FPGA Implementation of 3-Bit Quantized Multi-Task CNN for Contour Detection and Disparity Estimation

Masayuki MIYAMA (a), Member

SUMMARY
Object contour detection is a task of extracting the shape created by the boundaries between objects in an image. Conventional methods limit the detection targets to specific categories, or miss-detect edges of patterns inside an object. We propose a new method to represent a contour image where the pixel value is the distance to the boundary. Contour detection becomes a regression problem that estimates this contour image. A deep convolutional network for contour estimation is combined with stereo vision to detect unspecified object contours. Furthermore, thanks to similar inference targets and common network structure, we propose a network that simultaneously estimates both contour and disparity with fully shared weights. As a result of experiments, the multi-tasking network drew a good precision-recall curve, and F-measure was about 0.833 for FlyingThings3D dataset. L1 loss of disparity estimation for the dataset was 2.571. This network reduces the amount of calculation and memory capacity by half, and accuracy drop compared to the dedicated networks is slight. Then we quantize both weights and activations of the network to 3-bit. We devise a dedicated hardware architecture for the quantized CNN and implement it on an FPGA. This circuit uses only internal memory to perform forward propagation calculations, that eliminates high-power external memory accesses. This circuit is a stall-free pixel-by-pixel pipeline, and performs 8 rows, 16 input channels, 16 output channels, 3 by 3 pixels convolution calculations in parallel. The convolution calculation performance at the operating frequency of 250 MHz is 9 TOPs/s.

key words: contour detection, disparity estimation, Multi-Task, convolutional neural network, stereo vision, FPGA

1. Introduction

Image recognition is a process of inferring the characteristics of an entity projected onto an image by analysis of the image, and is one of the pattern recognition techniques. Object detection is a task to detect objects in an image, and is a main task of the image recognition. Object contour detection is a task that finds the boundaries between objects in an image. It is not only useful by itself, but also used as a pre-process for the object detection.

Disparity estimation by stereo vision finds the disparity from the correspondence of pixels shown in two left right images simultaneously captured from cameras installed parallel. Then, the distance from the camera to the object corresponding to the pixel is determined by the principle of trigonometry. This method has been used to detect objects, and is also used to detect object boundaries.

DNN (Deep Neural Network) is a neural network with more than three layers, and in recent years its application to image recognition has been advanced and rapidly developed. Category classification is a task to find the category of the object shown in the image, and the application of the DNN dramatically improved the recognition accuracy [1]. Its applications are evolving to (category identified) object detection [2], [3], semantic segmentation that performs category classification pixel by pixel [4], [5], motion / disparity estimation to find corresponding pixels between two images [6]–[9], and object contour detection [10]–[13].

The amount of computation of DNN is enormous, and a large amount of power is required for training and inference. DNN calculations are usually performed with floating-point numbers, but low bit quantization and dedicated hardware are being developed to reduce power consumption [14], [15]. If we quantize the number to 1 bit, the product of convolution can be done with XNOR (eXclusive NOR) and the sum can be done with popcount (counting 1), so we can realize hardware with high power efficiency [16], [17]. Most of the low bit quantization of DNN and its hardware implementation have been aimed at category classification. There are few regression applications.

This paper proposes a convolution network for generic object contour detection using stereo vision. In the proposed method, the contour image is represented by the distance to the object boundary as a pixel value. Estimating this contour image during training increases the penalty for the inner false positive edge according to the distance from the boundary and reduces false positives. Furthermore, adoption of stereo vision makes it easy to detect not only specific category objects but also general object contours. It can further reduce false positives in non-boundaries within objects.

This paper also proposes a multi-task CNN (Convolutional Neural Network) for contour detection and disparity estimation with fully shared weights. The inference target of disparity estimation, distance between two pixels, is the same goal as contour estimation by the proposed method. It is reasonable to share all weights of the multi-tasking network and run both estimations simultaneously. The network reduces the calculation amount of training and inference by half, and achieves good accuracy.

Then we quantize both weights and activations of the proposed network to 3 bits with high precision. In addition, we devise hardware dedicated to the 3-bit quantization network, which features forward propagation calculations using only internal memory. As a result, data transfer with the outside of the chip, which requires a large amount of power, is eliminated. This circuit computes convolutions of 8 rows,
16 input channels, 16 output channels and 3 by 3 pixels in parallel. The convolution calculation performance at the operating frequency of 250 MHz is 9 TOPs/s. To our knowledge, there is no report on the hardware implementation of the multi-task CNN for contour detection and disparity estimation that has been quantized to 3-bit.

This study is based on our previous works presented at international conferences [34], [38]. This work changes the structure of the multi-tasking network to improve accuracy. Then the network is quantized to 3-bit and implemented on an FPGA (Field Programmable Gate Array).

