SBIN: A Stereo Disparity Estimation Network using Binary Convolutions

Cristhian A. Aguilera

***Abstract*—Although the current advances on convolutional networks are outstanding, they mainly depend on extensive computational power, limiting the areas of applications. The latter applies for stereo disparity estimation, where current solutions can barely run on embedded devices. This work shows that it is possible to binarize an end-to-end stereo disparity network, which can be considered a step towards lightweight and potentially faster disparity estimation networks. This work shows the validity of the proposed approach through experimentation in two well-known datasets, sceneflow and kitti2012. The results show that a binary disparity model is possible but at the cost of performance. An EPE of 5.14 and 2.09 is achieved in sceneflow and kitti2012 accordingly.**

***Index Terms*—Stereo vision, Computer vision, Embedded devices, Binary networks.**

I. Introducción

La estimación de distancia es esencial para múltiples tareas en robótica y visión por computador. En robótica, permite a los robots interactuar con objetos y entornos dinámicos [1], y en visión por computador, permite generar información más precisa, para la detección y localización de objetos en el espacio [2].

Aunque existen múltiples sensores para la estimación de distancia, uno de los más populares y de bajo coste, son las cámaras estéreo, que no solo permiten estimar la distancia a objetos, sino también proveer información visual relevante para tareas relacionadas.

Una cámara estéreo, es en realidad un sistema de dos cámaras, separadas a una distancia fija llamada baseline b. Imágenes de ambas cámaras son rectificadas y alineadas, obteniendo la distancia a objetos, a partir de la búsqueda de correspondencia entre pixeles de ambas imágenes. La distancia entre un pixel de una imagen, y el correspondiente pixel de la otra imagen (disparidad), es inversamente proporcional a la distancia del objeto. Gracias a un proceso de calibración previo, la búsqueda de correspondencia es unidimensional, realizando la búsqueda solo a través del eje x; recorriendo todas las filas de las imágenes.

La estimación de distancia, utilizando cámaras estéreo, ha sido un tema recurrente en la literatura durante décadas [3]. Los últimos avances, principalmente basados en redes neuronales convolucionales (CNN), han logrado incrementar la capacidad de calcular la disparidad, logrando precisiones inferiores a un pixel (por ejemplo, [4]). Aunque los avances han sido significativos, estos avances se soportan en un alto poder computacional, que no es accesible para todo tipo de aplicaciones, sobre todo en las que existen limitaciones de tamaño o consumo energético; por ejemplo, aplicaciones que utilizan sistemas embebidos. Es por esto que soluciones con menor requerimientos computacionales y utilizables en un mayor espectro de situaciones son necesarias.

La búsqueda de soluciones, basadas en CNN, que requieran menor poder de cómputo, no es única al problema estéreo, y es un tema activo en la comunidad científica. En la actualidad, existen varias formas de enfrentar este problema, por ejemplo, podando redes [5], o a través de la cuantificación de los pesos de las redes. En el caso de la cuantificación de los pesos, va desde los ya comunes 16-bit, hasta el caso más extremo, 1-bit, es decir, redes convolucionales binarias (RCB).

Las RCB son redes, donde los pesos son binarios, volviendo las convoluciones más sencillas de resolver computacionalmente, a través de operaciones XNOR y *pop count* [6]. También, al ser los pesos binarios, las RCB ocupan alrededor de 32 veces menos memoria que una con precisión flotante (32-bit). Aunque las RCB han demostrado ser efectivas en tareas de clasificación (por ejemplo, [5]), su precisión y utilidad en otras áreas, como en la estimación estéreo, han sido vagamente estudiados. Si bien trabajos como los de Chen et al. [7], [8], utilizan redes binarias para estimar la disparidad, estas solo se limitan al modulo de extracción de características, y no al sistema de estimación completo, lo cual es considerado estándar en las soluciones actuales.

Este trabajo busca analizar el uso de RCB en la estimación de disparidad estéreo, En particular, entender los pasos necesarios para la binarización de una red estéreo de principio a fin (*end-to-end*), definiendo lo cambios más importantes a tomar en cuenta, y también la precisión final que se puede lograr.

Los principales aportes de este trabajo son los siguientes:

- A través de un análisis experimental, se demuestra que es posible adaptar una red estéreo del tipo *end-to-end*, para funcionar con convoluciones binarias.
- Un análisis de las partes más relevante del modelo y su influencia en la binarización.
- Ya que el esquema de solución de las redes estéreo es relativamente genérico, el método de binarización puede ser utilizado en otros modelos *end-to-end*, donde se utilice *soft argmin* como capa final.
- Todo el código es libre y disponible a través de www.github.com/ngunsu/SBIN

II. Trabajo Relacionado

A. *Estimación de Disparidad*

La estructura actual de los métodos de estimación de disparidad, es similar a los propuestos décadas atrás [3]. Métodos

C. A. Aguilera, Departamento de Ciencias de la Ingeniería, Universidad de Los Lagos, Chile (cristhian.aguilera@ulagos.cl).

