de imágenes.
- **ImageProcessor:** Abstracción de como debe ser la unidad básica de procesamiento.

Como se puede observar (*Figura 26*), y como se menciono en el *capítulo 5.2.1. A nivel general*, se diferencian 2 partes (en la imagen anterior, separadas por la linea discontinua horizontal). En la fracción superior se encuentra el denominado *IPS Core* (donde se realiza el procesamiento real de las imágenes), mientras que en la parte inferior se encuentran el programa principal (conformado por: *MainProgram* y *IPSFactory*) y el *IPS GUI*. El diseño en mayor detalle de cada una de estas partes serán descriptas a continuación.

5.3.3. IPS Core

El núcleo de procesamiento se encuentra en este modulo. A modo simplificado, el diseño de clases del *IPS Core* se puede observar en el diagrama de la *Figura 27*.

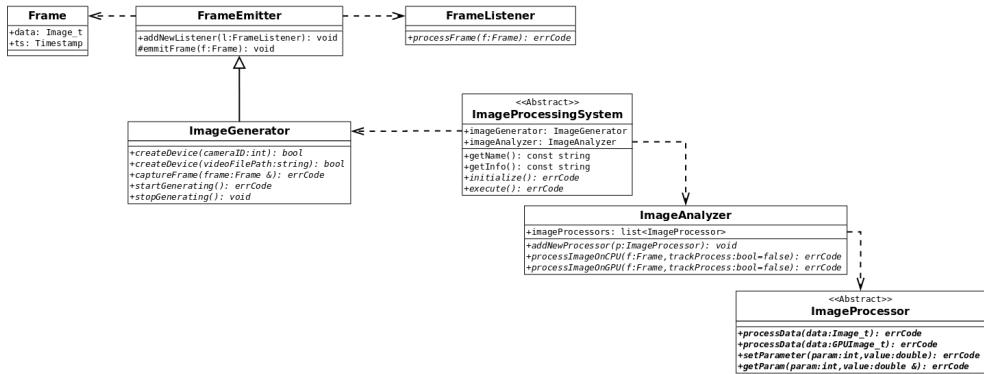


Figura 27: Diseño de clases simplificado de un *IPS Core*.

Describiremos a continuación la funcionalidad de cada una de las clases:

- **Frame:** Es el tipo de dato con la que se maneja el sistema para almacenar la imagen capturada por el dispositivo con el tiempo de captura asociado.
- **FrameListener, FrameEmitter:** Funcionan en conjunto como notificador (*FrameEmitter*) y receptor -procesador- (*FrameListener*) de *Frame*. Esto es lo que se denomina *Observer* en patrones de diseño [17].
- **ImageGenerator:** Clase encargada de abstraer la generación de imágenes. Dependiendo del método llamado el *ImageGenerator* puede ser configurado para obtener imágenes desde una cámara de vídeo (*createDevice(cameraID)*) o bien desde un archivo de vídeo (*createDevice(videoFilePath)*). Además brinda 2 opciones para devolver nuevos *Frame*, por un lado los puede generar automáticamente (si se llama a *startGenerating()*) y distribuyéndolos/notificando a los *FrameListener* asociados (asumiendo que ya fueron asociados por medio del método *addNewListener*), o puede hacerlo a pedido (por medio de la llamada *captureFrame(Frame)*) devolviendo de a un único *Frame*. Se pensó además, en caso de ser necesario, la posibilidad de realizar *buffering*²³ agilizando por medio de la paralelización (usando mas de un core) la captura de frames desde el dispositivo y su posterior procesamiento²⁴.
- **ImageProcessor:** Abstracción de la unidad base funcional, esto es, toda transformación, manipulación y análisis que se haga sobre las imágenes tomadas (*Frame*) se realiza por medio de un *ImageProcessor*. Como se puede ver, existen 4 métodos principales, los cuales deben ser implementados por aquellos que hereden de éste. Estos son: *getParam*, *setParam*, utilizados para obtener y configurar los parámetros del *ImageProcessor*, mientras que *processData(Image_t)*, *processData(GPUImage_t)*, determinan donde se realizará el procesamiento de la imagen (en CPU o en GPU²⁵). La existencia de este último (*processData(GPUImage_t)*) surgió debido a que en un principio se planteo la posibilidad de realizar análisis mas exhaustivos sobre imágenes de mayor tamaño lo que necesariamente requería un poder de procesamiento mucho mayor, pero al mismo tiempo se debía mantener la velocidad, para lograrlo se considero la posibilidad de realizar este trabajo en la GPU.

²³<http://es.wikipedia.org/wiki/Buffer>

²⁴Actualmente no se implemento debido a que a la hora de realizar las pruebas se logro cumplir con la performance requerida.

²⁵http://en.wikipedia.org/wiki/Graphics_processing_unit

- **ImageAnalyzer**: Clase encargada de contener y almacenar ordenadamente a los *ImageProcessor* utilizados para la transformación/análisis del *Frame*. En conjunto con los *ImageProcessor* conforma la unidad principal de procesamiento. Debido a que el procesamiento del *Frame* se realiza de manera secuencial, la orden de ejecución de cada *ImageProcessor* esta dada por el orden en el que son agregados al *ImageAnalyzer* (por medio del método *addNewProcessor*), además, como se puede observar, el *Frame* puede ser analizado tanto en CPU como en la GPU si cada uno de los *ImageProcessor* cuenta con la implementación correspondiente. Se puede realizar un tracking del proceso por medio del parámetro adicional *trackProcess*, el cual permite recopilar información sobre el tiempo invertido en procesar el *Frame* en cada uno de los *ImageProcessor*.
- **ImageProcessingSystem**: Abstracción de un *IPS Core*. Tiene asociado un nombre e información (sobre que hace) que lo identifican (los cuales se pueden obtener por medio de *getName()*, *getInfo()* respectivamente). Los métodos a ser implementados por cada uno de los *IPS Core* desarrollados son solamente 2: *initialize()* y *execute()*. El primero tiene como fin cargar todo lo necesario para su correcto funcionamiento, mientras que el segundo, es el encargado de realizar el proceso completo asociado al *IPS* (esto es, obtener nuevos frames del *ImageGenerator* y procesar cada uno de éstos).

5.3.4. IPS GUI

El conjunto de clases que conforman la interfaz gráfica (GUI) utilizada para la configuración del *IPS Core* se denomina *IPS GUI*. Un diseño de clases simplificado y a un nivel de abstracción se lo puede observar en el diagrama de la *Figura 28*.

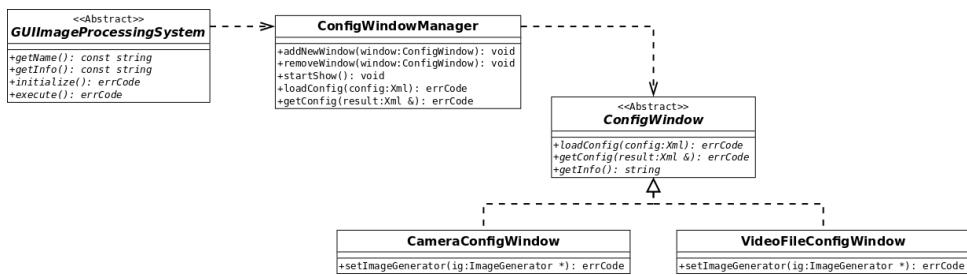


Figura 28: Diseño de clases simplificado de un *IPS GUI*.

Este diseño es el que se usará en la implementación (concreta) de los *IPS GUI* a desarrollar, como lo es el *WHIPS* (descripto en el siguiente capítulo). Presentaremos las principales características e interacciones que existen entre las clases del *IPS GUI* comenzando de lo mas particular a lo mas general.

- **CameraConfigWindow:** Clase utilizada para configurar los parámetros de la cámara web.
- **VideoFileConfigWindow:** Clase utilizada para configurar y manipular el vídeo (adelantar, retroceder, posicionar el vídeo en un instante determinado).
- **ConfigWindow:** Interfaz de una ventana de configuración (utilizada principalmente para configurar los *ImageProcessor*). Tiene asociado un nombre identificador (el cual puede ser obtenido por *getName()*). Cada ConfigWindow brinda la posibilidad además de poder cargar/descargar una configuración determinada por medio de un xml, debido a que en general, la configuración de cada *ImageProcessor* requiere de una previsualización a través de la interfaz gráfica.
- **Config WindowManager:** Clase encargada de manejar y administrar la forma en la que se mostraran las *Config Window* (tamaño, posición, orden de secuencia), como así también la carga y descarga de la configuración correspondiente a cada una de las ventanas por medio de un xml (a través de los métodos *loadConfig(xml)*, *getConfig(xml)*).
- **GUImageProcessingSystem:** Representación (abstracta) del *IPS GUI*. Al igual que el *IPS Core*, tiene asociado un nombre que lo identifica y una breve descripción sobre la finalidad del mismo (los cuales son accedidos por medio de los métodos *getName()*, *getInfo* respectivamente()), que en general, corresponden al nombre e información del *IPS Core*. Por otro lado cuenta con 2 funciones principales, que son *initialize()* utilizada para inicializar el *IPS Core* asociado y el sistema de ventanas a utilizar por el *IPS GUI*, por otro lado se encuentra *execute()* que ejecuta el sistemas de ventanas para comenzar la configuración del *IPS Core* y su posterior ejecución.

5.4. WHIPS

5.4.1. Introducción

Como se menciono en los capítulos anteriores, la funcionalidad del sistema se encuentra en las aplicaciones que pueda ejecutar, las cuales fueron

modeladas como *ImageProcessingSystem*. Éstos tienen un objetivo y funcionalidad específica. Como objetivo principal de este trabajo se planteo desarrollar una aplicación que por medio de una webcam pudiese capturar la altura de la superficie libre en tiempo real. Es por esto, y por medio del *framework* desarrollado y detallado anteriormente, se implemento un *IPS* con tal finalidad, el cual recibió el nombre de **WHIPS** (**Wave Height Image Processing System**). En este capitulo se mostrara el diseño y detalles del funcionamiento del mismo.

5.4.2. Diseño

El diseño de *WHIPS* parte de la base abstracta (diseño de abstracción) enseñado anteriormente. Por un lado se encuentra la interfaz gráfica y por el otro la unidad de procesamiento.

La unidad de procesamiento (o “*WHIPS Core*”), como se lo muestra en la *Figura 29*, cuenta con la clase principal **WaveHeightIPS** encargada de llevar a cabo todo el proceso de análisis de imagen para capturar la altura de la superficie libre. Depende de 4 *ImageProcessor* para lograrlo, estos son **IRectifier**, **IClipping**, **IBorderDetector**, **IAnalyzer**, cada uno con una funcionalidad determinada y representando a una de las etapas explicadas en el *capítulo 3.3 Técnicas utilizadas*. Como se puede observar en el diagrama, estos son clases abstractas que sirven solo para brindar una interfaz a las clases que hereden de estas. Esta capa de abstracción fue agregada para facilitar (y por medio de la utilización de la **WaveHeightIPFactory**) la posibilidad de cambiar cada uno de los *ImageProcessor* (concretos) de forma fácil sin alterar la la estructura general del modulo (por si se deseaba utilizar o probar distintos algoritmos en alguna de las etapas del proceso completo). En este caso en particular se muestran las 4 implementaciones correspondientes a cada una de las fases del proceso (cada uno de los *ImageProcessor*), éstos son **PerspectiveRectifier**, **MiddlePointClipping**, **CannyBorderDetector** y **WaveHeightAnalyzer** respectivamente. En el siguiente capitulo se detallara el funcionamiento de cada uno de estos.

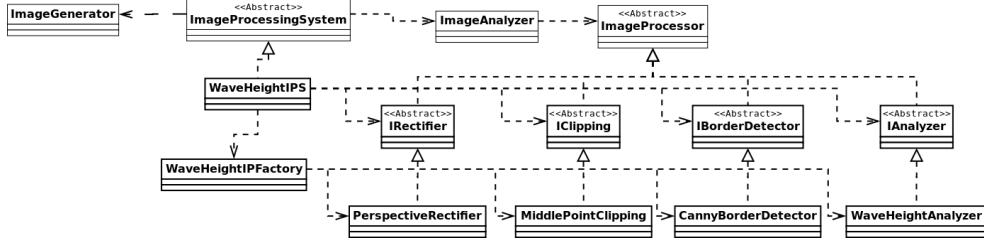


Figura 29: Diseño de clases del *WHIPS Core*.

Por otro lado se encuentra la interfaz gráfica (*Figura 30*) o “*WHIPS GUI*”, donde se pueden diferenciar 3 partes (remarcadas con distintos colores en la figura) con respecto a las etapas del proceso y sus respectivas clases. Éstas son la etapa de configuración o pre-procesamiento (**GUIPerspectiveRectifier**, **GUIMiddlePointClipping** y **GUICannyBorderDetector**), la etapa de procesamiento (**GUIRealTimeDataDisplayer**), y la etapa de post-procesamiento o análisis de datos (**DataAnalyzerWindow**).

Para la primera (configuración) las clases involucradas tienen como finalidad la configuración completa de los *ImageProcessor* pertenecientes al *WHIPS Core* (en el capítulo siguiente se explicara la relación que existe entre cada una de estas clases con los *ImageProcessor* que configuran). Estas clases heredan de una interfaz común (**ImgProcConfigWindow**) por el simple hecho de que poseen funcionalidad similar (lo que ahorra repetir código).