The structure of this article is as follows. Section 2 describes related works. Section 3 describes the proposed contour representation method and multi-tasking CNN structure. Section 4 describes the experimental setup and shows the results. Section 5 describes the hardware architecture based on this network and the results of implementing the hardware on the FPGA. Section 6 concludes this paper.

2. Related Work

2.1 Conventional Object Contour Detection

As conventional object contour detection methods, Canny edge detection, active contour (Snakes) and graph cut are representative [18]. Canny edge detection does not distinguish between inside and outside of the object. The Snakes needs setting of the initial contour. The accuracy of the graph cut depends on the energy function used.

Authors in [19] detects lanes and general objects from stereo images of on-vehicle cameras. This study creates an image of the vehicle front seen from directly above (IPM: Inverse Perspective Mapping image) using the positional relationship between the camera and the road. Using the left IPM image, lanes are detected from the precondition that the lanes are three parallel solid-dotted-solid lines. Then, obstacles are detected from the differences between the left and right IPM images. In Ref. [20], the bounding box of the vehicle in front is detected using gray level symmetry and edge symmetry. The distance to the vehicle ahead is estimated from the position of the bottom of the bounding box. Vehicle matching is performed using left and right images to correct errors in previous distance estimation. In Ref. [21], the foreground is extracted from the disparity image to generate a bounding box of object candidates. The pedestrian is determined by a neural network using a gradient image in the bounding box. These techniques are specific to each application.

In Ref. [22], first the object contour is extracted from one of the stereo images. One image is divided by designating the inside and the outside of the interest object. Then the energy minimization framework, which integrates both stereo correspondence and object boundary constraints, extracts the corresponding contours of the other image. In Ref. [23], premising that the object and the background are clearly separated in the disparity image, object edges are extracted from a disparity image. Using the object edges as an initial contour, the object in the image is divided with high accuracy by the active contour. These methods aim at accurate contour extraction of the interest foreground object. They do not divide all objects in the image automatically.

2.2 Object Contour Detection by DNN

Authors in [10] classifies an image patch into one of the shape classes when the center of the image patch is an object boundary. This method performs multi-class classification with multiple shape classes or non-boundary class instead of two-class classification with boundary class or non-boundary class. In learning, even if the shape class is wrong, in the case the pixel is an object boundary, the loss is intentionally decreased. Although this method does not need to specify object categories that can be detected, it also detects boundaries within the object.

Authors in [11] uses the encoder-decoder architecture for object contour detection, and uses pre-trained VGG16 [1] for the encoder. Object detection is performed in combination with multiscale combinatorial grouping algorithm. The annotation targets of the dataset such as PASCAL VOC used for training are objects of several specific categories, and it is considered that the contour of the trained category object is more easily extracted.

Authors in [12] detects object boundaries for each object category. The network structure is fully convolutional, the lower layer performs category-independent edge detection, and the upper layer performs category-aware semantic edge detection. Class-labeled object boundary data (in a large amount) is required, and detected boundaries are only trained classes.

In Ref. [13], a disparity image is created using RGB stereo images, and the RGB-D images is used to detect objects by Faster R-CNN [3]. This method aims to improve the detection accuracy by adding a D image to correctly recognize the object boundary. The categories of objects are restricted, and the method is not intended for general object detection.

2.3 Disparity Estimation by DNN

FlowNet [6] is a fully convolutional network with a UNET [5]-like encoder-decoder architecture for optical flow estimation. It can also be applied to disparity estimation using stereo vision. The encoder extracts the feature map from a pair of two images while gradually lowering the resolution, and the decoder gradually increases the resolution to generate the motion flow for each pixel from the encoded feature map. The encoder has 6 layers, each with a different resolution. One layer has one or two sub-layers of convolution with LeakyReLU activation, and no batch normalization. The layers of level 1 and 2 have one sub-layer and other deeper layers have two sub-layers. The decoder performs transpose convolution with LeakyReLU on four layers. The transpose convolution performs convolution and upsampling at the same time. The final prediction is a simple
Many works have already been done on low bit-width quantized CNNs and the dedicated hardware for the quantization network, and the results were summarized in survey papers [14], [15]. The inference accuracy of 8-bit quantization is almost the same as that of single-precision floating-point numbers, and it is known that the accuracy does not drop so much even if the bits are lower than 8-bit. Research on 1-bit quantization has been also in progress [16], [17]. Many of these studies target the category classification.

Although it is smaller in number than category classification, low-bit quantization of semantic segmentation networks has also been studied [29]. There are some FPGAs for semantic segmentation [30], [31]. Xilinx provides xdnns, an 8-bit inference CNN circuit [32]. It has also announced a 4-bit quantized circuit for inference, which is used to realize semantic segmentation [33]. We have already implemented a 3-bit quantized CNN for semantic segmentation on an FPGA [34].