basados en técnicas de aprendizaje, reemplazan a módulos diseñados cuidadosamente, como el cómputo de volumen de costo, a través del uso de datos etiquetados, mostrando avances significativos en sus resultados (por ejemplo, [4], [9], [10]). En un comienzo, los intentos se enfocaron en reemplazar solo un módulo. Por ejemplo, en [11], los autores proponen reemplazar algoritmos tradicionales de correspondencias, por un modelo basado en aprendizaje automático. Dicha propuesta, mejora considerablemente la precisión de la solución, pero también aumenta considerablemente el tiempo de ejecución. Propuestas posteriores, continuaron con esta metodología, y ahora han evolucionado a soluciones de principio a fin, que reemplazan completamente los módulos diseñados a mano. GC-Nets [10] es vital en este desarrollo, ya que propone el primer modelo de principio a fin, introduciendo elementos críticos para futuros desarrollos, como lo es la función *soft argmin*. *Soft argmin* es utilizada en múltiples trabajos posteriores, por ejemplo, [4], [12], [13].

Aunque los avances en estimación estéreo son indudables, la mayoría se soportan en capacidades de cómputo extensivas, que no permiten una inferencia en tiempo real en dispositivos pequeños, de bajo costo y bajo consumo energético; como los utilizados en robótica. En este sentido, en Anytime Stereo [12], se propone un método que permite compensar entre precisión y tiempo de ejecución, logrando ejecutar en tiempo real su solución en un sistema embebido del tipo Jetson TX2. De manera similar, Aguilera et al [13], propone un modelo aún más rápido, logrando tiempos de ejecución en tiempo real en incluso dispositivos más pequeños, como la Jetson Nano.

Las soluciones previamente mencionadas, consideran tiempo real, como la inferencia de imágenes estéreo de tamaño 720p, en menos de 33 ms. Estos tamaños son pequeños para las capacidades de hardware actuales, por lo que técnicas más rápidas y eficientes son necesarias. Es por esto que en este artículo, se propone un primer avance en este camino, mostrando que es posible cuantificar una red estéreo de manera binaria, para a futuro tener soluciones más rápidas y pequeñas, que permitan trabajar con imágenes de mayor tamaño y a una mayor velocidad.

B. Redes Convolucionales Binarias

Una red convolucional binaria, es un caso extremo de cuantificación, donde los pesos y activaciones solo tienen dos posibles valores; 1-bit. Una RCB es eficiente en términos de memoria utilizada, requiriendo alrededor de 32 veces menos memoria que una red con pesos flotantes; 32-bit. Ya que los pesos y activaciones son binarias, estás teóricamente, también son eficientes desde el punto de vista computacional, simplificando las convoluciones a operaciones XNOR y *pop count* [14].

Los primeros esfuerzos en la binarización de redes se enfocaron en los pesos, manteniendo las activaciones en precisión flotante (por ejemplo, [15]). Lo anterior, con el objetivo de reducir los requerimientos de memoria para almacenar las redes, y mejorar también los tiempos de inferencia, ya que las convoluciones eran más simple de resolver computacionalmente; mayoritariamente en CPUs. Trabajos posteriores, como [6], [14] agregaron las activaciones al proceso de binarización, proponiendo una implementación rápida, a través de la utilización de operaciones XNOR y *pop count*. De manera particular, XNOR-Net [6] avanzó un paso más lejos y con su propuesta, demostró que es posible alcanzar altas precisiones en conjunto de datos tradicionales; hasta ese momento todos los intentos usaban conjunto de datos pequeños para realizar sus evaluaciones. Trabajos posteriores han sido mayoritariamente mejoras incrementales de los trabajos anteriores, reduciendo substancialmente la distancia entre redes binarias y sus contra partes flotantes. Por ejemplo, el trabajo de [16], se mejora la forma de computar los pesos y se propone una nueva función de perdida, para reducir el número de cambios de signos en el proceso de actualización de pesos. Redes más modernas, agregan conexiones residuales, que han demostrado mejorar la precisión en tareas de clasificación [17]. Conexiones residuales ayudan a los gradientes de una red binaria. En este trabajo se utilizan conexiones residuales, en cada caso posible, siguiendo las sugerencias de [17]–[19].

Es importante mencionar que, aunque la mayoría de los trabajos previamente descritos, dicen ser redes convolucionales binarias, en estricto rigor no lo son; al menos no completamente. En [6] llegan a la conclusión que la primera y la ultima capa de una red binaria, deben ser flotantes para lograr una alta precisión. Esta recomendación es seguida por la mayoría de las implementaciones actuales, incluidas las de este artículo.