La segunda etapa tiene asociada la clase **GUIRealTimeDataDisplayer**, la cual muestra en tiempo real los datos procesados por el *WHIPS Core*, esto es, la altura de la superficie libre obtenida luego de procesar las imágenes de entrada (cámara de video o archivo). Adicionalmente, ésta clase muestra al usuario la imagen que esta siendo procesada para darle una idea sobre el transcurso del ensayo.

La ultima etapa (análisis de datos), esta conformada principalmente por la clase **DataAnalyzeWindow**, la cual muestra de manera gráfica los resultados de todo el análisis extrayendo y calculando valores de interés (*capítulo 2.6 Aplicaciones en laboratorio*). Como se puede observar, la clase fue diseñada independientemente a las demás (no hereda de ninguna otra clase) ya que debía funcionar independientemente del sistema de ventanas manejado por el **Config WindowManager**.

Por otro lado se puede observar que existen 2 implementaciones de *ConfigWindow* llamadas **CameraConfigWindow** y **VideoFileConfigWindow** que son utilizadas por el *GUIWaveHeightIPS*, ya sea para configurar

la cámara o el archivo de vídeo dependiendo que se use como sistema de entrada (osea, como este configurado el *ImageGenerator*).

En el siguiente capítulo se dará información mas detallada sobre el funcionamiento de cada una de las clases.

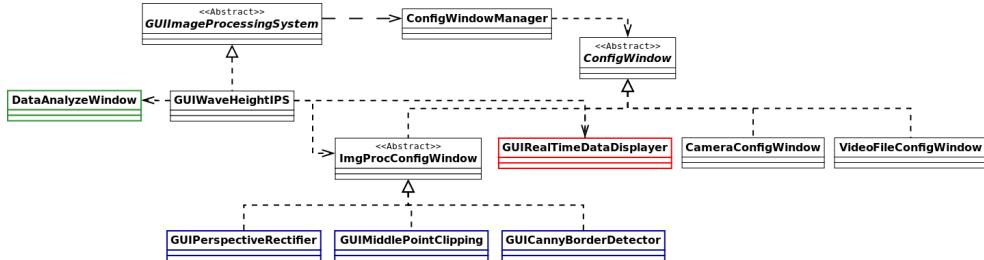


Figura 30: Diseño de clases del *WHIPS GUI*.

5.4.3. Funcionamiento WHIPS Core

WHIPS Core hace referencia a la unidad de procesamiento principal del *ImageProcessingSystem* encargado de detectar la altura de la superficie libre del fluido. Esta representado por la clase *WaveHeightIPS* la cual carga y coordina el procesamiento de las imágenes por medio de varias clases auxiliares.

El procedimiento del *WHIPS Core* se lo puede describir como una serie de pasos realizados en un orden determinado. Estos pasos son:

1. Obtener una imagen del *ImageGenerator* asociado a la clase (ya sea que este configurado para tomar imágenes desde un archivo o de una cámara).
2. Procesar por medio del *ImageAnalyzer* la imagen obtenida, estando este ultimo conformado por 4 *ImageProcessor*:
 - **IRectifier**: Encargado de transformar el sistema de coordenadas de la imagen tomada en un sistema de coordenadas “conocido” alineado al plano sobre el cual se esta calculando la altura. En particular esta acción se realiza por medio del *PerspectiveRectifier* utilizando el algoritmo “*Perspective Transform*” explicado en el *capítulo 4.2 Perspective transform*.
 - **IClipping**: Encargado de realizar el recorte de la imagen correspondiente a la zona que se quiere analizar. Esto lo hace actualmente el *MiddlePointClipping* el cual usa el punto de reposo de

la superficie seleccionado por el usuario y un parámetro adicional k que indica la cantidad de columnas a analizar. Lo que hace es simplemente descartar de la imagen de entrada, todos las columnas de píxeles que no estén contenidas entre el rango de columnas $[c.x - \frac{k}{2}, c.x + \frac{k}{2}]$, donde c es el punto de reposo seleccionado por el usuario (y $c.x$ la coordenada en x del punto c , correspondiente a la columna x de la imagen). De esta forma ahorramos procesar información que no es requerida (en el *capítulo 3.3.2 Recortador*, se detalla en mayor profundidad)

- **IBorderDetector:** Tiene como objetivo modificar la imagen de entrada de forma tal que la resultante contenga solo contornos. En este caso se implemento el *CannyBorderDetector* el cual transforma la imagen (de entrada) aplicándole en una primera instancia, un filtro que reduce la cantidad de colores (y por ende la cantidad de datos analizar o complejidad de la imagen). Luego aplica un segundo filtro denominado *Gaussian Blur* (*capítulo 4.3 Gaussian Blur*) para suavizar la imagen antes de realizar el análisis de contornos, que para lograrlo, aplica como ultimo paso, el algoritmo de *Canny* (*capítulo 4.4 Canny edge detector*). Dejando como resultado una imagen con solo contornos.
- **IAnalyzer:** Dada una imagen de contornos y una serie de parámetros, determina la altura de la superficie libre. La implementación de este *ImageProcessor* se lo denomino *WaveHeightAnalyzer* ya que se basa en el algoritmo “*Wave height analyzer*” (*capítulo 4.5 Wave height analyzer*).

3. Como ultimo paso, guarda el resultado devuelto por el *IAnalyzer* en un archivo previamente establecido por el usuario.

Este procedimiento (pasos 1,2,3) se realizan dentro de un bucle hasta que el usuario indica finalizar, o el *ImageGenerator* no puede obtener mas imágenes (que esto sucede cuando las imágenes son obtenidas de un archivo de vídeo).

Además, el *WaveHeightIPS* brinda la posibilidad de realizar el análisis en el CPU o GPU si los correspondientes *ImageProcessor* lo permiten. Para este caso en particular solo se implementó para procesar las imágenes en la CPU.

5.4.4. Funcionamiento WHIPS GUI

Este modulo tiene, a grandes rasgos, 3 finalidades distintas, estas son:

1. Brindar al usuario una interfaz gráfica para poder configurar de forma fácil y ágil los parámetros que requieren los *ImageProcessor* del *WHIPS Core* mencionados anteriormente.
2. Mostrar el proceso de computo en tiempo real (indicar de forma interactiva la altura de la superficie de ola calculada en los últimos segundos).
3. Mostrar y procesar los datos obtenidos una vez finalizado el proceso de captura.

El *WHIPS GUI* se basa en una clase principal (*GUIWaveHeightIPS*) la cual carga y ordena el sistema de ventanas como así también la inicialización y ejecución del *WHIPS Core*. Debido a la forma en la que los *ImageProcessor* deben ser configurados (en un orden secuencial, donde en general la configuración de uno depende de como fueron configurados los anteriores), la clase *Config WindowManager* fue diseñada para cumplir con tal fin, permitiendo solo pasar a la siguiente fase de configuración (y por ende a la siguiente *ConfigWindow*) si y solo si se configuro correctamente el paso (*ConfigWindow*) que esta siendo mostrado actualmente.

Adicionalmente, para facilitar aun mas la finalidad (1), el *GUIWaveHeightIPS* brinda la posibilidad de cargar y guardar las configuraciones de las distintas *ConfigWindow*, que en caso de realizar múltiples ensayos con la misma configuración (como es usual) ahorra tiempo al usuario evitando restablecer todos los parámetros de nuevo.

Por otro lado se brinda la posibilidad de manejar sesiones donde se pude guardar fecha, autor, nombre, descripción, entre otros, asociadas a un experimento dado.

A manera esquemática, podemos mostrar la funcionalidad (secuencia de pasos) que brinda este modulo por medio del diagrama (conjunción de diagrama de clases y diagrama de flujo) de la *Figura 31*.

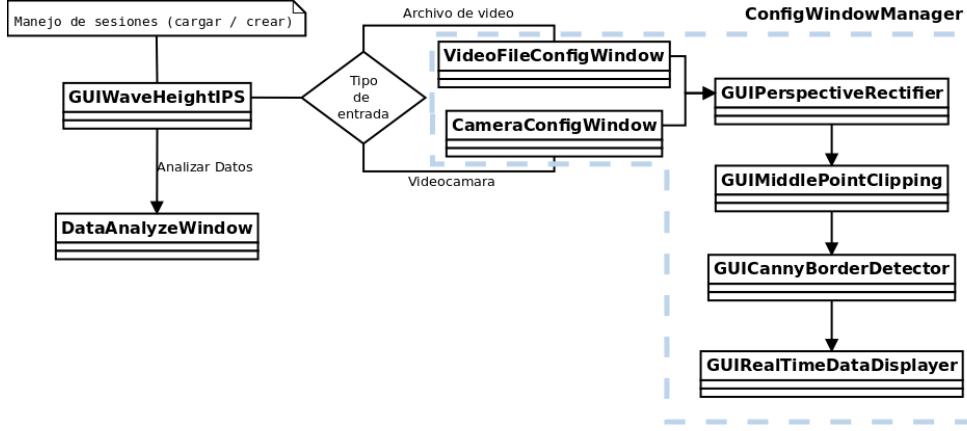


Figura 31: Esquema funcional y estructural representativo del *WHIPS*.

En un principio, se le brinda al usuario la posibilidad de crear o cargar una sesión ya existente. Puede además, ejecutar directamente la ventana de análisis de datos (*DataAnalyzeWindow*), la cual funciona como modulo aparte (puede mostrarse datos ya obtenidos de una sesión anterior, o de datos externos no necesariamente de una sesión anterior). Esto esta relacionado con la funcionalidad (3) anteriormente enumerada.

Dependiendo del tipo de entrada seleccionado (que puede ser una cámara de vídeo o un archivo), el sistema muestra la ventana de configuración correspondiente (en la figura, *VideoFileConfigWindow* y *CameraConfigWindow*). Luego se pasa al ciclo de configuración de los diferentes *ImageProcessor* del *WHIPS Core*, por medio de las ventanas de configuración *GUIPerspectiveRectifier*, *GUIMiddlePointClipping* y *GUICannyBorderDetector*. Cabe destacar, que en el caso del tipo de entrada ser un archivo de vídeo, se muestra intercaladamente (entre cada una de estas ventanas) el *VideoFileConfigWindow* para brindar la posibilidad de adelantar o retrasar el vídeo a la posición que se requiera (esto es de utilidad si las diferentes configuraciones necesarias están asociadas a distintos frames en el vídeo). Este conjunto de ventanas son utilizadas para lograr (1).

Por ultimo, una vez configurado todo los *ImageProcessor*, el usuario puede comenzar la captura de datos y observarlos en tiempo real gracias a la ventana *GUIRealTimeDataDisplayer* (finalidad (2)).

En el siguiente capítulo se describe una serie de ensayos realizados donde se muestran las interfaces gráficas de cada una de las ventanas que componen al *IPS GUI*.

6. Aplicación en un canal 2D

6.1. Canal de laboratorio

Los ensayos que se llevaron a cabo en este trabajo se realizaron en el *Laboratorio de Hidráulica²⁶* de la *Facultad de Ciencias Exactas, Físicas y Naturales (UNC)*.

En la *Figura 32* se muestra el canal en donde se realizaron los experimentos, el cual posee una longitud de 25 metros, 0.50 m de ancho y 0.80 m de altura. Cuenta con una pala mecánica *HR Wallingford²⁷* (*Figura 33*) la cual genera los oleajes que se quieran simular introduciendo por computadora H y T (altura y periodo de onda respectivamente, *capítulo 2.1 Introducción*) en el caso del oleaje regular, mientras que el oleaje irregular se lo simula estableciendo los valores H_s , T_p y $\gamma = 3,3$ con espectro JONSWAP.

Por otro lado, el software utilizado por la pala mecánica permite extraer el espectro de la señal medido por los sensores y el valor de H_s del oleaje simulado, que sirve como referencia comparativa para los datos capturados por el sistema implementado para este trabajo.



Figura 32: Canal del Laboratorio de Hidráulica F.C.E.F.yN. donde se realizaron los ensayos.

²⁶<http://www.efn.uncor.edu/investigacion/hidraulica/>

²⁷<http://equipment.hrwallingford.co.uk/>



Figura 33: Pala mecánica *HR Wallingford* generadora de oleaje bidimensional.

Para el caso del sistema implementado en este trabajo, las mediciones de la altura de ola se realizaron utilizando una cámara Logitech C910²⁸ y en paralelo utilizando los sensores de nivel resistivo (*Figura 10c*), los cuales tienen la capacidad de realizar un muestreo de 100 tomas por segundo (o equivalentemente 100 Hz).

6.2. Preparación de las mediciones

Para poder realizar una medición se debe preparar tanto el canal (y el entorno) como el sistema. En una primera instancia se debe acondicionar el canal de forma tal que se pueda alcanzar una máxima diferenciación de la superficie libre, esto es, conseguir el mayor contraste posible entre el líquido y el aire (variando las posiciones e intensidades de la luz que afectan la zona a medir en el canal y utilizando distintos tipos de papeles o contactos sobre la cara opuesta a medir del canal, evitando de esta manera, capturar objetos que se encuentran por detrás del mismo).