YOLO, a CNN for object detection, outputs a bounding box along with the class of the object and infers the size and aspect ratio of the bounding box as a regression problem [2]. TinyYOLO, a small version of YOLO, has been quantized with 1-bit weights and 4-bit activations for all layers except the first and last layers, and implemented on an FPGA using FINN [35]. Here, FINN is a framework for implementing quantized DNN on FPGAs [36]. No report was found that quantized the CNNs for motion or disparity estimation, which was the same regression problem as bounding box estimation by YOLO.

3. Proposed Method

3.1 Object Contour Representation

In general, the number of pixels at boundaries is much less than the number of pixels at non-boundaries. In the conventional method, in order to eliminate this imbalance, if it is miss-determined that the boundary is non-boundary during learning, the penalty increases compared to the opposite case. And the misjudgment penalty is independent of the distance from the boundary. As a result, the edges of the patterns inside the object tend to be considered boundaries, which increases false positives and degrades precision.

We propose a new representation of object contours to solve this problem. This representation is shown in Fig. 1 (a). Figure 1 (a) is an example of three regions. The pixel value at the region boundary is 0. The non-boundary value is a distance to the boundary, in contrast to the conventional method by which it is always 1. For example, when the boundary is erroneously estimated inside the object as shown by the dotted line in the upper right of the Fig. 1 (a), the loss of each pixel is the distance from the boundary with our method. But with the conventional method, it is always 1 as shown in Fig. 1 (b). By penalizing learning according to the distance to the boundary, false decisions within the object are reduced. Figure 1 (c) and (d) are examples of the contour image using two methods, respectively.

This is the same idea as the “distance field” used in convolution and the output is scaled to the input resolution using linear interpolation.

FlowNet2 is a high-precision motion estimation CNN configured by combining several FlowNets [7]. Recently, researches have been conducted on unsupervised learning of motion estimation CNNs using the loss based on the classical optical flow constraint [8], and CNNs for monocular motion estimation [9].

2.4 Multi-Task DNN

Multi-task learning is learning multiple different tasks simultaneously. In machine learning, it has been studied before the development of deep learning [24], [25]. Multi-task learning implicitly augments the data, suppresses overfitting by regularization, and potentially speeds up learning. There are two methods for multi-tasking DNN, one is to share the parameters hard and the other is to share them softly (regularize the parameters of each task as a whole).

In [26], the authors propose a multitasking network that performs semantic segmentation, instance segmentation, and disparity estimation at the same time. In instance segmentation, a vector to the center coordinate of the instance to which each pixel belongs is estimated for each pixel, clustering using the Hough transform is performed, and the pixels are assigned to the instances. Then, they propose a method to dynamically assign weights to the loss of each task during learning.

In [27], the authors propose a network that performs both semantic segmentation and disparity estimation. Disparity estimation is learned without a teacher. At the time of learning, a right image is created using the disparity estimated from the left image, a difference from the true right image is created, and this is used for loss. In addition, consistency with the semantic segmentation image is added to the loss. Semantic segmentation is supervised learning. The left semantic segmentation image is warped with the estimated disparity to create the false right semantic segmentation image. The consistency between this and the right semantic segmentation image is added to the loss.

In [28], the authors have incorporated semantic and instance segmentations into the network for accurate monocular disparity estimation. The network estimates the pixel disparities while considering their category instance by instance, and collects them to generate the entire disparity image.

The studies of [26] and [28] detect instances and estimate disparities at the same time, but the types of instances are limited. Also, the decoders for each task are independent and nothing is shared. In the study of [27], there was no segmentation for each object. We could not find any multitasking reports of object contour detection and disparity estimation as proposed here.

2.5 DNN Quantization and Hardware Implementation

Many works have already been done on low bit-width flow constraint [8], and CNNs for monocular motion estimation [9].
computer graphics collision detection [37]. We could not find any literature that uses the field for object contour detection. In this research, a contour image whose pixel value is an eight nearby distance to the object boundary is estimated from the input image. After thresholding the estimated image, the contour of the object is detected. This representation makes contour detection a regression problem called contour estimation, which facilitates multi-tasking with disparity estimation.

3.2 Network Structure

Like FlowNet [6], we also adopted a fully convolutional network similar to UNET [5] for our multitasking network. The network configuration is shown in Fig. 2. The left side of the figure is the encoder, and the right side is the decoder. The solid orange arrow represents the process flow and is U-shaped. The encoder inputs a pair of left and right images captured by a stereo camera and generates a feature map with a gradual reduction in resolution. The decoder inputs the feature map and generates a contour image and a disparity image while gradually increasing the resolution. Notice that the decoder of the multitasking network is independent for each task usually, but this network is only one.