Otro importante tópico son estrategias de aprendizaje. En general entrenar una red binaria desde cero es difícil, y múltiples estrategias han sido recomendadas para facilitar su entrenamiento. Dentro de esas estrategias, dos han demostrado ser de utilidad en diferentes problemas. Primero, en el trabajo de [20] se propone un entrenamiento por partes, empezando por la capa más alta, hasta la más baja. En relación a este trabajo, en [21], siguen una estrategia similar, pero agregando un esquema de destilación del conocimiento, utilizando una red flotante como profesor de la red binaria (estudiante).

En cuanto a estimación de disparidad utilizando RCB, no existen muchos casos de éxito en la literatura. Chen et al. en [7], [8] proponen la binarización de la etapa de extracción de características de [11], para generar una red de estimación de disparidad en base a distancias. Si bien la propuesta es novedosa, esta es limitada, ya que no considera una etapa de post-procesamiento basada en técnicas de aprendizaje automático (modelos *end-to-end*). Es por esto que, en este trabajo se estudia como aplicar la binarización a modelos del tipo *end-to-end*, que utilizan una función *soft argmin* para estimar la disparidad de imágenes estéreo. Los modelos del tipo *end-to-end* tienden a ser preferidos en la literatura, ya que pueden integrar contexto visual a la estimación de disparidad, lo que es particularmente beneficioso en zonas de texturas homogéneas.

III. Modelo Base

Usualmente, una red binaria se construye a partir de una contraparte de precisión flotante. Dicho de otra manera, se toma un modelo ya funcional y se adapta para su operación de tipo binaria. Lo anterior no quiere decir que, los pesos entrenados de una red de precisión flotante funcionen en su

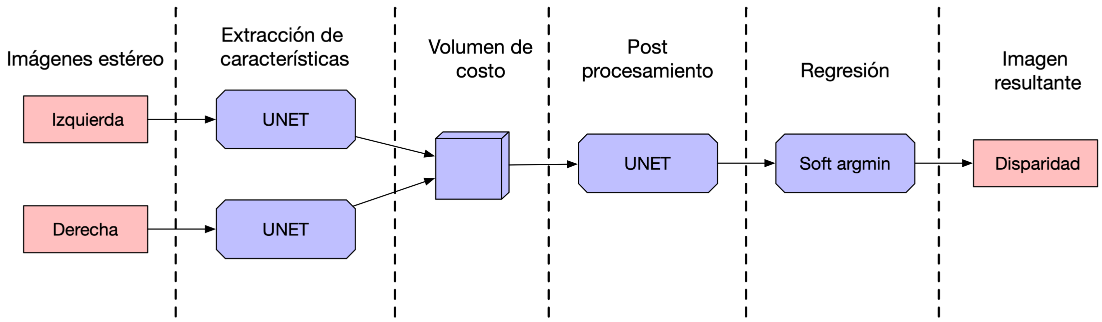


Fig. 1. Modelo base *Festereo*: [13].

versión binaria, solo que es un buen punto de partida. En este artículo se utiliza como base *Festereo* [13] (ver Figura 1), el cual es un modelo estándar de estimación de disparidad, que fue diseñada para funcionar en dispositivos embebidos de bajo coste. Se seleccionó *Festereo* por las siguientes razones:

- Los pasos de *Festereo* son consistentes con múltiples otros modelos, incluyendo la fase de regresión. Lo anterior, con el objetivo de obtener conclusiones, que sean de potencial utilidad para otros métodos de principio a fin de estimación de disparidad.
- Todas las convoluciones en *Festereo* son 2D. La mayoría de los modelos de estimación de disparidad usa convoluciones 3D, en su fase de post-procesamiento. La binarización e implementación de convoluciones 3D no ha sido estudiada rigurosamente en la literatura, por lo que este trabajo solo considera binarización de convoluciones 2D.
- Familiaridad y conocimiento en relación a *Festereo*.

A. *Descripción del Modelo*

El modelo cuenta con cuatro etapas:

1. Extracción de características: Una red siamesa, del tipo U-Net [22] es utilizada para extraer características de ambas imágenes de entrada; imágenes de las cámaras izquierda y derecha. La salida de esta red, es un vector de características de tamaño $H/R \times W/R \times F$, donde H, es la altura de la imagen en pixels, W el ancho de la imagen en pixels, R un factor de reducción de tamaño, para un procesamiento más rápido, y F el tamaño del vector de características para cada pixel.
2. Volumen de costo: Ya que la búsqueda de disparidad es unidimensional, en esta etapa se generan todos los posibles candidatos para cada pixel. Asumiendo una disparidad máxima d de pixeles, entonces para cada pixel de la imagen izquierda, todos sus posibles candidatos son concatenados, obteniendo una matriz de salida de tamaño $H/R \times W/R \times F \times d/R$. Este proceso no involucra redes neuronales, y es implementado directamente en CUDA, para su rápido cómputo. El valor típico de d es 128.
3. Post-procesamiento: Obtener la disparidad a través de la distancia mínima entre vectores no es recomendada, por lo que esta fase filtra y mejora el volumen de costo, antes de la regresión. Se utiliza una red U-Net como en la extracción de características, pero de menor tamaño.
4. Regresión: Una capa *soft argmin* [10] es utilizada para obtener la disparidad de cada pixel de la imagen izquierda. En esta etapa, también se realiza una interpolación, para obtener la disparidad en el tamaño original de la imagen de entrada.