Por otro lado se debe configurar el sistema. Vamos a mostrar el caso en el que se realiza una medición nueva (desde cero), los pasos a seguir son:

²⁸<http://www.logitech.com/es-es/webcam-communications/webcams/devices/6816>

- Crear una nueva sesión (*Figura 34*)

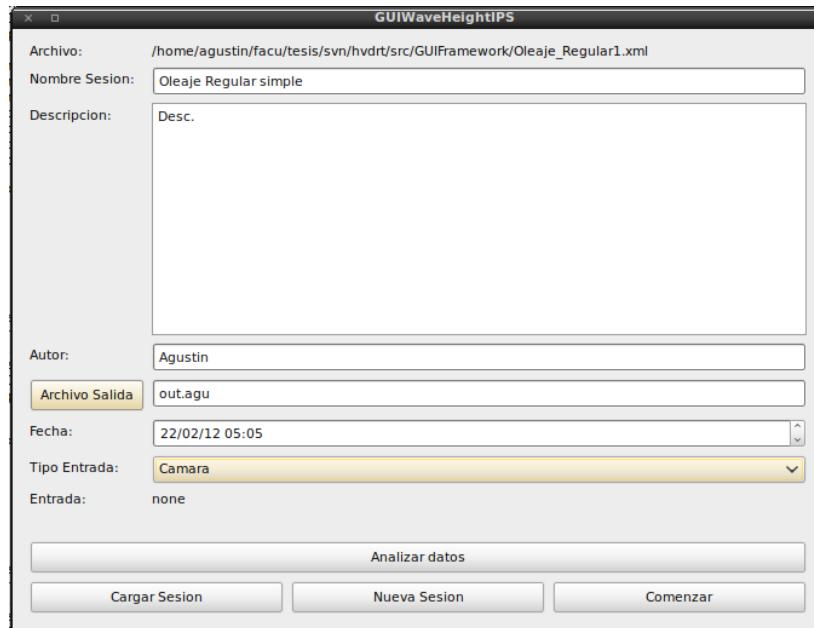


Figura 34: Ventana: crear/cargar una sesión.

- Configuración de la cámara utilizada (foco, zoom, brillo, etc), como se lo muestra en la *Figura 35*.

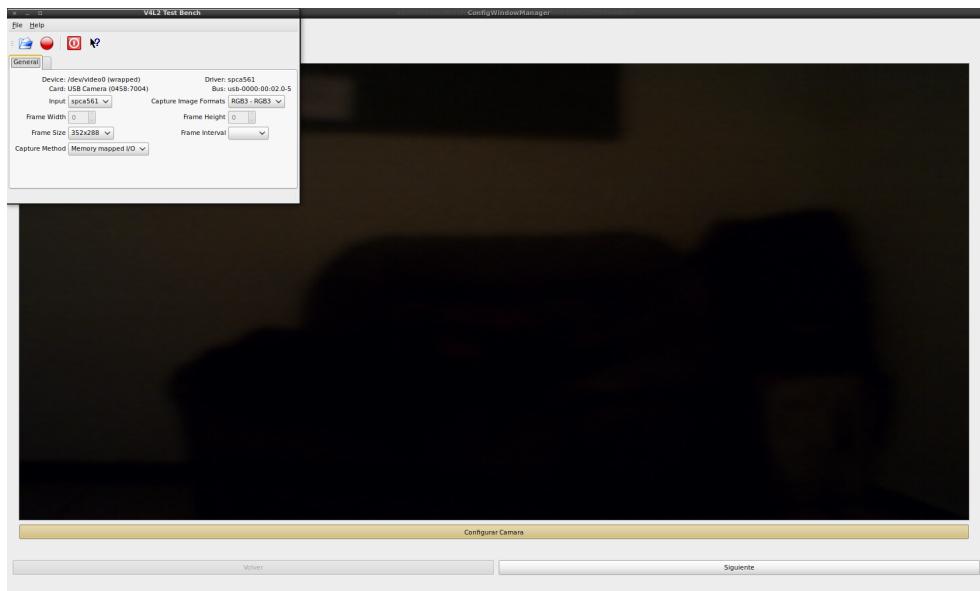


Figura 35: Ventana de configuración (*CameraConfigWindow*): Configuración de la cámara.

- Configurar la escala y zona de análisis (*Figura 36*).

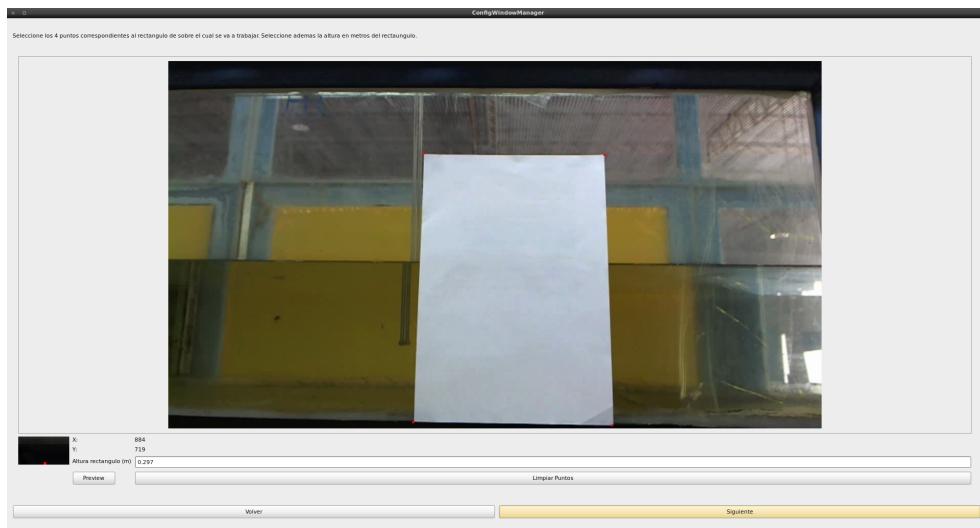


Figura 36: Ventana de configuración (*GUIPerspectiveRectifier*): Configuración de zona de análisis, en este caso el rectángulo blanco (donde los 4 puntos rojos en cada uno de los vértices fueron seleccionados por el usuario).

- Configurar el punto de reposo de la ola (para lograr esto el nivel del fluido debe estar en estado estacionario). La ventana asociada se la muestra en la *Figura 37*.

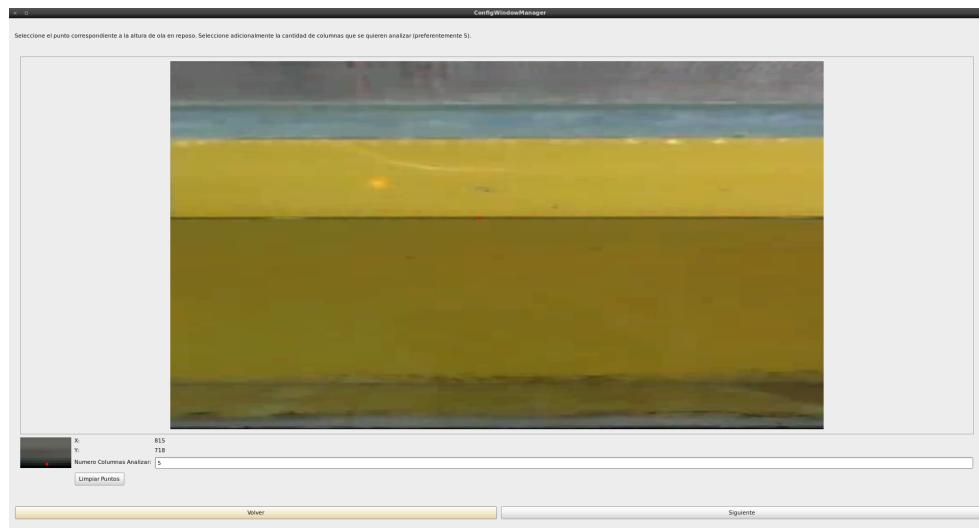


Figura 37: Ventana de configuración (*GUIMiddlePointClipping*): Configuración del punto de reposo de ola (indicado por el pequeño punto rojo seleccionado por el usuario).

- Configuración de los parámetros del algoritmo para la detección de contornos (*Figura 38*).

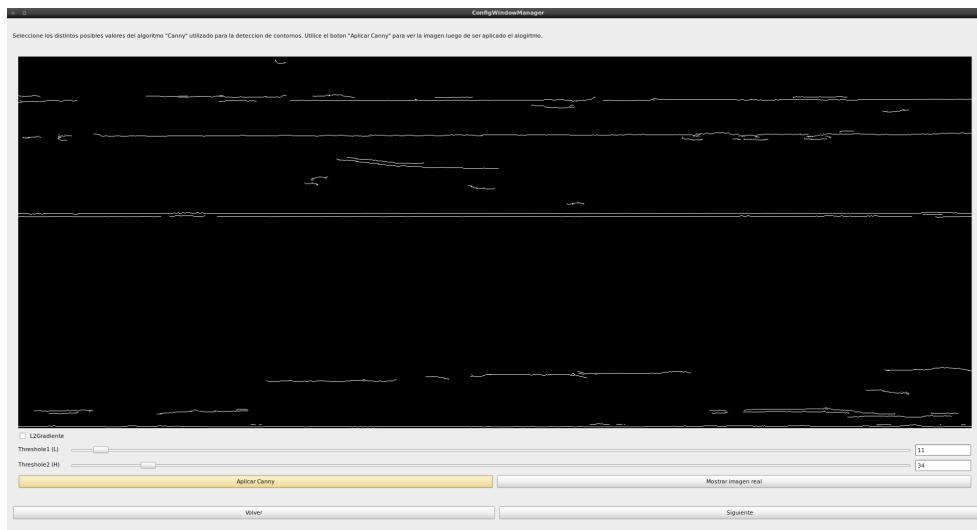


Figura 38: Ventana de configuración (*GUICannyBorderDetector*): Configuración de los parámetros asociados al algoritmo de *Canny*.

Una vez configurado el sistema se prosigue a producir el oleaje que se quiere analizar por medio de la pala mecánica antes mencionada (introduciendo por computadora los valores H y T en el caso de oleaje regular y H_s , T_p y $\gamma = 3,3$ con espectro JONSWAP para oleaje irregular) y por medio de una ultima ventana (*Figura 39*) se le brinda al usuario la posibilidad de comenzar a capturar las imágenes y observar en tiempo real los datos procesados.

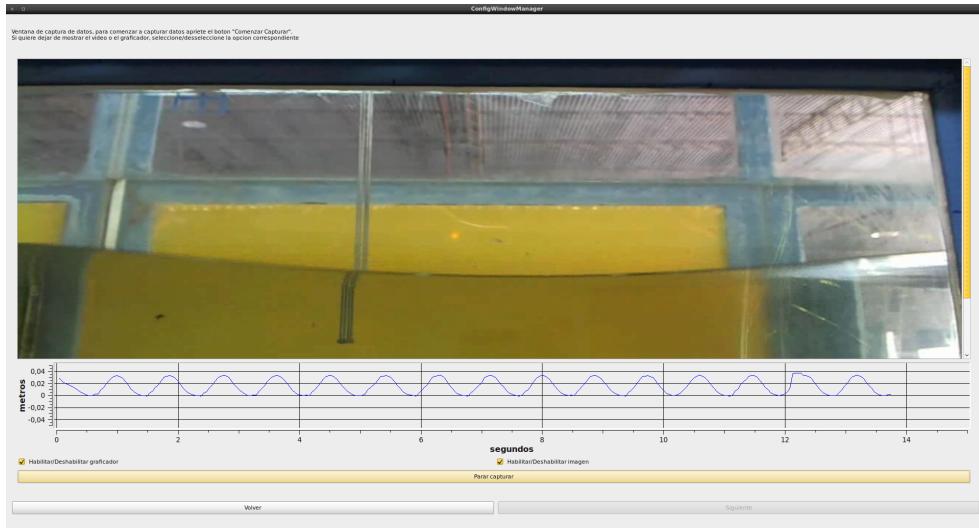


Figura 39: Ventana (*GUIRealTimeDataDisplayer*): Muestra los datos procesados en tiempo real al igual que las imágenes capturadas por el dispositivo.

6.3. Ensayos propuestos

En hidráulica marítima normalmente se realizan ensayos con oleajes regulares (monocromáticos) o irregulares (con muchas frecuencias asociadas). Debido a que el oleaje irregular tiene mayor peso en el estudio ingenieril (ya que es mas representativo de la realidad) y en el caso del sistema es el mas interesante para medir (debido a su irregularidad), se decidió realizar la mayoría de los ensayos en base a oleajes irregulares. Por otro lado se intento mantener constante los factores externos (como luz del entorno, intensidad, zona de captura) a excepción de la posición de la cámara en donde se trabajo con 2 ubicaciones distintas con el fin de verificar como reaccionaba el sistema (en cuanto a precisión y calidad de los datos obtenidos en base a los capturados por el sensor resistivo) para cada una de éstas.

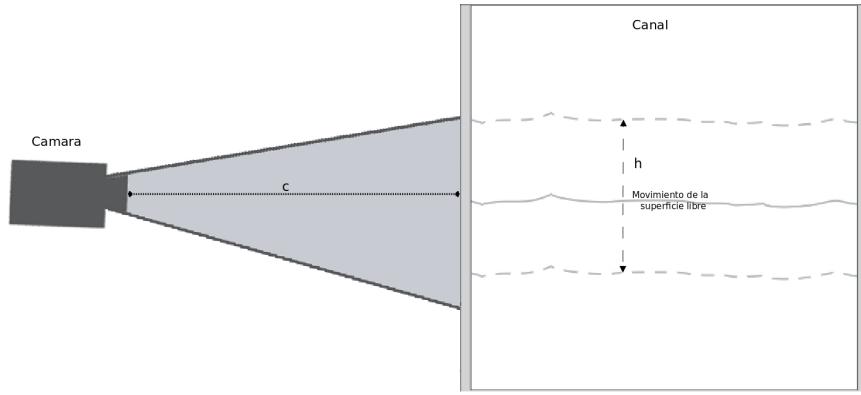
Los ensayos propuestos para la primera ubicación de la cámara (*P1*, *Figura 40a*) se listan a continuación:

- **EN1:** Capturar la altura de ola en un oleaje regular con parámetros $H = 6 \text{ cm}$ y $f = 1 \text{ Hz}$.
- **EN2:** Capturar la altura de ola en un oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 5 minutos.