Our previous work [38] used FlowNet as the network structure. Considering FPGA implementation, we change the original FlowNet as follows. Using the post-pool (orange dotted line) feature map instead of the original pre-pool (blue dotted line), as shown in Fig. 2 reduces the number of channels for each skip connection. This results in saving memory capacity. To make the activation quantization easy, not only the encoder but also the decoder performs the batch normalization after the convolution. FlowNet adopts the transposed convolution in decoding. The transposed convolution requires mask processing because the position of the input pixels differ depending on the position of the target pixel. This study replaces transposed convolution to the combination of bilinear interpolation and convolution. The combination of bilinear interpolation and convolution is simple, and it is unnecessary to change the convolution circuit.

3.3 Quantization Method

The method of weight quantization will be described. The first training was done without clipping the upper and lower weight limits. The second training was performed with the first training result as the initial value with clipping the upper and lower limits to twice the standard deviation of the layer and the encoder output of the same layer. ‘conv’ represents a convolution of 3 by 3 pixels. The number added to the layer name indicates the hierarchy (resolution) level of the layer. Since the network inputs two color images on the left and right, the number of input channels is six. The ‘Predict’ layer outputs two channels, the object contour image described in the previous subsection and the disparity image. However, if either contour estimation or disparity estimation is performed alone, as in the experiment in the next section, the number of output channels is one.
weight distribution. The accuracy of the second result was no less than that of the first result. Therefore, we set twice the standard deviation as the range of weight quantization. The bias for all convolutions is set to zero.

The activation quantization will be described. The output of the activation function, that is, the input of the next convolution is called activation. This network always performs batch normalization after convolution, and then activates with ReLU. The distribution of activations after batch normalization has a mean of 0 and a variance of 1. After that, since ReLU is performed, the activation value becomes 0 or more. Therefore, the quantization range of activations for all layers are uniformly set to 0 or more and X or less. X is a constant here.

The traditional quantization procedure in Ref. [29] first trains a network in which both weights and activations are floating point numbers. Then use this training result (the weight for the highest validation accuracy) as the initial value to train the 8-bit network. It then gradually trains the low-bit quantization network, reducing both weights and activations by 1 bit. We compared the traditional procedure with the procedure setting training results of the floating-point network as initial weights for all quantized networks. The results of preliminary experiments have shown that the latter procedure works better.

A training method for quantization networks proposed in [16] is adopted. This method uses quantized weights and activations in the forward propagation calculation, but the weights are stored as floating point numbers and the gradients are calculated by the floating point numbers in the backpropagation calculation and the stored weights are updated by the gradients.

4. Experiment

In our previous work [38], we have already reported the experimental results explaining the advantages of the proposed contour representation over the conventional method. This time, we will show the experimental results on the performance of the multi-tasking network compared to the dedicated network. The experimental results of the quantized network will be also described.

4.1 Setup

The comparison targets are the following four networks:

- DN: proposed network only for disparity estimation,
- CN: proposed network only for contour estimation,
- DC: proposed multi-task network for both estimation,
- FN: multi-task FlowNet for both estimation.

DN and CN have the same structure with one output channel. DC has the same structure as the DN and CN, except that the output has two channels. This difference is negligible when considering the computational complexity and memory capacity of the entire network. Therefore, when comparing the case where disparity estimation and contour estimation are performed in parallel with DN and CN (DN+CN) and the case where both estimation is performed with DC, the amount of calculation of DC is half that of DN+CN. And the memory capacity for the convolution weight parameter and feature map is halved. FN is a baseline approach in this experiment. The structure of FN is described in Sect. 2.3. However, unlike the original, the final layer has two output channels and predicts not only disparity but also the object contour.

We used datasets of FlyingThings3D and Driving [39], [40]. These are datasets developed for motion estimation and disparity estimation studies. The ground truth of a contour image was created from an object index image (an image having a number uniquely assigned to each object as a pixel value) included in the data set.

FlyingThings3D (FT) is a dataset of unrealistic stereo images drawn by 3D computer graphics. In the image, objects such as desks and chairs fly in a three-dimensional space. The resolution is 960 × 540 pixels. We independently removed data that made motion estimation training difficult. The number of scenes we use is 1,090 and the number of images is 10,340. Of these, 90 \% were used for training and 10 \% for inference.

Driving (DR) is a dataset of realistic stereo images. The image, shooting in the forward direction from the car driver’s point of view, is drawn in 3D computer graphics. The objects that appear are cars, streetlights, trees, buildings, roads. We used “15mm_focallength / scene_forward / slow” for training and “15mm_focallength / scene_backward / slow” for inference. Each is 800 continuous stereo images, and the resolution is 960 × 540 pixels. We cropped all images so that they have a width of 720 and a height of 360, with the origin on the upper left as (120, 0). The first 700 images of the inference sequence were used.