Para obtener mayor detalle de la implementación, ver artículo original [13].

IV. RED BINARIA

En esta sección se describe el procedimiento de binarización utilizado en este trabajo, y los cambios realizados al modelo base, con tal de obtener una RCB de estimación de disparidad.

A. *Binarización*

El entrenamiento de una red binaria, requiere del uso de pesos y activaciones de precisión flotante. Durante el proceso de actualización de pesos, usado por los métodos de optimización como *stochastic gradient descent*, es necesario realizar pequeñas actualizaciones, que son imposibles de lograr solo con valores binarios [23].

En Courbariaux et al. [14] se propuso una forma de lidiar con el problema descrito anteriormente. La idea es utilizar valores flotantes de pesos y activaciones, durante todo el procedimiento de entrenamiento. En cada paso hacia adelante, los valores en precisión flotante de los pesos son convertidos a binarios; los valores de precisión flotante son almacenados. En cada paso hacia atrás (*backpropagation*), los valores son derivados desde la versión binaria a la flotante, y los valores finalmente son actualizados en la versión flotante de los pesos. Este método de entrenar redes es considerado estándar y es el que utilizamos en este trabajo. Es importante mencionar que, cuando la red ya esta entrenada, este procedimiento se vuelve innecesario, y que los valores binarios pueden ser utilizados directamente.

En este artículo seguimos el proceso de binarización de pesos w y activaciones a de [17]. Los pesos son convertidos de acuerdo a la Ecuación 1 y las activaciones de acuerdo a la ecuación 2. w_r y a_r, corresponden a la versión de precisión flotante de los pesos y activaciones, y w_b y a_b, a su contraparte binaria.

$$w_b = Sign(w_r) = \begin{cases} -1, & \text{si } w_r < 0 \\ 1, & \text{otro caso} \end{cases} \tag{1}$$

$$a_b = Sign(a_r) = \begin{cases} -1, & \text{si } a_r < 0 \\ 1, & \text{otro caso} \end{cases} \tag{2}$$

En la implementación de este articulo, se evito utilizar el termino de ganancia introducido en [17]. Después de varios experimentos, no existieron diferencias significativas en su uso u omisión.

B. Cambios a Modelo Base

Se realizaron tres modificaciones al modelo base, con tal de disminuir la brecha de precisión entre la versión flotante y la resultante binaria. Se modificaron los componentes de la red U-Net, que son utilizados para la extracción de características y para el proceso de post-procesamiento. En particular se realizaron las siguientes modificaciones:

1. Se cambio las activaciones no-lineales ReLU [24], por activaciones PReLU [25].
2. En cada nivel de U-Net, se agregaron conexiones residuales a las convoluciones. Cuando el tamaño inicial es diferente al final, se utilizaron conexiones residuales con convoluciones 1×1 de precisión flotante, como en [19].
3. Se cambio la estrategia de reducción de tamaño de *Max-pooling* a *Average Max-pooling*

La Figura 2 muestra visualmente los cambios 1 y 2 realizados.

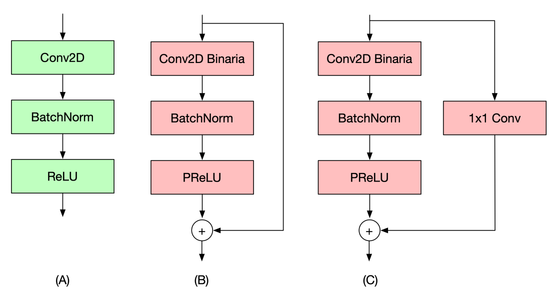


Fig. 2. Cambios aplicados a U-Net. En (A), convoluciones originales del articulo [13]. En (B) y (C) cambios realizados para la binarización de la red; (C) se utiliza cuando el tamaño inicial difiere del final en la suma residual.

V. Experimentos

A. Configuración

Con tal de realizar una comparación justa, se siguió un procedimiento similar de entrenamiento y pruebas, al utilizado con la red base [13]. Es decir, se entreno y evaluó inicialmente en el conjunto de datos sintéticos sceneflow [26], y luego se volvió a entrenar y evaluar en el conjunto de datos reales kitti2012 [27]. Para el caso de los modelos entrenados en kitti2012, no se utilizaron los modelos entrenados en sceneflow como base; a diferencia de lo ocurrido en [13], el *fine-tuning* resulto perjudicial en los experimentos con RCB.

La implementación binaria, al igual que su contraparte flotante, utiliza la función de perdida *smooth L1* de [12]. La extracción de características, reduce el tamaño original de la imagen ocho veces ($R = 8$). Se utiliza Adam [28] como el optimizador, con una velocidad de aprendizaje de 1E-3, y con valores beta de 0.9 y 0.999. Se utiliza tamaño de *batch* igual a 6, y el planificador *StepLR*. La implementación se realizó en Pytorch 1.6, utilizando Pytorch Lighting 1.0.8 [29].