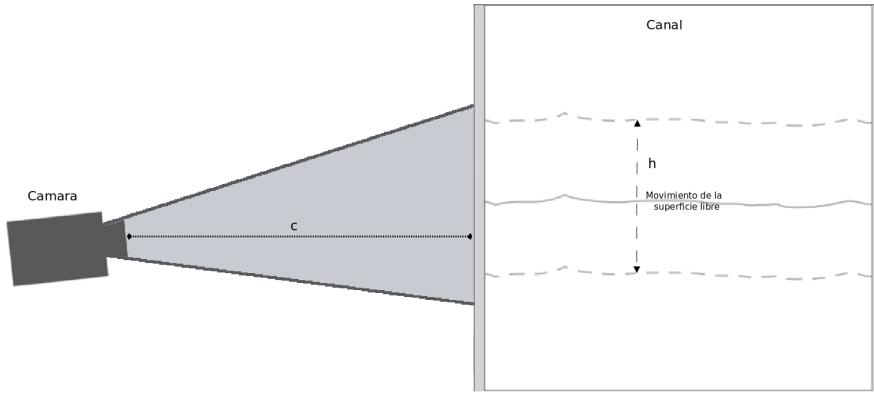
- **EN3:** Capturar la altura de ola en un oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 30 minutos.

Para la segunda posición de la cámara ($P2$, *Figura 40b*):

- **EN4:** Capturar la altura de ola en un oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 5 minutos.
- **EN5:** Capturar la altura de ola en un oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 30 minutos.



(a) Posición 1: El nivel de reposo de la superficie libre se encuentra aproximadamente a la altura del lente de la cámara. Ésta se encuentra a una distancia $c \approx 0,5 \text{ m}$ con respecto a la pared del canal (*zona de captura*).



(b) Posición 2: El nivel de reposo de la superficie libre se encuentra por encima del lente de la cámara. Ésta se encuentra posicionada a $c \approx 0,90 \text{ m}$ de la pared del canal (*zona de captura*).

Figura 40: Posiciones de la cámara en los distintos ensayos realizados. En ambas figuras se muestra la posible altura h de la superficie libre (tanto por encima como por debajo del nivel de reposo).

Donde recordando que H es la altura de ola (en el caso de un oleaje regular), H_s es la altura de ola significante en un oleaje irregular, y f_p es la frecuencia de pico o la frecuencia en la que el espectro del oleaje contiene mayor energía.

En conjunto con los ensayos propuestos anteriormente, se realizarían mediciones utilizando los sensores resistivos con los que cuenta el labora-

torio con el fin de poder comparar los resultados obtenidos. De manera esquemática, como se lo muestra en la *Figura 41*, podemos observar la columna de píxeles analizados por el sistema para detectar la altura de ola al costado del sensor resistivo (debido a que el sistema se basa en detectar contornos, la columna de píxeles no puede estar sobre el sensor). A causa de esto, los resultados obtenidos por el sistema implementado se encuentran desfasados en el tiempo con respectos a los del sensor resistivo, por lo que, es necesario modificar (previamente a la comparación) los datos de manera tal que queden sincronizados en el tiempo.

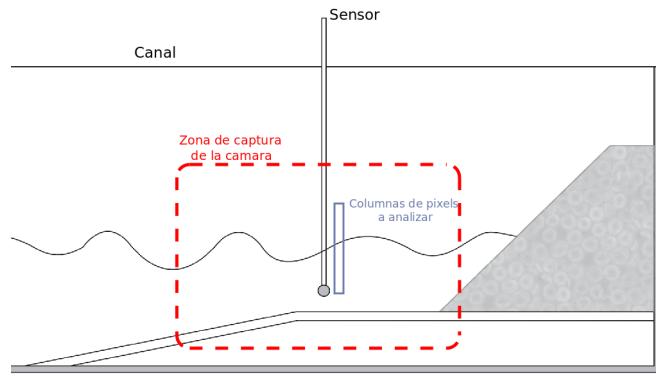


Figura 41: Esquema de la zona de captura y analizada por el sistema implementado en el canal de laboratorio.

6.4. Mediciones y análisis de los resultados

6.4.1. Que medimos?

En cada uno de los ensayos realizados (mencionados anteriormente) el sistema implementado captura la altura de ola a lo largo del tiempo en un punto determinado del canal. Este conjunto de datos (altura de ola en el tiempo) fue comparado con los obtenidos por los sensores resistivos utilizados en el laboratorio con el fin de poder determinar la precisión del sistema implementado. En base a esto se puede realizar dos tipos de comparaciones distintas: a) los datos “crudos” (altura en el tiempo); b) realizar un análisis espectral del oleaje obtenido por el sistema y por el sensor y comparar los parámetros asociados al mismo (H y f en el caso del oleaje regular, o H_s y f_p en conjunto con los gráficos espectrales asociados - $S(f)$ y *JONSWAP*- a los oleajes irregulares).

Todas las comparaciones llevadas a cabo se realizaron por medio del

“modulo” de análisis desarrollado en este trabajo, el cual se muestra en la *Figura 42*. En la parte superior se muestran los datos crudos obtenidos asociados a un oleaje irregular, por debajo a la derecha se observa en rojo el calculo del espectro $S(f)$ y en color magenta el espectro de *JONSWAP* (*capítulo 2.4 Descripción espectral del oleaje.*). En la parte inferior izquierda se muestran además otros parámetros calculados como H_s , T_p , H . Esta ventana permite además cargar mas de un conjunto de datos en simultaneo lo que permite comparar hasta tres oleajes diferentes al mismo tiempo con sus respectivos espectros y parámetros asociados.

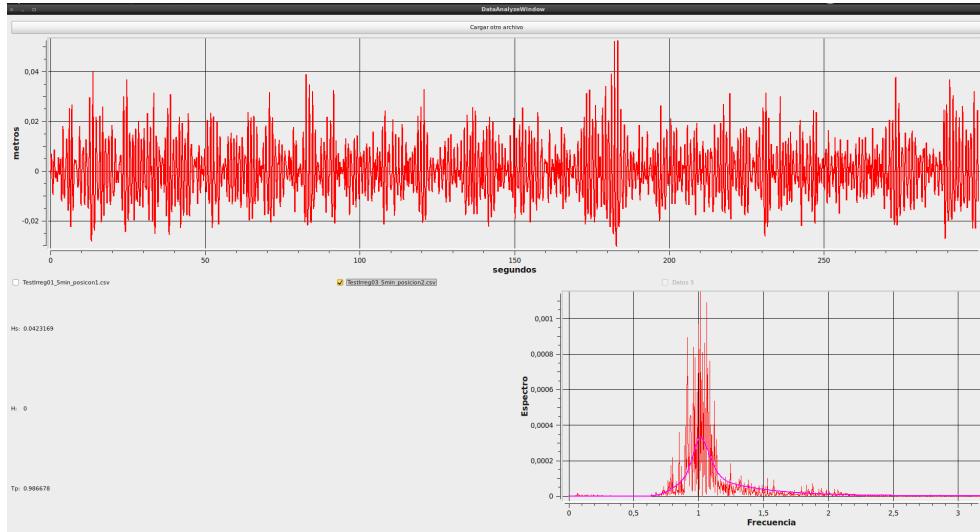


Figura 42: *DataAnalyzeWindow*: Ventana en donde se realizan los análisis espectrales y comparaciones de los datos obtenidos (por el sistema implementado o por el sensor resistivo).

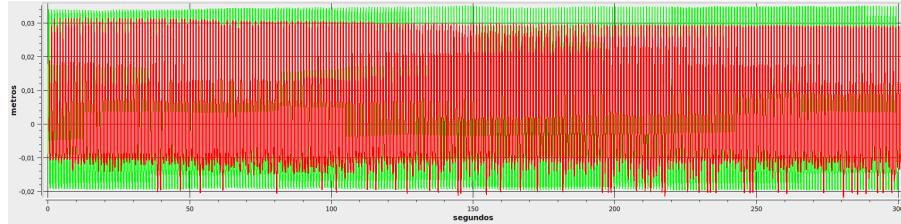
En los siguientes subcapítulos se detalla la comparación realizada en cada uno de los ensayos realizados.

6.4.2. EN1: Oleaje regular

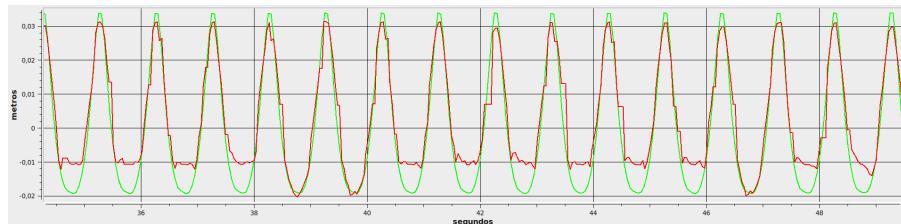
Se trata de un ensayo en donde se simula un oleaje con una altura de ola de 6 cm ($H = 6 \text{ cm}$) a una frecuencia de 1Hz ($f_p = 1 \text{ Hz}$), esto es, se genera una ola de 6 cm de altura por segundo (por medio de la pala mecánica mencionada anteriormente).

En la *Figura 43* podemos ver los datos obtenidos por ambos sensores superpuestos en un mismo gráfico, en 43a se observan los resultados obtenidos

a lo largo de todo el ensayo, donde a simple vista la altura de ola capturada por la cámara difiere por encima (0,0047 metros aproximadamente) como por debajo (0,0074 metros aproximadamente) de la altura capturada por el sensor en base al punto de reposo (valor 0 m en el gráfico). Podemos observar además en 43b la presencia de ruido durante la captura realizada por el sistema, en donde la gran mayoría de las ondas (la parte inferior o *seno* de la ola -color rojo-), esta cortada abruptamente (por ejemplo las primeras 4 olas muestran un corte abrupto en su parte inferior, mientras que la 5ta y 6ta no). La causa de esto es el incorrecto posicionamiento de la cámara como será explicado en el apartado 6.5.



(a) Datos de los 5 minutos de captura por ambos métodos y superpuestos.



(b) Ampliación (zoom) de una fracción de los datos capturados por ambos métodos y superpuestos.

Figura 43: Comparación entre los datos crudos obtenidos por ambos métodos (cámara y sensor resistivo) de un oleaje regular. En color rojo se muestra el obtenido por el sistema implementado (vídeo cámara) y en verde el capturado por el sensor resistivo.

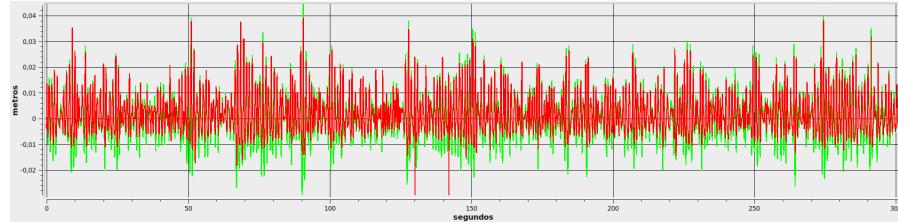
Por otro lado se obtuvieron los siguientes valores de H y f para cada uno de los sistemas de medición:

Método	H (m)	f (Hz)
sensor resistivo	0.051	1
sistema implementado	0.041	0.984

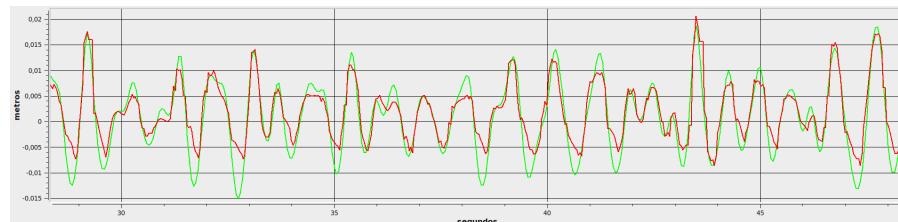
Tabla 1: Comparación de los valores H y f del oleaje regular realizado en EN1.

6.4.3. EN2: Oleaje irregular de 5 minutos

Para este ensayo se simulo un oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 5 minutos por medio de la pala mecánica. Los datos capturados de la altura de ola por ambos métodos (sistema implementado y sensor resistivo) se muestran en la *Figura 44*.



(a) Datos obtenidos durante los 5 minutos.



(b) Datos obtenidos durante 50 segundos (segundo 28 al 48 aproximadamente).

Figura 44: Superposición de los datos obtenidos por el sistema implementado (en rojo) y el sensor resistivo (en verde) del oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 5 minutos (EN2).

Como se puede observar (*Figura 44b*), en la gran mayoría de los *senos* (parte inferior) de las olas, el sistema implementado difiere bastante (hasta 0,0153 metros aproximadamente en los peores casos) con respecto al obtenido por el sensor resistivo. Por otro lado se puede ver la presencia de pequeños

saltos a lo largo de la linea roja del gráfico, esto es otro problema relacionado con la velocidad de captura de la cámara el cual se detalla en el capítulo 6.5.