We used L1 loss as the loss function during learning. Over the entire image, L1 loss averages the absolute difference values between the same position pixels of the inferred image and the correct image. The optimization method was Adam, the learning rate was 1e-4, and the mini-batch size was 8. For the size of random cropping, FT was 384 × 384 pixels, and DR was 512 × 256. In training of FT, random initialization was performed and 50 epochs were executed. In DR training, we started with the best model of FT, and executed training of 25 epochs.

Quantized networks were tested next. The quantization range of weights for all convolutions was set to −0.0625 to 0.0625. The quantization range of all activations was set to 0 to 3. These ranges were determined by preliminary experiments. Training procedure is described in Sect. 3.3. The learning rate of Adam was 1e-6 until 4-bit quantization. From the 3-bit quantization, we set the rate at 1e-5 for FT and 1e-4 for DR. Each quantized network was trained for 25 epochs.

Regarding the execution environment, the computer’s CPU was Intel Core i9-7900X 3.30GHz, the memory was 64 GB, and the GPGPU (General Purpose Graphics Processing Unit) was an NVIDIA GeForce GTX 1080 ti. Deep
Table 1  L1 loss comparison for 4+1 networks. The row (C) describes L1 Loss for contour estimation and (D) for disparity estimation. The accuracy of the multi-tasking network DC is slightly lower compared to that of each dedicated network, but higher than that of the FN.

| Data | FN | CN | DN | DC | DC3 |
|------|----|----|----|----|-----|
| FT(C) | 3.634 | 2.576 | - | 2.804 | 3.539 |
| FT(D) | 2.746 | - | 2.403 | 2.571 | 3.138 |
| DR(C) | 6.648 | 4.365 | - | 4.516 | 6.355 |
| DR(D) | 5.842 | - | 4.815 | 4.958 | 6.065 |

Fig. 3  Estimation results of FlyingThings3D. You can see that the edges inside the objects are not falsely detected as contours. In the disparity estimation results, the shape of each object is clearly shown.

Fig. 4  Estimation results of Driving. It is difficult to detect contours of the leaves because they are too fine. The upper part of the vehicle is difficult to estimate the disparity because of the window transparency.

learning platform used PyTorch 0.4.1 [41]. We used the FlowNet2 source code implemented by PyTorch [42].

4.2 Results

Table 1 shows the L1 loss when the four networks infer FT and DR. The row (C) describes L1 Loss for contour estimation and (D) for disparity estimation. FT (C) is contour estimation and there is no DN (disparity estimation) result. FT (D) is disparity estimation and there is no CN (contour estimation) result. The same applies to DT (C) and DT (D). The accuracy of the multi-tasking network DC is slightly lower compared to that of each dedicated network, but higher than that of the FN. Figures 3 and 4 show DC results of FT and DR respectively. The contour detection results are created by overwriting the binary contour to the left image. The threshold of the contour detection is 3. You can see that the edges inside the objects are not falsely detected as contours in FT. In DR, it is difficult to detect contours of the leaves because they are too fine. In the disparity estimation results of FT, the shape of each object is clearly shown. In DR, the upper part of the vehicle is difficult to estimate because of the window transparency.

The experimental results of quantization of the multi-tasking network DC are shown in Fig. 5. In both inferences, as the number of bits decreased, the accuracy decreased. By increasing the learning rate, the 3-bit accuracy was kept good. However, even with high learning rates, 2-bit accuracy remained poor.

Fig. 5  L1 loss of quantized network. In both inferences, as the number of bits decreased, the accuracy decreased. By increasing the learning rate, the 3-bit accuracy was kept good. However, even with high learning rates, 2-bit accuracy remained poor.

Fig. 6  Precision-recall curves. The closer the P-R curve is to the upper right, the higher the accuracy. The accuracy of DC was much better than that of FN, although lower than that of CN. The accuracy of 3-bit DC was even better than that of FN in the case of FT. It was equal to or better than that of FN in the case of DR.