Se utilizó una GPU RTX2080 TI para el entrenamiento y evaluación de todos los experimentos.

B. Métricas

En este trabajo, se evalúa el rendimiento de la solución propuesta utilizando dos métricas:

- Error de punto final (EPE)
- Porcentaje de error en pixeles (ERR)

EPE es el error de disparidad promedio (Ecuación 3), donde L_d es el resultado de la disparidad en la imagen izquierda y GT_d son las etiquetas de disparidad (*groundtruth*). (x, y) son las posiciones validas en la imagen, donde existe etiquetas.

$$EPE = \frac{\sum_{x,y} L_d(x,y) - GT_d(x,y)}{\#GT_d} \tag{3}$$

ERR es porcentaje de error, dado un umbral de pixeles (Ecuación 5), donde L_d es el resultado de la disparidad en la imagen izquierda, GT_d son las etiquetas de disparidad (*groundtruth*) y T es un umbral definido; usualmente T toma valores entre 2 y 5. (x, y) son las posiciones validas en la imagen, donde existe etiquetas.

$$diff(z) = \begin{cases} 1, & \text{if } |z| > T \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$ERR_T = \frac{\sum_{x,y} diff(L_d(x,y) - GT_d(x,y))}{\#GT_d} \tag{5}$$

Ambas métricas son estándar y han sido utilizadas en numerosos artículos (por ejemplo, [10], [12], [13]).

C. Resultados

Los resultados son presentados en el Cuadro I y II. Para ambos casos se analizaron tres modelos, en dos precisiones diferentes: flotante y binaria; la intención es ver si hay relación entre ambos casos. Los tres modelos son: Festereo [13], que corresponde al modelo base, Festereo RES, que corresponde al modelo base, pero con conexiones residuales, y finalmente la propuesta detallada en la sección IV. Adicionalmente, se evaluó Anynet [12] en kitti2012, para comparar la precisión del modelo binario con otro diferente al base[1].

En el caso del conjunto de datos sceneflow, Cuadro I, es posible observar que la propuesta de este trabajo, en precisión flotante, presenta un comportamiento similar al modelo base. Esta similitud desaparece en el caso binario, donde el único modelo que puede lograr un desempeño similar, al con precisión flotante, es el propuesto en este trabajo. Lo anterior valida los cambios propuestos, ya que sin estos, la diferencia es significativa. Adicionalmente, es posible observar a través de Festereo RES, que no basta solo con agregar conexiones residuales para la binarización de una red de estimación de disparidad.

[1]Los modelos evaluados, corresponden a modelos diseñados para funcionar en dispositivos embebidos.

TABLA I
RESULTADOS EN EL CONJUNTO DE DATOS SCENEFLOW. MENOR ES MEJOR.

Modelo	Precisión	EPE	ERR_2	ERR_3	ERR_4	ERR_5
Festereo [13]	flotante	3.9170	**0.2081**	**0.1737**	**0.1505**	**0.1329**
Festereo RES	flotante	4.2853	0.2263	0.1887	0.1634	0.1443
Propuesta	flotante	**3.9076**	0.2110	0.1758	0.1520	0.1340
Festereo	Binaria	23.3545	0.9242	0.8905	0.8555	0.8212
Festereo RES	Binaria	22.7121	0.9232	0.8890	0.8525	0.8176
Propuesta	Binaria	**5.1494**	**0.2110**	**0.1758**	**0.1520**	**0.1340**

TABLA II
RESULTADOS EN EL CONJUNTO DE DATOS KITTI2012. MENOR ES MEJOR.

Modelo	Precision	EPE	ERR_2	ERR_3	ERR_4	ERR_5
Anynet [12]	flotante	———	0.2651	0.1793	0.1238	0.0906
Festereo [13]	flotante	1.7450	0.1591	0.1089	0.0800	0.0614
Festereo RES	flotante	1.8240	0.1759	0.1194	0.0859	0.0654
Propuesta	flotante	**1.6536**	**0.1524**	**0.1039**	**0.0751**	**0.0575**
Festereo	Binaria	9.5500	0.7525	0.6883	0.66092	0.5423
Festereo RES	Binaria	7.1200	0.6964	0.6159	0.5175	0.4349
Propuesta	Binaria	**2.0979**	**0.2022**	**0.1370**	**0.1007**	**0.0785**

En el caso del conjunto de datos kitti2012, Cuadro II, los resultantes son similares al caso anterior. Una sorpresa inesperada, es que la propuesta en precisión flotante, tuvo mejores resultados que la red base.