Por otro lado (*Figura 45*) se comparan los espectros obtenidos por el sensor resistivo y por el sistema. Claramente se puede observar (a través del espectro de *JONSWAP*) que por un lado la simetría de ambos espectros se asemeja, pero que el asociado a los datos obtenidos por el sistema implementado es inferior al del obtenido por los del sensor resistivo (como se lo percibe en la *Figura 44b* en donde las ondas capturadas por el sensor resistivo poseen una mayor amplitud que las capturadas por el sistema).

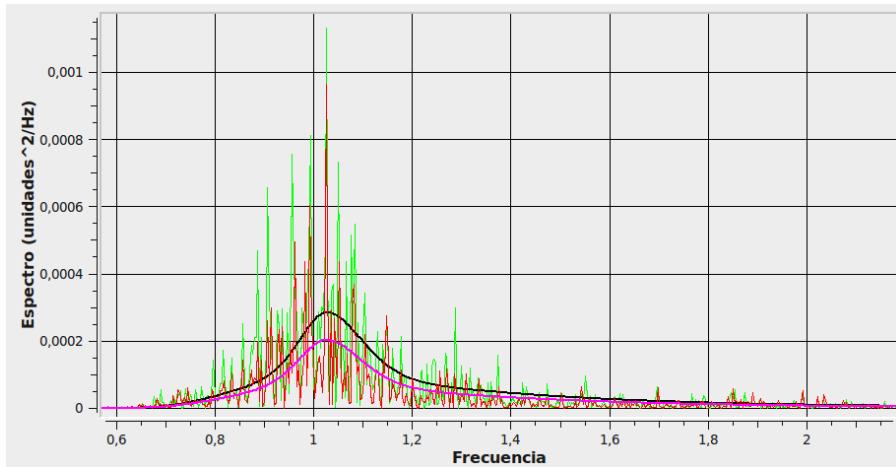


Figura 45: Espectros calculados para los datos obtenidos por ambos métodos (sistema implementado y sensor resistivo) para un oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 5 minutos (**EN2**). En verde (y negro) se observa el espectro resultado (y espectro de *JONSWAP* respectivamente) de los datos capturados por el sensor resistivo mientras que en rojo (y magenta) el espectro calculado (y espectro de *JONSWAP*) por el sistema.

Se calcularon además los parámetros de interés (H_s , H , T_p) asociados al oleaje capturado por ambos métodos, como se lo muestra en la siguiente tabla:

Método	H_s (m)	H (m)	T_p (s)
sensor resistivo	0.040	0.024	0.974
sistema implementado	0.034	0.016	0.975

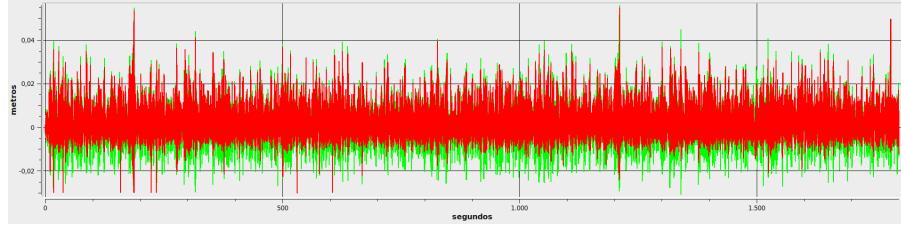
Tabla 2: Comparación de los parámetros H_s , H y T_p asociados a los datos capturados por el sistema implementado y el sensor resistivo en el ensayo **EN2**.

Claramente se observa una gran diferencia entre las alturas de olas calculadas (H), mientras que el periodo de pico (T_p) se asemeja bastante.

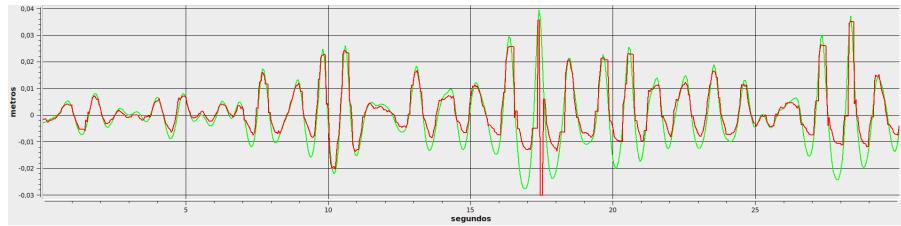
6.4.4. EN3: Oleaje irregular de 30 minutos

En general, los ensayos realizados en ingeniería (y en el laboratorio) duran alrededor de los 30 minutos. Con este ensayo se puso a prueba la factibilidad de capturar durante un periodo relativamente prolongado de tiempo la altura de ola, debido a que otras técnicas anteriormente utilizados en el laboratorio no eran capaces de hacerlo por falta de memoria o velocidad de computo (solo se lograba realizar un video de 10 minutos con su posterior procesamiento de aproximadamente unos 30 minutos).

Para este ensayo (oleaje irregular con parámetros $H_s = 5\text{ cm}$ y $f_p = 1\text{ Hz}$ y duración de 30 minutos) se capturaron los datos mostrados en la *Figura 46*. A grandes rasgos se puede ver que nuevamente los *senos* (parte inferior de las olas) difieren en gran medida (los datos obtenidos con el sistema son menores que los del sensor resistivo). Además se puede observar (*Figura 46b*) la existencia de cambios repentinos de una *cresta* (parte superior de la ola) a un *seno* cerca del segundo 17.5 causado por una errónea detección de la altura de ola (explicado en el *capítulo 6.5*).



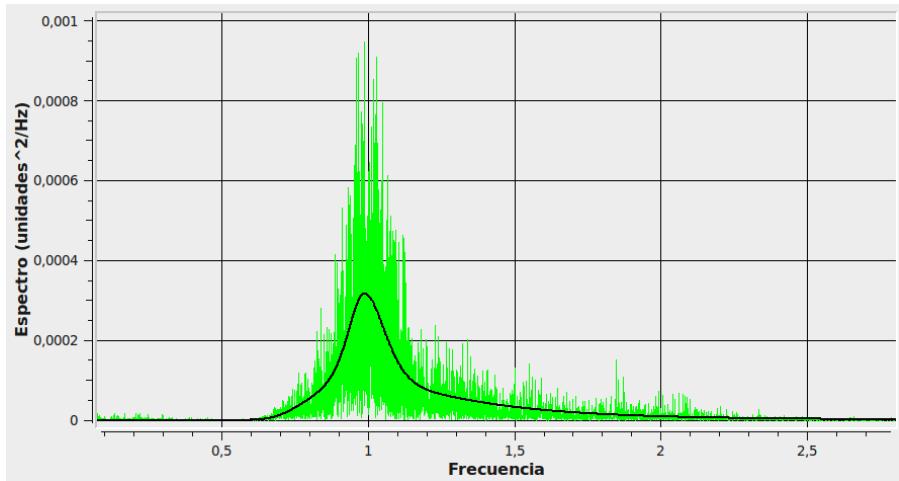
(a) Datos obtenidos durante los 30 minutos.



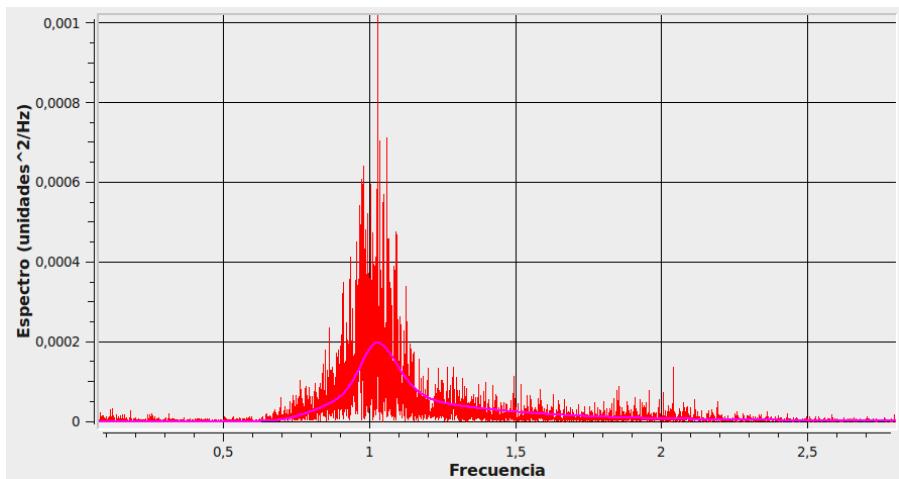
(b) Datos obtenidos durante los 30 primeros segundos.

Figura 46: Superposición de los datos obtenidos por el sistema implementado (en rojo) y el sensor resistivo (en verde) del oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 30 minutos (EN3).

En las *Figuras 47* y *48* se muestran los respectivos espectros, donde se observa por medio del espectro de JONSWAP una diferencia mayor que los ensayos anteriores en la simetría y altura del mismo. Por medio de los parámetros mostrados en la *Tabla 3* se puede apreciar con mayor detalle esto.

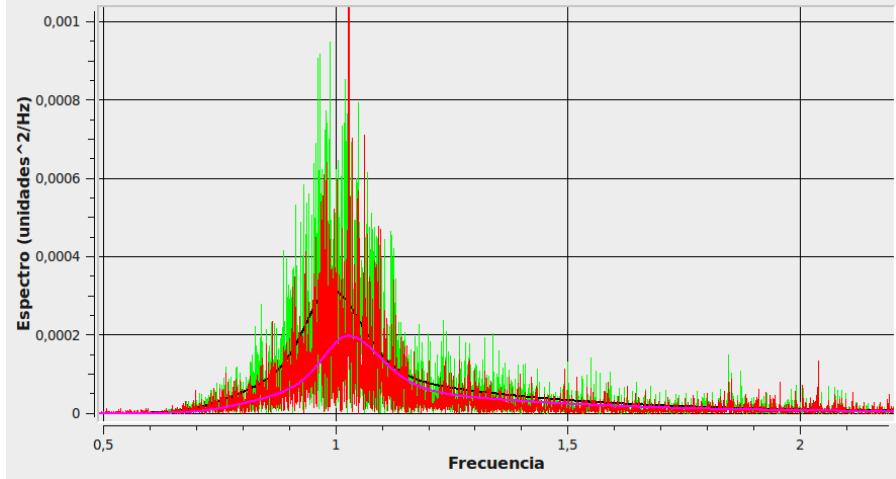


(a) Espectros asociado a los datos capturados por el sensor resistivo.



(b) Espectros asociado a los datos capturados por el sistema.

Figura 47: Espectros calculados para los datos obtenidos por ambos métodos (sistema implementado y sensor resistivo) para un oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 30 minutos (**EN3**).



(a) Superposición de los espectros de ambos métodos.

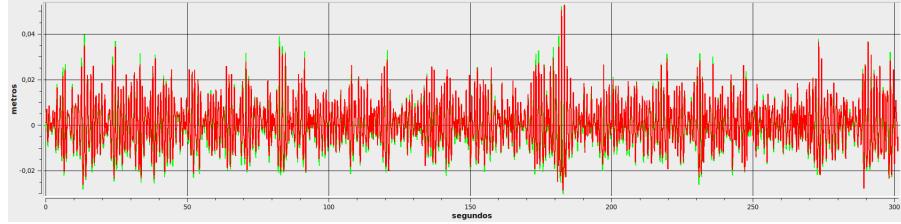
Figura 48: Espectros calculados para los datos obtenidos por ambos métodos (sistema implementado y sensor resistivo) para un oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 30 minutos (**EN3**). En verde (y negro) se observa el espectro resultado (y espectro de *JONSWAP* respectivamente) de los datos capturados por el sensor resistivo mientras que en rojo (y magenta) el espectro calculado (y espectro de *JONSWAP*) por el sistema.

Método	$H_s \text{ (m)}$	$H \text{ (m)}$	$T_p \text{ (s)}$
sensor resistivo	0.040	0.023	1.011
sistema implementado	0.033	0.015	0.973

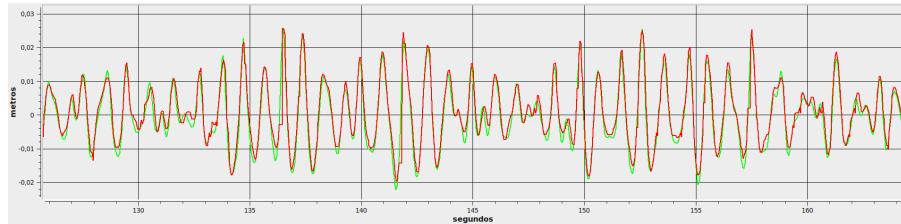
Tabla 3: Comparación de los parámetros H_s , H y T_p asociados a los datos capturados por el sistema implementado y el sensor resistivo en el ensayo **EN3**.

6.4.5. EN4: Oleaje irregular de 5 minutos

A diferencia del ensayo anterior de 5 minutos (**EN2**), este se realizó con la cámara en una posición distinta (*Figura 40b*) logrando mejores resultados debido a que al estar la cámara más alejada la interferencia detectada (junto con la velocidad de captura de la cámara) es menor (en el *capítulo 6.5* se detallan las causas). En la *Figura 49* se pueden observar los resultados obtenidos por ambos métodos.



(a) Datos obtenidos durante los 5 minutos.