The results of contour detection will be described in detail. The accuracy metrics precision, recall, and F-measure...
Table 2  F-measure(precision/recall)s for five networks. The F-measure of DC was 0.833 for FT and 0.738 for DR. The F-measure of 3-bit DC was 0.813 for FT and 0.652 for DR, and those were still high when comparing to FN with floating number.

| Data | FN       | CN       | CN3      | DC       | DC3      |
|------|----------|----------|----------|----------|----------|
| FT   | 0.764 (0.745/0.785) | **0.849 (0.802/0.901)** | 0.827 (0.788/0.870) | 0.833 (0.797/0.875) | 0.813 (0.788/0.840) |
| DR   | 0.641 (0.588/0.704) | **0.752 (0.741/0.763)** | 0.654 (0.620/0.695) | 0.738 (0.725/0.752) | 0.652 (0.618/0.690) |

are defined as follows.

\[
\text{precision} = \frac{tp}{tp + fp}
\]

\[
\text{recall} = \frac{tp}{tp + fn}
\]

\[
F - \text{measure} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}
\]

Here, \( tp \) is the number of true positives, \( fp \) is the number of false positives, and \( fn \) is the number of false negatives. A true positive is to correctly judge a boundary as a boundary. A false positive is to erroneously judge a non-boundary as a boundary. A false negative is to erroneously judge a boundary as a non-boundary.

The method of creating the precision-recall (P-R) curve is described. For the estimated image from the network, a pixel having a value less than or equal to the threshold was determined as a boundary, and the final binary image is created. The thresholds were 1, 2, 3, 4, 5, 6, 7, 8. By changing the threshold for each estimated image, eight binary images were created, and precisions and recalls were calculated. For each threshold, precisions and recalls for all binary images were averaged, and a P-R curve was drawn with eight average values.

Figure 6 shows P-R curves of five networks: FN, CN, 3-bit quantized CN, DC, and 3-bit quantized DC, for both datasets. The closer the P-R curve is to the upper right, the higher the accuracy. The accuracy of DC was much better than that of FN, although lower than that of CN. The accuracy of 3-bit DC was even better than that of FN in the case of FT. It was equal to or better than that of FN in the case of DR. The best F-measures of the five networks for both datasets are shown in Table 2. The F-measure of DC was 0.833 for FT and 0.738 for DR. The F-measure of 3-bit DC was 0.813 for FT and 0.652 for DR, and those were still high when comparing to FN with floating number.

5. Hardware Architecture

Figure 7 shows a block diagram of the multi-task processor for contour and disparity estimation. FM, feature memory, is composed of 10 banks, and 10 consecutive rows adding two rows above and below to the middle eight rows are read simultaneously. Each FM bank has a 48-bit word, storing 3-bit activations of 16 channels. One column of 10 consecutive rows is read from FM every cycle. RF reassembles this into eight data of three consecutive rows, and sends them to IP. IP and subsequent blocks are divided into eight units, and execute eight rows in parallel. Each IP unit performs bi-linear interpolation of twice the height and width. Each MW unit combines three consecutive columns to make a window of 3 by 3 16-channel activations. 16 copies of this are sent to the corresponding CV unit. CV performs convolution calculations. Each CV unit is composed of 16 CVEs, element circuits of CV, and processes 16 output channels in parallel. The 16 output channels are processed by a BR unit, performing batch normalization and ReLU. It is followed by max pooling with an MP unit. 16 channels generated by the MP unit are written to an FM bank as one word via WF. Weight memory WM connecting to CV and parameter memory PM connecting to BR are omitted in this diagram.

Figure 8 shows a block diagram of the CVE. K00 to K15 perform 3 by 3 kernel calculations for 16 input channels. The result of the kernel calculation is given to the adder tree. The accumulator accumulates the product-sum operation results of all input channels for one output channel. Since both activation and weight are 3-bit in this network, one digit of multiplication output can be realized by one 6-input LUT (Look Up Table). In other words, multiplication of 3-bits can be realized with only 6 LUTs.

This circuit calculates convolution of 3 by 3 pixels in 8 rows, 16 input channels and 16 output channels in parallel. When the operating frequency is \( F = 250 \text{ MHz} \), the
number of operations (OPs) per second in the convolution calculation is $N = 9$ (TOPs/s). Here, one product–sum operation is divided into product and sum, and is counted as two operations. Batch normalization, ReLU, and max pooling operations are not included here.

6. FPGA Implementation

We implemented the proposed multi-task processor on Alveo U200 from Xilinx. The Alveo U200 is an accelerator card involving an FPGA for image processing and AI inference. The development environment was Vitis 2020.1 from Xilinx. The circuit scale of the processor are shown in Table 3.

Next, we compared with Vitis AI. Vitis AI consists of FPGA-based DNN processors developed by Xilinx, and development environment including quantization and compiler tools [43]. The neural network developed with machine learning frameworks such as Caffe and Tensorflow is quantized by 8-bit, compiled, and executed on the DNN processor. The throughput of UNET by the DNN processor with 700 MHz frequency is 44 fps (frame per second). On the other hand, the throughput of the proposed processor is 134 fps.