Si bien existe una perdida de precisión debido a la binarización de la red, esta es de alrededor de un 25 %, lo que es acorde a los resultados obtenidos en otras áreas (por ejemplo, [21]). Esta perdida puede ser significativa para soluciones que requieran precisiones milimétricas, pero no así para otras que no requieran una precisión tan detallada, por ejemplo, en soluciones de robótica móvil para evitar obstáculos[2].

Visualmente, los resultados muestran una degradación en los detalles de los objetos. Por ejemplo, en la Figura 3 se puede observar, que si bien el mapa de disparidad da cuenta de los objetos en la imagen, estos carecen de detalles finos. De manera similar, los resultados visuales en el conjunto de datos kitti2012 (Figura 4) también carecen de detalles finos. Lo anterior, no es solo a causa de binarización de la red, si no también, debido a la estrategia de reducción de tamaño y posterior interpolación de los resultados; esto también sucede en [12], [13].

D. Análisis de Componentes

En esta sección, se describe algunas de las decisiones de diseño, que se llevaron a cabo para la generación del modelo propuesto.

D1. Pooling: Aunque *Max-pooling*, es probablemente, una de las técnicas de reducción de tamaño más utilizadas, esta no es necesariamente adecuada para las RCB. La salida de una convolución binaria, es de precisión entera (suma), con posibles valores positivos y negativos. Los valores negativos son causados por pesos con el valor -1, y los positivos por pesos con valor +1; ya que la red es binaria. En una RCB, los valores máximos tenderán a ser principalmente positivos por lo que, *Max-pooling* tenderá a dar preferencia a pesos con signo positivo, lo que puede ser perjudicial durante el entrenamiento. Por la razón anterior, en este trabajo se decidió utilizar *Average Max-pooling*, que no preferencia valores positivos por sobre valores negativos.

Nuestra propuesta, utilizando *Max-pooling* obtiene un EPE de 5.7 en sceneflow, en contraste con un EPE de 5.15, cuando *Average Max-pooling* es utilizado.

D2. Activaciones no Lineales: Uno de los cambios necesarios para poder entrenar una RCB, es cambiar todas las activaciones ReLU. La razón, es simple. Activaciones ReLU, generan valores cero con valores negativos de entrada, situación no ideal con RCB. Siguiendo, las recomendaciones de [16], utilizamos PReLU, ya que aprende una ganancia positiva que es aplicable a valores negativos de entrada.

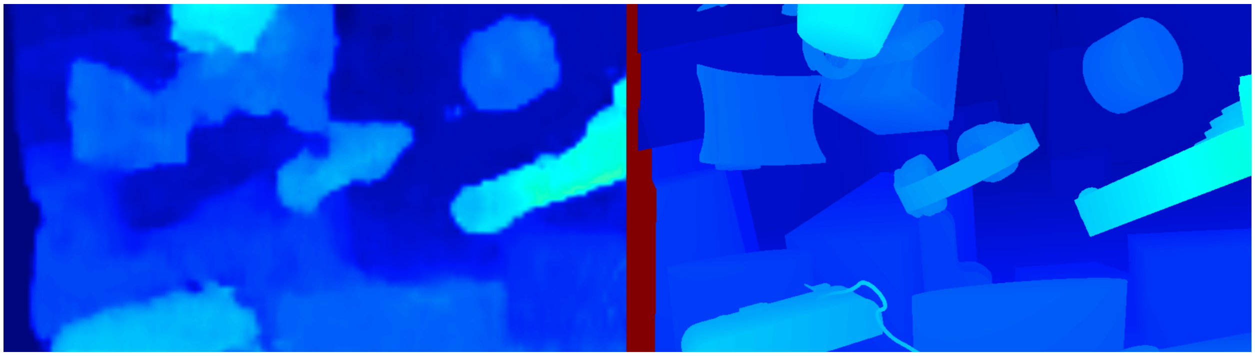
D3. Características vs Post-Procesamiento: Una prueba adicional que se realizó en este trabajo, fue ver la influencia de la extracción de características y el post-procesamiento, en la binarización de la red. Se realizaron dos experimentos. El primero, consistió en solo proceder con la binarización de la extracción de características, dejando el post-procesamiento con precisión flotante. El segundo, similar al primero, pero procediendo con la binarización del post-procesamiento, pero dejando la extracción de características con precisión flotante. El Cuadro III muestra el resultado de la prueba.

TABLA III
PRUEBA DE CARACTERÍSTICAS VS POST-PROCESAMIENTO, EN KITTI2012. MENOR ES MEJOR.