(b) Datos obtenidos durante 38 segundos (del segundo 126 al 164).

Figura 49: Superposición de los datos obtenidos por el sistema implementado (en rojo) y el sensor resistivo (en verde) del oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 5 minutos (**EN4**) con la cámara ubicada a 90 cm del canal.

Se puede ver que los datos coinciden en su gran mayoría (donde la máxima diferencia encontrada es de 0,0055 metros), además de que no se perciben a grandes rasgos cambios abruptos como en los casos anteriores. Si ahora observamos los espectros asociados a los datos (*Figura 50*) podemos decir que se asemejan tanto en simetría como amplitud con un pequeño desfasaje sobre el eje x (en el espectro de *JONSWAP* asociado a los datos capturados por el sistema en base al generado por los datos del sensor resistivo) y de menor amplitud (causado por la diferencia en los *senos* detectados de las olas).

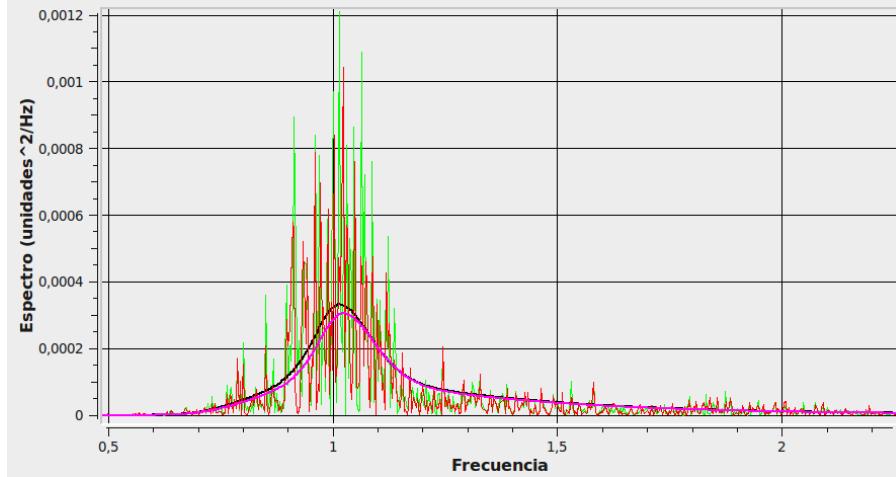


Figura 50: Espectros calculados para los datos obtenidos por ambos métodos (sistema implementado y sensor resistivo) para un oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 5 minutos (**EN2**). En verde (y negro) se observa el espectro resultado (y espectro de *JONSWAP* respectivamente) de los datos capturados por el sensor resistivo mientras que en rojo (y magenta) el espectro calculado (y espectro de *JONSWAP*) por el sistema.

En la siguiente tabla (*Tabla 4*) podemos observar nuevamente la similitud de los 3 valores de interés.

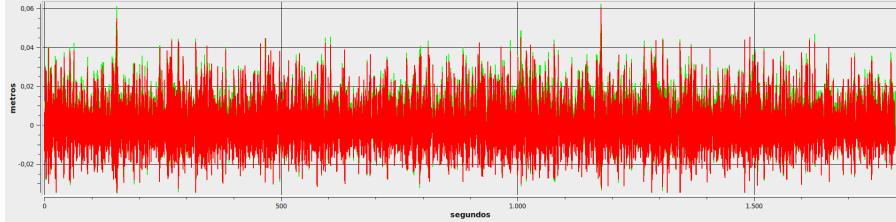
Método	$H_s \text{ (m)}$	$H \text{ (m)}$	$T_p \text{ (s)}$
sensor resistivo	0.042	0.024	0.987
sistema implementado	0.041	0.023	0.978

Tabla 4: Comparación de los parámetros H_s , H y T_p asociados a los datos capturados por el sistema implementado y el sensor resistivo en el ensayo **EN4**.

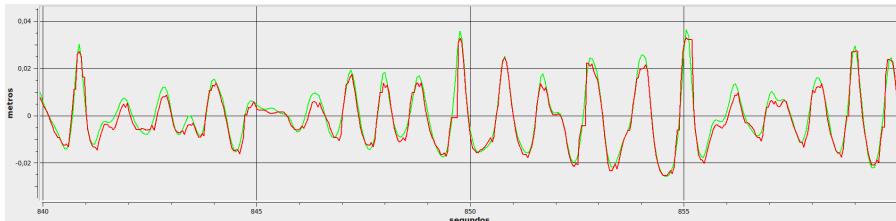
6.4.6. EN5: Oleaje irregular de 30 minutos

En este ensayo se simuló un oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 30 minutos y posicionando la cámara a unos 90 cm del canal (como lo muestra la *Figura 40b*). A diferencia del ensayo anterior, se decidió modificar la configuración interna del algoritmo de detección de contornos (*Canny, capítulo 4.4. Canny edge detector*) estableciendo valores de umbrales más bajos de forma tal que la detección sea más sensible. Los resultados obtenidos se muestran en la *Figura 51*. Se puede observar que a

diferencia de **EN3** (el otro ensayo de 30 minutos) se obtuvieron mejores resultados (con mayor precisión) pero no así en comparación al ensayo anterior (**EN4**).



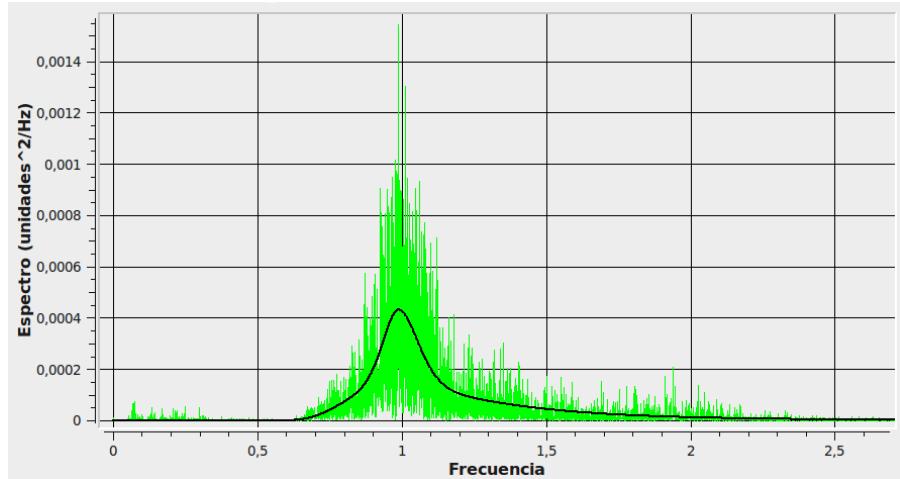
(a) Datos obtenidos durante los 30 minutos.



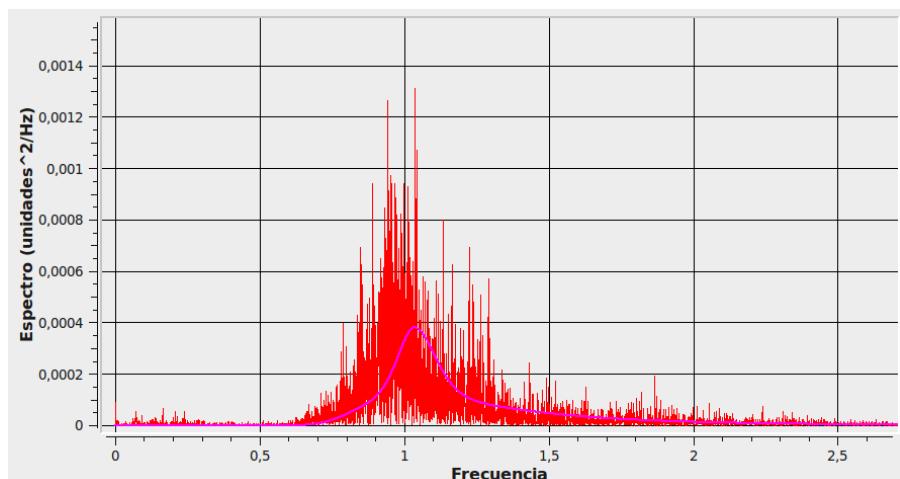
(b) Datos obtenidos durante 20 segundos (del segundo 840 al 900).

Figura 51: Superposición de los datos obtenidos por el sistema implementado (en rojo) y el sensor resistivo (en verde) del oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 30 minutos (**EN5**) con la cámara ubicada a 90 cm del canal.

En las *Figuras 52* y *53* se pueden observar los distintos espectros resultantes y en la *Tabla 5* los valores de los parámetros de interés.



(a) Espectros asociado a los datos capturados por el sensor resistivo.



(b) Espectros asociado a los datos capturados por el sistema.

Figura 52: Espectros calculados para los datos obtenidos por ambos métodos (sistema implementado y sensor resistivo) para un oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 30 minutos (**EN5**).

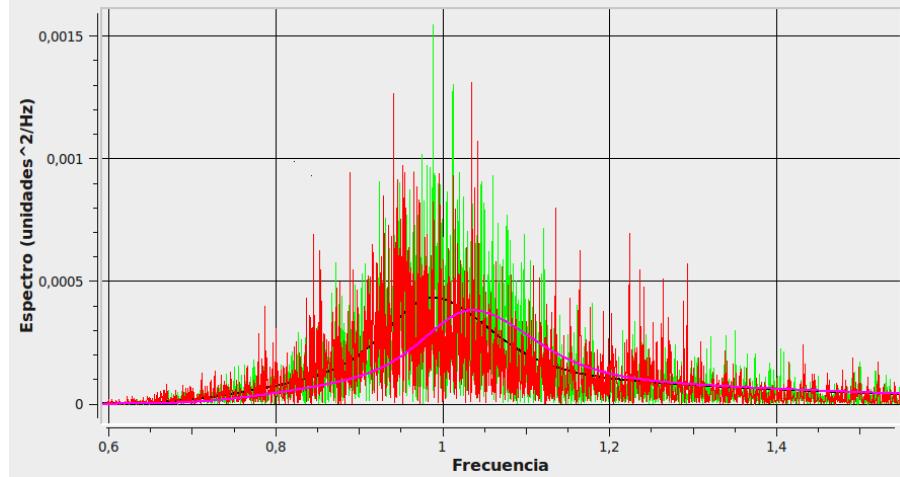


Figura 53: Espectros superpuestos calculados de los datos obtenidos por ambos métodos (sistema implementado y sensor resistivo) para un oleaje irregular con parámetros $H_s = 5 \text{ cm}$ y $f_p = 1 \text{ Hz}$ durante 30 minutos (**EN5**). En verde (y negro) se observa el espectro resultado (y espectro de *JONSWAP* respectivamente) de los datos capturados por el sensor resistivo mientras que en rojo (y magenta) el espectro calculado (y espectro de *JONSWAP*) por el sistema.

Método	$H_s \text{ (m)}$	$H \text{ (m)}$	$T_p \text{ (s)}$
sensor resistivo	0.046	0.027	1.011
sistema implementado	0.047	0.024	0.966

Tabla 5: Comparación de los parámetros H_s , H y T_p asociados a los datos capturados por el sistema implementado y el sensor resistivo en el ensayo **EN5**.

6.5. Consideraciones generales

6.5.1. Ángulo y posición de captura

Uno de los posibles errores a la hora de detectar la altura de la superficie libre se debe a la posición y ángulo de la cámara con respecto al nivel en reposo del fluido en el canal. Como se señaló anteriormente, los primeros ensayos se realizaron en la posición P1 (*Figura 40a*), en donde la lente de la cámara se encuentra casi a la altura de la superficie libre (en reposo). Cuando la pala mecánica comienza a producir el oleaje a medir, la superficie libre se desplaza por encima y por debajo del punto de reposo (como lo indican las

flechas de la figura). En el momento en el que la superficie libre se encuentra por debajo de la altura del lente de la cámara (flecha hacia abajo o $-h$), se produce un efecto no deseado que es la captura de la superficie del fluido a lo largo del ancho del canal, la cual genera reflejos e introduce ruido a la hora de detectar los contornos de la ola.

En la *Figura 54*, se muestra por encima la imagen capturada por la cámara de vídeo antes de ser procesada, y por debajo luego de ser procesada y aplicado el algoritmo de detección de contornos (*Canny*). Esta figura hace referencia a la altura de ola por encima del nivel de reposo donde se captura de forma correcta. En cambio, en la *Figura 55*, se puede observar el otro caso, en donde la superficie del fluido esta por debajo del lente de la cámara y observándose el ruido introducido en la detección de contornos por medio de una doble linea con trozos de bordes entre las mismas provenientes del reflejo mencionado anteriormente.