7. Conclusion

In order to detect only the outline of an object, we proposed a new representation method of the object contour. The network we proposed for general object contour detection combined with stereo vision produces an image whose pixel value is a distance to the nearest boundary. The experimental results showed that the proposed method drew a good precision-recall curve, and the F-measure was about 0.83 for FlyingThings3D dataset. Furthermore, we proposed a network that completely shares weights between contour estimation and disparity estimation. In this multi-tasking network, the computational complexity and memory capacity were halved, and the drop in accuracy was slight. We improved the multi-task CNN for FPGA implementation, and quantized both weights and activations to 3-bit. With 3-bit quantized CNN processor, all calculations are performed using only the FPGA internal memory, which eliminates power consumption due to data transfer with external memory during the forward propagation calculation. It features a pixel-by-pixel pipeline without stall, and performs 3 by 3 pixels convolution calculation of 8 rows, 16 input channels, and 16 output channels in parallel. The implemented FPGA performs convolution calculations with throughput of 9 TOPs/s at an operating frequency of 250 MHz. Future work is to develop a CNN dedicated processor that performs semantic segmentation, and disparity estimation at the same time.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP 18K11350.

References

[1] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” International Conference on Learning Representations, 2015.
[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: unified, real-time object detection,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.779–788, 2016, doi: 10.1109/CVPR.2016.91.
[3] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: towards real-time object detection with region proposal networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol.39, no.6, pp.1137–1149, June 2017.
[4] E. Shelhamer, J. Long, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol.39, no.4, pp.640–651, April 1 2017.
[5] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: convolutional networks for biomedical image segmentation,” In: N. Navab, J. Hornegger, W. Wells, and A. Frangi (eds) Medical Image Computing and Computer-Assisted Intervention MICCAI 2015, Lecture Notes in Computer Science, vol.9351, Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28.
[6] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazbars, and V. Golkov, “FlowNet: learning optical flow with convolutional networks,” 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7-13 Dec. 2015.
[7] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “FlowNet 2.0: evolution of optical flow estimation with deep networks,” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.1647–1655, 2017, doi: 10.1109/CVPR.2017.179.
[8] A. Ahmadi and I. Patras, “Unsupervised convolutional neural networks for motion estimation,” 2016 IEEE International Conference on Image Processing (ICIP), pp.1629–1633, 2016, doi: 10.1109/ICIP.2016.7532634.
[9] C. Godard, O.M. Aodha, and G.J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.6602–6611, 2017, doi: 10.1109/CVPR.2017.699.
[10] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang, “Object contour detection with a fully convolutional encoder-decoder network,” The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.193–202, 2016.
[11] Y. Liu, M.M. Cheng, D.P. Fan, L. Zhang, J.W. Bian, and D. Tao, “Semantic edge detection with diverse deep supervision,” arXiv:1804.02864v2 [cs.CV].
[12] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, “DeepContour: a deep convolutional feature learned by positive-sharing loss for contour detection,” 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 7-12 June 2015, Boston, MA, USA.
[13] C. Guindel, D. Martin, and J.M. Armingol, “Stereo vision-based convolutional networks for object detection in driving environments,” In: R. Moreno-Diaz, F. Pichler, and A. Quesada-Arencibia (eds) Computer Aided Systems Theory EUROCAST 2017, Lecture Notes in Computer Science, vol.10672, Springer, Cham.
[14] E. Wang, J.J. Davis, R. Zhao, H.-C. Ng, I. X. Niu, W. Luk, P.Y.K. Cheung, and G.A. Constantinides, “Deep neural network approximation for custom hardware: where we’ve been, where we’re going,” ACM Computing Surveys, vol.52, no.2, Article 40, Publication date: May 2019.

[15] T. Simons and D.-J. Lee, “A review of binarized neural networks,” MDPI Electronics, vol.8, 661, 2019. doi:10.3390/electronics8060661.

[16] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks: training neural networks with weights and activations constrained to +1 or -1,” arXiv:1602.02830 [cs.LG].

[17] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “XNOR-Net: ImageNet classification using binary convolutional neural networks,” In: B. Leibe, J. Matas, N. Sebe, and M. Welling (eds) Computer Vision ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol.9908, Springer, Cham. https://doi.org/10.1007/978-3-319-46493-0_32.

[18] R. Szeliski, “Computer vision algorithms and applications,” Springer.

[19] M. Bertozzi and A. Broggi, “GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection,” IEEE Trans. Image Process., vol.7, no.1, pp.62–81, Jan. 1998.

[20] M. Bertozzi, A. Broggi, A. Fascioli, and S. Nichiele, “Stereo vision-based vehicle detection,” Proc. IEEE Intelligent Vehicles Symposium 2000, Dearborn (MI), USA, Oct. 3-5, 2000.

[21] L. Zhao and C.E. Thorpe, “Stereo and neural network-based pedestrian detection,” IEEE Trans. Intell. Transp. Syst., vol.1, no.3, pp.148–154, Sept. 2000.