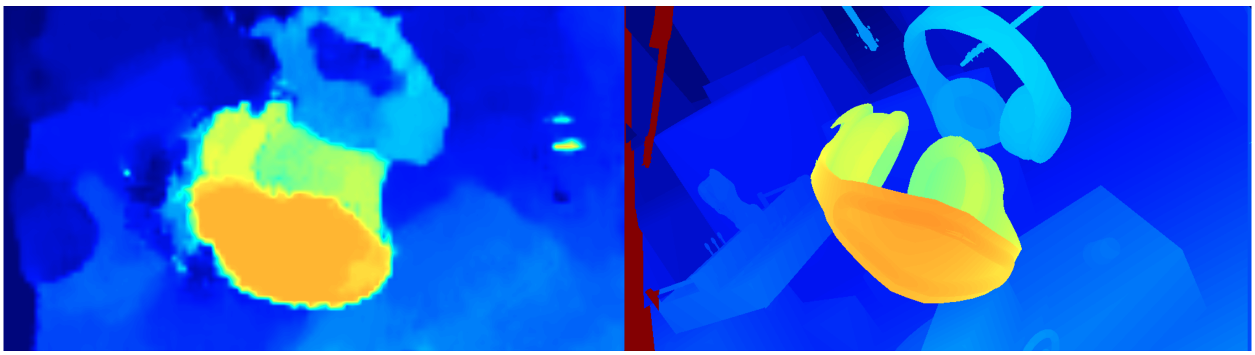
Modelo	Extracción de características	Post procesamiento	EPE	ERR_3
Propuesta	Binaria	Flotante	2.0131	0.1247
Propuesta	Flotante	Binaria	2.4901	0.1877

Los resultados en el Cuadro III, muestran que la binarización del post-procesamiento es crítica para un buen desempeño. La binarización de las redes U-Net de extracción

[2]Todas las soluciones de estimación de disparidad evaluadas en este trabajo buscan la velocidad por sobre la precisión, buscando un justo balance que permita su uso en sistemas embebidos.

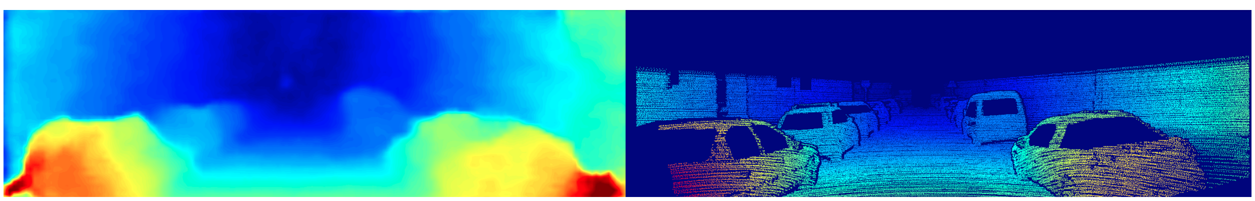


(a)

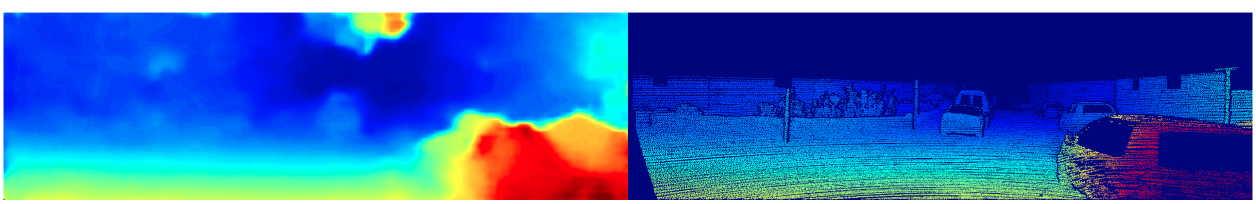


(b)

Fig. 3. Ejemplos de estimación de disparidad, en el conjunto de validación de sceneflow. A la izquierda la estimación y a la derecha el *groundtruth*.



(a) Caso bueno



(b) Caso malo

Fig. 4. Ejemplos de estimación de disparidad, en el conjunto de validación de kitti2012. A la izquierda la estimación y a la derecha el *groundtruth*; solo los pixeles validos. En 4a un caso de estimación bueno, en 4b un caso de estimación malo.

de características, no disminuye la capacidad del modelo. En cambio, la binarización de la red U-Net de post-procesamiento, es la que genera mayor perdida de desempeño.

Los resultados son también consistentes con el trabajo desarrollado por Chen et al. [7]. Cuando se considera solo la binarización de la extracción de características, se genera una perdida de alrededor de un 20 % en precisión.

D4. Volumen de Costo: La salida de una convolución binaria, es entera, dentro del rango de un número entero de 8-bit. Una mejora significativa al tiempo de procesamiento de esta red de disparidad y otras, sería poder calcular el volumen de costo con esa precisión. El volumen de costo es uno de los elementos que consume más tiempo de ejecución, que se incrementa a medida que el tamaño de la imagen de entrada aumente. Si, a diferencia de ahora, se pudiera calcular como precisión int-8 en vez de flotante, significaría una mejora substancial en la velocidad de cálculo, posibilitando el uso de imagen de entradas de mayor tamaño.

En este artículo se probo inocentemente utilizar el volumen de costo en precisión int-8, sin obtener resultados positivos.