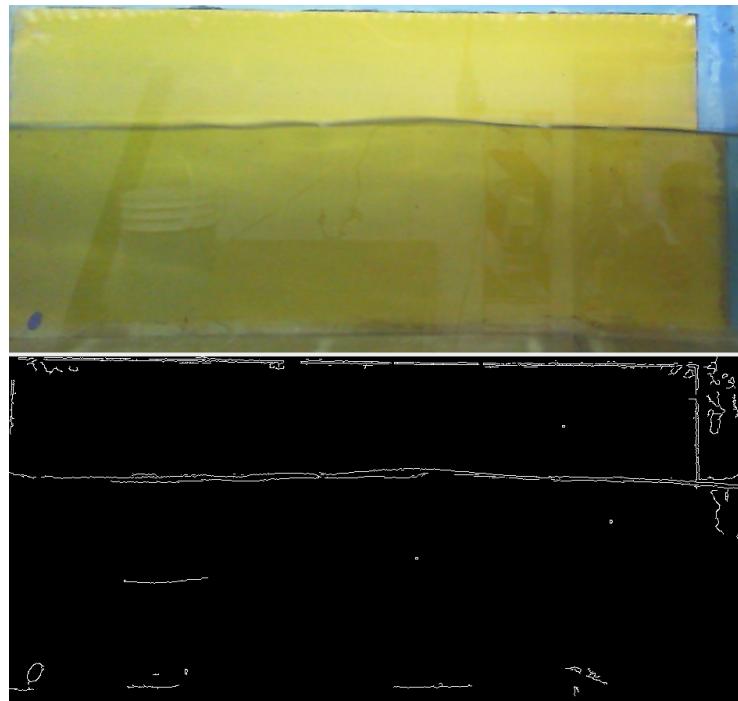


Figura 54: Errores introducidos por la posición y ángulo de la cámara a la hora de detectar los contornos de la superficie libre. Altura de la superficie libre por encima del lente de la cámara.

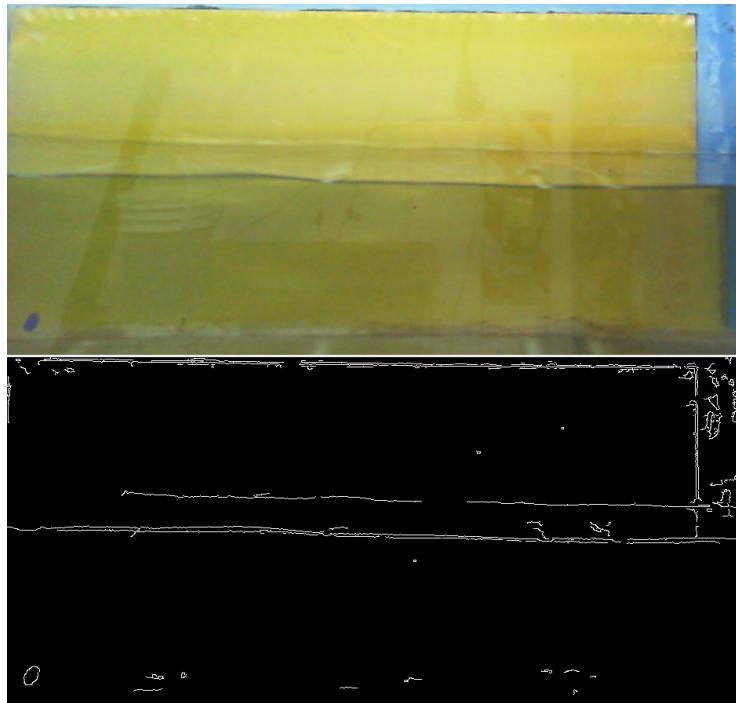


Figura 55: Errores introducidos por la posición y ángulo de la cámara a la hora de detectar los contornos de la superficie libre. Altura de la superficie libre por debajo del lente de la cámara.

Esta situación (*Figura 55*), en donde se detecta una “doble superficie” puede generar una falsa detección de la altura de ola (mas específicamente el *seno* o parte inferior de la ola) dando como resultado una altura menor a la real como se lo puede ver en la *Figura 56*. Se puede observar que las *crestas* de las olas son detectadas correctamente mientras que los *senos* no. En este caso se puede deducir que la posición del lente de la cámara se encuentra aproximadamente a la altura del borde superior de cada uno de los rectángulos azules, que es el punto donde los datos comienzan a diferenciarse (los obtenidos por el sistema -rojo- y los del sensor resistivo -verde-).

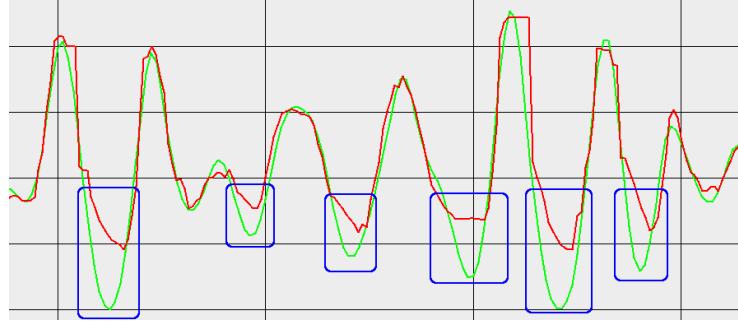
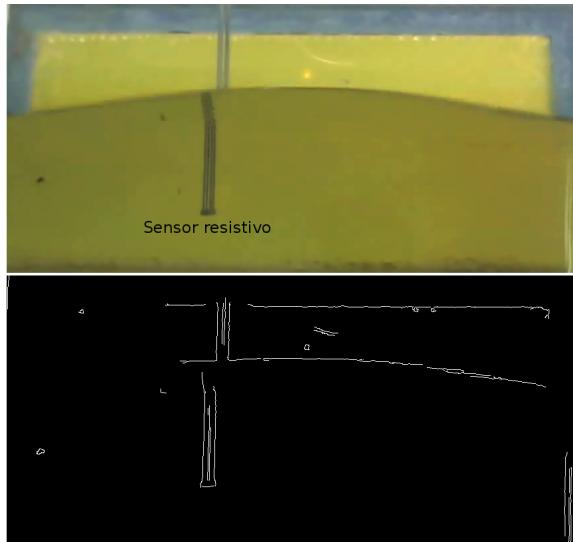


Figura 56: Errores (remarcados en azul) introducidos por la posición y ángulo de la cámara a la hora de detectar los contornos de la superficie libre cuando la altura de la superficie libre se encuentra por debajo del lente de la cámara. En rojo se muestra los datos capturados por el sistema implementado mientras que en verde los del sensor resistivo.

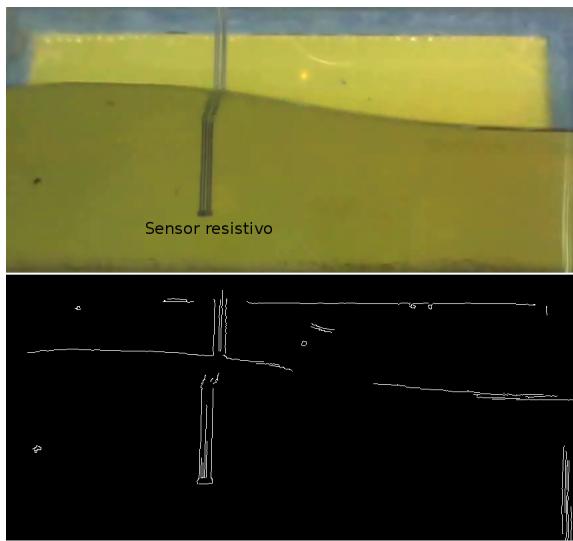
Claramente esto fue lo que sucedió en todos los ensayos realizados con la cámara ubicada en la posición P1 (**EN1**, **EN2**, **EN3**).

6.5.2. Velocidad de captura de la cámara

La velocidad de captura de la cámara juega un importante papel a la hora de poder detectar la superficie libre. En la *Figura 57* se muestran 2 imágenes (*Figuras 57a* y *57b*) capturando la misma ola en dos instantes de tiempos distintos y aplicándole el algoritmo de *Canny* (detección de contornos) en ambos casos con los mismos parámetros de configuración (iguales valores de umbrales en ambas figuras). En el primer instante (*Figura 57a*) se puede observar como a la izquierda del sensor resistivo el contorno de la ola se observa de una manera difusa provocando que el algoritmo de *Canny* falle detectando el contorno de la misma (imagen inferior). En el segundo instante (*Figura 57b*) se puede ver que el contorno de la ola ahora se encuentra difuso a la derecha del sensor resistivo como también lo muestra la imagen luego de aplicar el algoritmo de *Canny*. Esta situación se reproduce en cualquier parte (no solo a la derecha e izquierda del sensor) debido a que la ola se mueve a lo largo de todo el canal.



(a) Comparación del primer instante de captura de la ola.



(b) Comparación del segundo instante de captura de la ola.

Figura 57: Dos capturas realizadas del movimiento de la ola antes y después de aplicar el algoritmo de detección de contornos en donde se muestra las posibles fallas de detección de la superficie libre.

Si ahora observamos *Figura 58*, donde se muestra la imagen ya procesada

con el algoritmo de *Canny* y remarcando la zona de análisis en rojo (columnas de píxeles utilizadas por el algoritmo *Wave height analyzer* mencionado en el *capítulo 4.5*), ésta situación produce que el sistema, por la forma en la que fue implementado el algoritmo de la detección de altura (*Wave height analyzer*) y teniendo en cuenta la distancia (en píxeles) existente entre los dos puntos P1 y P2 (en la imagen representada por d), de como resultado alguna de las siguientes dos situaciones:

1. Si la distancia d entre P1 y P2 es mayor que V , esto es $d > V$ (recordando que V es la constante que indica la máxima variación de altura de ola aceptable en píxeles, mencionada junto con el algoritmo), y asumiendo que no se encontró ningún contorno al rededor del punto sobre el cual se está analizando (en este caso alrededor de P1), el algoritmo (y por ende el sistema) da como resultado la ultima altura de ola detectada (P1).
2. Si la distancia d entre P1 y P2 es menor o igual que V ($d \leq V$), entonces el sistema al no encontrar contornos a los costados del punto P1, devuelve P2.

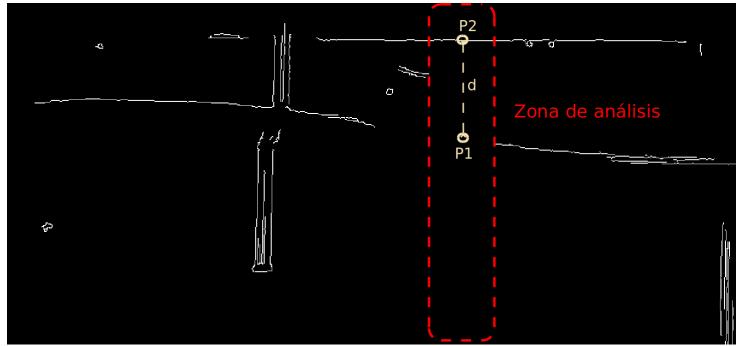


Figura 58: Posibles puntos detectados por el sistema (P1, P2, los cuales se encuentran a una distancia d) como altura de ola en caso de que el algoritmo de contornos no detecte la superficie libre.

Las consecuencias de la primera situación (1) antes descripta se las puede observar en la *Figura 59*, donde los datos devueltos por el sistema (altura de ola) están graficados en rojo. Se puede ver que existen zonas (en azul) donde la altura permanece constante, esto es debido a que al no encontrar el contorno de la superficie libre, el sistema devuelve la ultima altura de ola detectada.

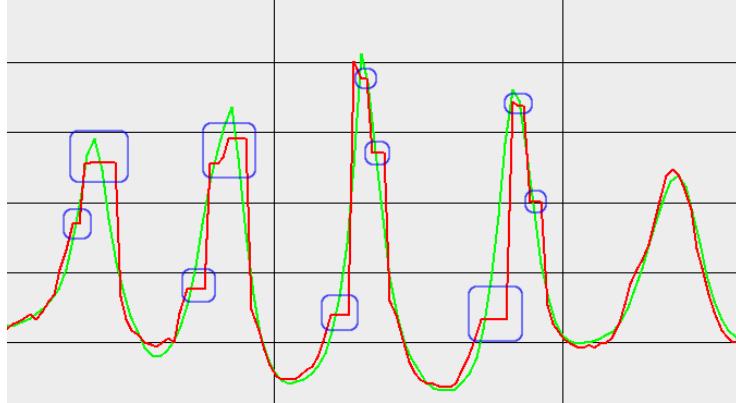


Figura 59: Errores (remarcados en azul) causados por la imposibilidad de detectar la superficie libre, en donde el sistema devuelve durante cierto tiempo la altura de ola detectada en la captura anterior. En rojo se observa los datos tomados por el sistema implementado y en verde se muestra los datos obtenidos por el sensor resistivo.

Por otro lado podemos analizar la segunda situación (2), la cual a su vez se puede dividir en dos situaciones mas: a) Cuando la nueva altura de ola detectada (el P2 mencionado anteriormente) es un contorno “fijo” que se mantiene a lo largo de todas las imágenes (por ejemplo un borde del canal el cual en cada imagen procesada siempre se detecta el mismo contorno); o b) cuando el nuevo P2 detectado pertenece algún contorno no “fijo” y por ende puede aparecer y desaparecer a lo largo de las imágenes procesadas (como por ejemplo reflejos de luces momentáneos sobre el vidrio del canal, o partículas en el fluido, etc).

Para el primer caso (a), se puede detectar casi al comienzo del ensayo, ya que en estos casos, si sucede, el algoritmo es incapaz de salir de ese contorno (por como fue implementado), y por ende, siempre se devuelve la misma altura de ola (que es el contorno fijo mencionado).

La segunda situación (b), aunque en general no sucede, es mas difícil de evitar. En la *Figura 60* se observa como hay un salto abrupto en la continuidad de los datos obtenidos, debido a la falla en la detección de la altura de ola y en donde se detectó un contorno “temporal” ubicado en el vértice del arco remarcado. Como se puede ver además, en la siguiente ola detectada (a la derecha) este error no sucede nuevamente.