[22] R. Ju, T. Ren, and G. Wu, “StereoSnakes: contour based consistent object extraction for stereo images,” 2015 IEEE International Conference on Computer Vision (ICCV), 7-13 Dec. 2015, Santiago, Chile.

[23] M. Gelautz and D. Markovic, “Recognition of object contours from stereo images: an edge combination approach,” Proc. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 3DPVT 2004, 9-9 Sept. 2004, Thessaloniki, Greece, Greece.

[24] S. Ruder, “An overview of multi-task learning in deep neural networks,” arXiv:1706.05098 [cs.LG].

[25] M. Crawshaw, “Multi-task learning with deep neural networks: a survey,” arXiv:2009.09796 [cs.LG]

[26] R. Cipolla, Y. Gal, and A. Kendall, “Multi-task learning using uncertainty to weigh losses for scene geometry and semantics,” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp.7482–7491, 2018, doi: 10.1109/CVPR.2018.00781.

[27] P. Chen, A.H. Liu, Y. Liu, and Y.F. Wang, “Towards scene understanding: unsupervised monocular depth estimation with semantic-aware representation,” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp.2619–2627, 2019, doi: 10.1109/CVPR.2019.00273.

[28] L. Wang, J. Zhang, O. Wang, Z. Lin, and H. Lu, “SDC-Depth: semantic divide-and-conquer network for monocular depth estimation,” 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp.538–547, 2020, doi: 10.1109/CVPR42600.2020.00062.

[29] H. Wen, S. Zhou, Z. Liang, Y. Zhang, D. Feng, X. Zhou, and C. Yao, “Training bit fully convolutional network for fast semantic segmentation,” arXiv:1612.00212 [cs.CV].

[30] S. Vogel, J. Springer, A. Gunter, and G. Ascheid, “Efficient acceleration of CNNs for semantic segmentation on FPGAs,” Proc. 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, https://doi.org/10.1145/3289062.3294006.

[31] M. Shimoda, Y. Sada, and H. Nakahara, “Filter-wise pruning approach to FPGA implementation of fully convolutional network for semantic segmentation,” In: C. Hochberger, B. Nelson, A. Koch, R. Woods, and P. Diniz (eds) Applied Reconfigurable Computing, ARC 2019. Lecture Notes in Computer Science, vol.11444, pp.371–386, Springer, Cham. https://doi.org/10.1007/978-3-030-17227-5_26.

[32] R. Szeliski and A. Siraasoo, “Xilinx ML suite,” IEEE Silicon Valley Machine Learning Compiler workshop- Fall 2018.

[33] “Convolutional neural network with int4 optimization on Xilinx devices,” WP521 (v1.0.1) June 24, 2020.

[34] M. Miyama, “FPGA implementation of 3-bit quantized CNN for semantic segmentation,” IOP Publishing, Journal of Physics: Conference Series, vol.1729, pages 012004, Jan. 2021, https://doi.org/10.1088/1742-6596/1729/1/012004.

[35] https://github.com/mohdumar644/TinyYOLO-BNN

[36] M. Blott, T.B. Preußen, N.J. Fraser, G. Gambardella, K. O’Brien, Y. Umuroglu, M. Leeser, and K. Vissers, “FINN-R: an end-to-end deep-learning framework for fast exploration of quantized neural networks,” ACM Trans. Reconfig. Technol. Syst. 1, 1, 22 pages, Sept, 2018, https://doi.org/10.1145/3242897.

[37] A. Fuhrmann, G. Sobottka, and C. Gros, “Distance fields for rapid collision detection in physically based modeling,” International Conference on Computer Graphics and Vision (Graphicon), 2003.

[38] M. Miyama, “Convolutional network for generic object detection with stereo vision,” Proc. 2019 2nd International Conference on Information Hiding and Image Processing, pp.25–29, Sept. 2019, https://doi.org/10.1145/3383913.3383921.

[39] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, “A Large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation,” IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[40] https://lmb.informatik.uni-freiburg.de/resources/datasets/SceneFlowDatasets.en.html

[41] https://pytorch.org/

[42] https://github.com/NVIDIA/flownet2-pytorch

[43] https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html

Masayuki Miyama was born on March 26, 1966. He received a B.S. degree in Computer Science from the University of Tsukuba in 1988. He joined PFU Ltd. in 1988. He received an M.S. degree in Computer Science from the Japan Advanced Institute of Science and Technology in 1995. He joined Innotech Co. in 1996. He received a Ph.D. degree in electrical engineering and computer science from Kanazawa University in 2004. He is an assistant professor at Kanazawa University Graduate School of Science and Technology. His present research focus is VLSI algorithm and architecture for real-time image processing.