VI. Conclusiones

La binarización de redes convolucionales, por un lado reduce el consumo de memoria requerida para almacenar los pesos, y por otro es potencialmente más rápida que una red del tipo flotante. Las características descritas, las hacen interesantes, para el uso en dispositivos de bajo consumo y limitado poder de cómputo; como sistemas embebidos.

En este trabajo, se demuestra experimentalmente que es posible la binarización de una red de estimación de disparidad del tipo *end-to-end*. En los experimentos desarrollados, la diferencia del error de punto final, va desde 0.44 a los 1.24.

Si bien en este trabajo se analizó la binarización de la red, se omitió la implementación eficiente de esta. Actualmente, las implementaciones binarias, no son genéricas, y la optimización de las implementaciones, son un trabajo de investigación en si mismo. Por lo anterior, se considera el paso lógico a seguir.

Finalmente, es interesante explorar a futuro el uso de entrenamientos más complejos, por ejemplo, utilizando destilación del conocimiento, para reducir aun más la brecha con las redes flotantes.

Agradecimentos

Este proyecto fue financiado por la Comisión Nacional de Investigación Científica y Tecnológica, a través del proyecto ANID FONDECYT 11180856.

Referencias

[1] S. Hong, M. Li, M. Liao, and P. van Beek, "Real-time mobile robot navigation based on stereo vision and low-cost gps," *Electronic Imaging*, vol. 2017, no. 9, pp. 10–15, 2017.

[2] H. Wu, H. Su, Y. Liu, and H. Gao, "Object detection and localization using stereo cameras," in *2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM)*, pp. 628–633, 2020.

[3] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, pp. 7–42, 2001.

[4] Y. Zhang, Y. Chen, X. Bai, J. Zhou, K. Yu, Z. Li, and K. Yang, "Adaptive unimodal cost volume filtering for deep stereo matching," *arXiv preprint arXiv:1909.03751*, 2019.

[5] M. Pietron and M. Wielgosz, "Retrain or not retrain? - efficient pruning methods of deep cnn networks," in *Computational Science – ICCS 2020* (V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, and J. Teixeira, eds.), (Cham), pp. 452–463, Springer International Publishing, 2020.

[6] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," *CoRR*, vol. abs/1603.05279, 2016.

[7] G. Chen, H. Meng, Y. Liang, and K. Huang, "Gpu-accelerated real-time stereo estimation with binary neural network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 12, pp. 2896–2907, 2020.

[8] G. Chen, Y. Ling, T. He, H. Meng, S. He, Y. Zhang, and K. Huang, "Stereoengine: An fpga-based accelerator for real-time high-quality stereo estimation with binary neural network," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 4179–4190, 2020.

[9] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5695–5703, June 2016.

[10] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," *CoRR*, vol. abs/1703.04309, 2017.

[11] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *CoRR*, vol. abs/1510.05970, 2015.

[12] Y. Wang, Z. Lai, G. Huang, B. H. Wang, L. Van Der Maaten, M. Campbell, and K. Q. Weinberger, "Anytime stereo image depth estimation on mobile devices," *arXiv preprint arXiv:1810.11408*, 2018.

[13] C. A. Aguilera, C. Aguilera, C. A. Navarro, and A. D. Sappa, "Fast CNN stereo depth estimation through embedded GPU devices," *Sensors*, vol. 20, p. 3249, jun 2020.

[14] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016.

[15] M. Courbariaux, Y. Bengio, and J. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *CoRR*, vol. abs/1511.00363, 2015.

[16] W. Tang, G. Hua, and L. Wang, "How to train a compact binary neural network with high accuracy?," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[17] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K. Cheng, "Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm," *CoRR*, vol. abs/1808.00278, 2018.

[18] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 345–353, Curran Associates, Inc., 2017.

[19] J. Bethge, M. Bornstein, A. Loy, H. Yang, and C. Meinel, "Training competitive binary neural networks from scratch," *CoRR*, vol. abs/1812.01965, 2018.

[20] H. Wang, Y. Xu, B. Ni, L. Zhuang, and H. Xu, "Flexible network binarization with layer-wise priority," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 2346–2350, Oct 2018.

[21] J. Xu, P. Wang, H. Yang, and A. M. López, "Training a binary weight object detector by knowledge transfer for autonomous driving," *CoRR*, vol. abs/1804.06332, 2018.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), (Cham), pp. 234–241, Springer International Publishing, 2015.

[23] T. Simons and D.-J. Lee, "A review of binarized neural networks," *Electronics*, vol. 8, p. 661, Jun 2019.

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.

[26] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[27] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[29] W. e. a. Falcon, "Pytorch lightning," 2019.

Cristhian Aguilera received the B.S. degree in automation engineer from the Universidad del Bío-Bío, Concepción, Chile, in 2008, the MSc degree in computer vision from the Universitat Autónoma de Barcelona (UAB), Barcelona, Spain, 2014, and the PhD degree in Computer Science in 2017 from the same university. His current research focuses on the industrial application of machine learning, using images from one or multiples spectra.