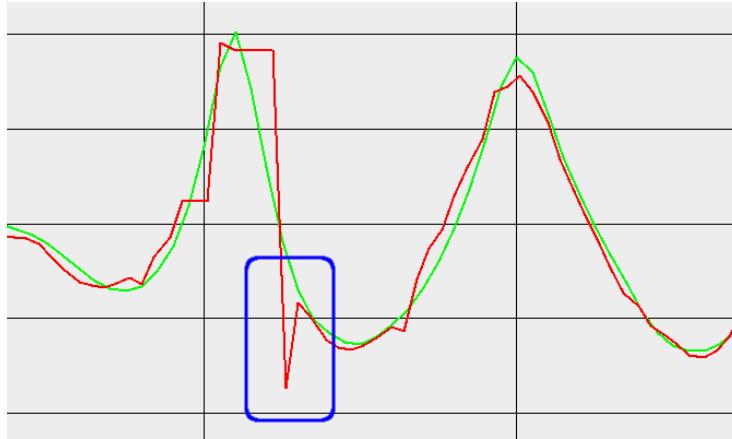


Figura 60: Error (remarcado en azul) causado por la imposibilidad de detectar la superficie libre, en donde el sistema devuelve la altura mas cercana al ultimo punto detectado. En rojo se observa los datos tomados por el sistema implementado y en verde los obtenidos por el sensor resistivo.

Claramente ambos casos (1 y 2), generan grandes diferencias en los valores capturados por el sistema implementado en comparación con los obtenidos por el sensor resistivo. Una posible situación que solventa estos errores es la disminución de los valores umbrales utilizados en el algoritmo de *Canny*. En la *Figura 61*, (la cual es la misma que la imagen superior de la *Figura 57b* luego de aplicar el algoritmo de *Canny*) se puede ver que la zona (remarcada en rojo) que anteriormente no se detectaba la superficie libre en este caso si lo hace, pero con una especie de interferencia (no es una linea continua como se lo observa a la izquierda del sensor en la imagen).



Figura 61: Interferencia en la detección de la superficie libre generada a causa de la utilización de valores bajos en los umbrales del algoritmo de *Canny*.

Esta interferencia varia mucho a medida que la ola se traslada generando un cambio brusco en los valores de la altura de la superficie libre detectada por el sistema. En la *Figura 62* se puede observar el resultado de tal situación.

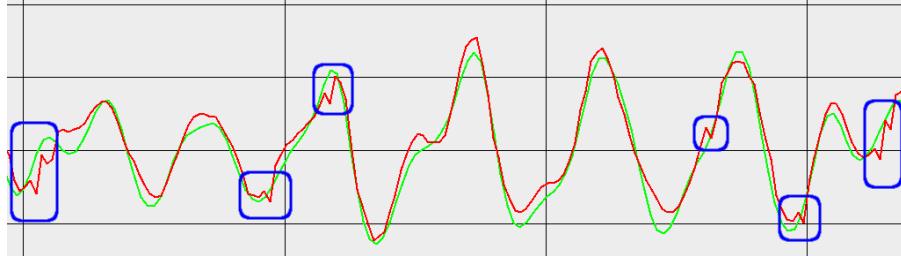


Figura 62: Errores (remarcados en azul) causado por los bajos valores de los umbrales del algoritmo de Canny utilizados para solventar el problema de detección de la superficie libre en movimiento. En rojo se observa los datos tomados por el sistema implementado y en verde los obtenidos por el sensor resistivo.

6.5.3. Superficie libre

Otro inconveniente que se pudo observar es el hecho de la doble linea (doble contorno) que se detecta al capturar la altura de la superficie libre como lo indica la *Figura 63*. En la imagen superior se muestra la imagen capturada por la cámara de vídeo, en la inferior el resultado de aplicar sobre la primera el algoritmo de detección de contornos (*Canny*). Este tipo de situación genera en algunos casos (cuando el movimiento de la ola es muy rápido/brusco) un error similar al planteado en el subcapítulo anterior, en donde los 2 bordes pueden ser no detectados de forma intercalada (primero se detecta uno y luego el otro) generando pequeños saltos (que en este caso son de aproximadamente 4 mm) en los datos capturados y por ende en la altura de ola resultado. Probablemente esto pueda ser resuelto utilizando distintos tipos de luces que eviten resaltar el borde de la superficie de esa forma. Otra posible solución (a investigar) podría ser colorear todo el líquido del canal con un tono opaco de forma tal que no se alcance apreciar la linea divisoria entre el agua y el aire.

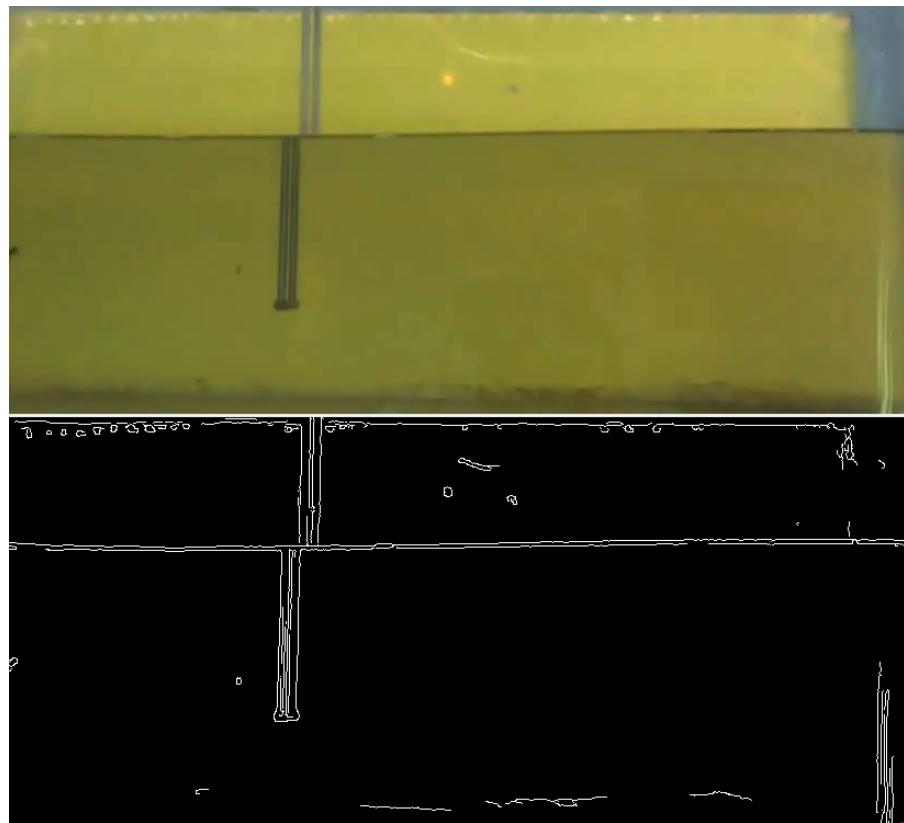


Figura 63: Doble contorno detectado de la superficie libre.

7. Conclusiones

En este trabajo se propuso desarrollar un sistema capaz de capturar la altura de la ola en un canal 2D de laboratorio por medio de una cámara web y en tiempo real. Como se pudo observar en los análisis de los resultados del capítulo anterior, la precisión del sistema esta ligada y fuertemente afectada por factores externos como la luz utilizada (intentando buscar siempre aquella que genere mayor contraste entre el fluido y la superficie y al mismo tiempo no genere reflejos sobre el vidrio del canal), la cantidad de partículas tanto en el fluido como en las paredes del canal, la posición y ángulo de la cámara la cual cumple un papel crucial, y además los valores de configuración para el algoritmo de detección de contornos, el cual varia dependiendo de los factores recientemente mencionados.

Se puede decir que el sistema logra capturar y analizar la altura de ola en tiempo real y a una velocidad superior a la mínima esperada (alcanzando 25 Hz) mostrando al mismo tiempo el proceso del ensayo realizado y los datos obtenidos (lo que quiere decir que puede correrse la interfaz gráfica en simultáneo con el proceso de captura). Aún así no se logró alcanzar una precisión milimétrica para poder realizar mediciones que requieran un error menor a 0.55 cm, donde las causas se debieron tanto por motivos de hardware (velocidad de captura), software (post-procesamiento de datos, utilización de otros algoritmos) como de factores externos (luminosidad, partículas en el fluido y en las paredes del canal, etc). Por otro lado se pudieron realizar mediciones en donde los errores de los valores de importancia como H_s y T_p se encuentran aproximadamente en el orden de los 0.1 cm y 0.009 segundos respectivamente, lo cual habilitaría al sistema para ser utilizado en diversas pruebas de ingeniería en laboratorio.

Se considera además que los algoritmos seleccionados para la implementación del sistema cumplieron con su finalidad de una manera satisfactoria tanto en calidad como velocidad, brindando asimismo al sistema una forma fácil de configurarlo.

Por otro lado se considera altamente viable la posibilidad de aumentar la precisión del sistema por medio de distintos algoritmos en la fase final que complementen los posibles errores introducidos por la velocidad de captura de la cámara, como así también en una etapa posterior a la detección de los datos (post-procesamiento) en la que se filtren y corrijan aquellos valores que se consideren erróneos.

Se logró además armar una interfaz gráfica que permitiese al usuario del sistema una fácil y rápida configuración a la hora de realizar las mediciones, dándole mucha flexibilidad sobre los parámetros utilizados en los algoritmos

de las distintas etapas.

Fuera del sistema implementado para detectar la altura de la superficie libre, se logró desarrollar además un *framework* el cual permitirá crear futuras aplicaciones (que brinden la oportunidad de medir otras variables hidrodinámicas) de una forma más fácil y rápida, dando también la opción de utilizar el GPU como unidad de procesamiento.

8. Trabajos futuros

Algunos de los trabajos futuros que se pueden realizar tomando como base el actual son:

- Desarrollar nuevos *IPS* (*Image Processing System*) que implementen otras funcionalidades para medir distintas variables hidrodinámicas utilizando el *framework* desarrollado, como por ejemplo detectar y analizar la forma de la rotura de la ola, realizar tracking de partículas en el fluido, etc.
- Desarrollar un algoritmo que permita detectar la altura de la ola en cada columna de píxeles de la imagen capturada por la vídeo cámara por medio de la paralelización y usando el GPU como unidad de procesamiento (con el objetivo de mantener la captura en tiempo real), logrando de esta forma determinar la altura de la superficie libre con mucho mayor precisión.
- Probar y comparar diferentes algoritmos utilizados en las distintas etapas con el fin de aumentar la precisión de captura de datos y disminuir la detección de ruido al máximo (intentar lograr que el sistema no sea tan sensible al entorno como los cambios de luces, partículas en la zona de captura, etc). Y realizar en una segunda etapa el post procesamiento de los datos obtenidos con el objetivo de eliminar por medio de distintas técnicas (uso de splines por ejemplo) aquellos datos erróneos y corregir al mismo tiempo aquellos que se consideren “desfasados”.

Referencias

- [1] www.horusvideo.com
- [2] Robert M. Sorensen, “Basic Coastal Engineering“ (Third Edition), Department of Civil and Environmental Engineering, Lehigh University, Bethlehem, Pennsylvania, 2006.
- [3] Robert G. Dean, Robert A. Dalrymple, “Water Wave Mechanics for Engineers and Scientists”, Prentice-Hall, Inc. Englewood Cliffs, New Jersey 07632.
- [4] G.I.O.C., “Documento de referencia, Volumen 1: Dinámicas”, www.smc.unican.es, Universidad de Cantabria, 2000.
- [5] Ricardo V. Petroni, “Hidráulica marítima y de estuarios”, Dunken, Buenos Aires, 2005.
- [6] Perez Muñoz, Juan C. “Optimización no lineal y calibración de cámaras fotográficas”. Medellín, 2009.
- [7] Medellín, 2009.P.D.M. Brady , M. Boutounet and S. Beecham. “Free Surface Monitoring Using Image Processing”. University of Technology Sydney, NSW, 2007.
- [8] <http://hdl.handle.net/2099.1/5945>, Cap. 5. Procesado de Imágenes.
- [9] Rong Zhang . “Image Rectification: Remove Projective and Affine Distortions”.
- [10] Ehsan Nadernejad, Sara Sharifzadeh and Hamid Hassanpour. “Edge Detection Techniques: Evaluations and Comparisons”. Applied Mathematical Sciences, Vol. 2, 2008.
- [11] Ing. Francisco J. Sánchez R., “Medición y Análisis de las Variaciones en el Nivel de un Modelo Físico Empleando Imágenes”, Universidad Autónoma Metropolitana Unidad Azcapotzalco, 2010.
- [12] S. Barreto Melo. “Transformaciones geométricas sobre imágenes digitales”. Universidad Distrital Francisco José de Caldas.
- [13] Gary Bradski, Adrian Kaehler, “Learning OpenCV: Computer Vision with the OpenCV Library”, O'Reilly, September 2008.
- [14] Christopher R. Wren, <http://alumni.media.mit.edu/~cwren/interpolator/>

- [15] "UNIVERSITY OF EDIMBURGH SCHOOL OF INFORMATIC".
<http://www.inf.ed.ac.uk/>
- [16] Canny Edge Detection. 09gr820, March 23, 2009
- [17] Gamma, Helm, Johnson, Vlissides. "Design Patterns: Elements of Reusable Object Oriented Software". 1994
- [18] <http://www.seafriends.org.nz/oceano/waves.htm>
- [19] <http://code.google.com/p/hvdrt/>