Communication



# Research on Information Fusion System Implementation Based on ARM-Based FPGA

Yu-Hsiang Tsai , Yung-Jhe Yan , Meng-Hsin Hsiao , Tzu-Yi Yu , Mang Ou-Yang *

Posted Date: 13 June 2023

doi: 10.20944/preprints202306.0863.v1

Keywords: augmented reality; information fusion; transparent display

*Communication*

# Research on Information Fusion System Implementation Based on ARM-Based FPGA

**Yu-Hsiang Tsai [1,3], Yung-Jhe Yan [2], Meng-Hsin Hsiao [2], Tzu-Yi Yu [3] and Mang Ou-Yang [1,2,*]**

[1] Institute of Electrical and Computer Engineering, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan; YuHsiang@itri.org.tw
[2] Institute of Electrical and Control Engineering, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan; jerryyan@nycu.edu.tw; a6090240807@gmail.com
[3] Industrial Technology Research Institute, Hsinchu 30010, Taiwan; ritayu@itri.org.tw
* Correspondence: oym@nycu.edu.tw

**Abstract:** This research uses ARM-based FPGA to implement the computation of a virtual-real fusion hardware system. It also evaluates the acceleration effect of virtual-real fusion execution speed after connecting related hardware acceleration and provides a reference for real-time information fusion system design. The hardware uses Xilinx Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit. It also obtains color and depth images with Intel® Realsense™ D435i depth cameras. This system receives image data through the Linux system built on the ARM and fed to the FPGA for object detection using CNN models. In addition, it develops a module for rapidly performing registration between the color images and the depth images. It can calculate the size and position of the display information on the transparent display according to the pixel coordinates and depth values of the human eye and the object. The fusion took about 47ms on a personal computer with a GPU RTX2060 and 25ms on the ZCU102. The acceleration is nearly 168%. Finally, the results were successfully transplanted into the retail display system for demonstration.

**Keywords:** augmented reality; information fusion; transparent display

## 1. Introduction

Augmented reality (AR) in the large-area direct-view transparent display, called an information fusion system, is convenient in public field applications. Such systems must tackle the challenge of merging information as the technique has to consider the relationships between the screen, background object, and the user's gaze direction to overlap the corresponding virtual information on the screen. Moreover, direct-view displays are more user-friendly and intuitive for users. As a result, the development and applications of direct-view transparent displays have gradually grown in our daily lives, regarded as an indispensable part of smart life applications [1-2]. In this system, artificial intelligence (AI) recognition technology was used to get the object and the user's gaze direction. Thus, the computing power capacity for AI and the tradeoff between computing power and power consumption is the most critical issue in this system.

In AI recognition technology, the establishment and training of deep learning models are mostly implemented in GPU-based computers. However, GPU-based computers are not the best choice if the trained model is to be applied to the field. Comprehensive consideration of speed, power consumption, size, and flexibility, FPGA-based embedded systems should be a better choice for realizing real-time deep learning identification and classification models.

One of the advantages of FPGA is including programmable logic unit, built-in memory, and DPS floating point unit. It performs high-speed data exchange with peripheral hardware so that the deep learning model can be realized and run on the FPGA entity in a chip-like manner [3]. The execution speed can be more than 3 to 5 times that of GPUs, and the average power consumption is also lower than GPUs. Therefore, FPGAs are more suitable for deep model identification or classification work in prediction or inference [4].

 doi:10.20944/preprints202306.0863.v1



The Implementation and Performance Summary of FPGA-based Accelerators, as indicated in references [5], has two discussion directions for accelerating CNN model computation through FPGA. The first approach is to compress the CNN model. Reducing memory footprint uses singular value decomposition (SVD). Filter clustering methods [6] simplify the connection numbers of convolution or fully connected layers through network pruning techniques [7-9] to reduce model complexity and prevent over-fitting. The method of enforcing weight sharing [10] is also used to reduce memory footprint. The other direction focuses on implementing CNN based on the architecture of FPGA hardware, including the design of hardware parallelization and the data types and arithmetic units.

In deep learning model computation, addition and multiplication operations are required for convolution, pooling, and fully connected layers. Typically, floating-point arithmetic is used to avoid sacrificing precision. However, the number of floating-point arithmetic units inside FPGA is limited, and the amount of data that needs to be temporarily stored is quite substantial, depending on the number of layers and parameters in the model. Common improvements include changes in data processing formats and algorithms, which also affect the design of parallel processing architecture. Many studies have explored ways to reduce the number of bits or data operations without sacrificing too much precision. Previous research [11-13] has proposed using fixed-point formats with 8 to 16 bits as the basic arithmetic format to improve data computation speed and reduce the amount of data without reducing prediction accuracy. However, this method requires additional optimization and adjustment or adding floating-point arithmetic in the data conversion step. Vitis AI is a development platform from Xilinx designed for artificial intelligence inference on Xilinx hardware platforms. It consists of optimized IP, tools, libraries, models, and example designs. It supports mainstream frameworks and the latest models and is capable of executing various deep-learning tasks. Vitis AI also provides a range of pre-trained models that can be retrained for specific applications.

In this research, the fusion information is shown on a transparent screen. Figure 1 shows the architecture of the system. Through the comparison of the deep learning model of ARM base FPGA, it took the YOLO model of a single image to recognize a Lego box as an example. It takes about 47ms to execute on the personal computer with GPU RTX2060 and 25ms when executed on the ZCU102 development kit. The acceleration is about 168%.

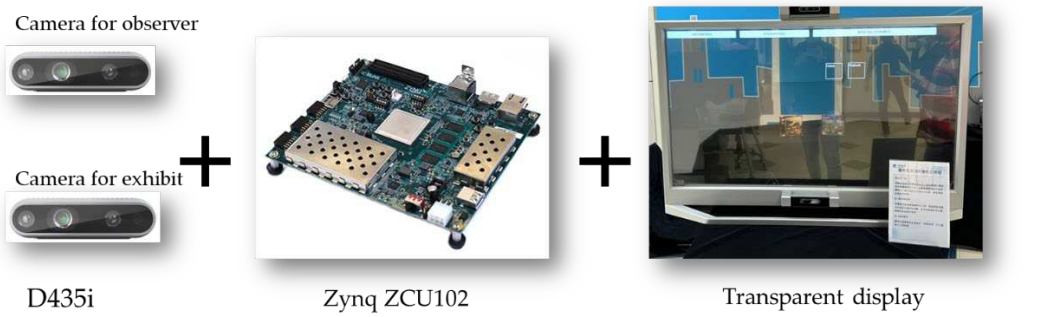


**Figure 1.** Device diagram in the information fusion system.

## 2. Information Fusion Method on Transparent Display

The information fusion system can generate an image in the line sight of an observer that sees the image overlapped on their interested exhibit. The side view of the information fusion system is shown in Fig.2. The system detects the observer's and the exhibit's positions relative to the transparent display. Thus, the system employs two depth color cameras to capture the color and depth images of the observer and the exhibit. The system receives these images and feeds the images into neural network models for locating the observer and the exhibit in the color images. After finding the observer's and the exhibit's position in the color images, their position in the depth image corresponding to the color images can be known. Thus, the position of the observer and exhibit in the real word can be derived by mapping the position in the image coordinates from the real word

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 June 2023 doi:10.20944/preprints202306.0863.v1



coordinate. Afterward, the system calculates the displayed position and the size of the fusion information according to the relative positions of the observer and the exhibit. Finally, the system displays the fusion information on the transparent display to provide additional details for the exhibit. The workflow of the information fusion system is shown in Fig. 3.

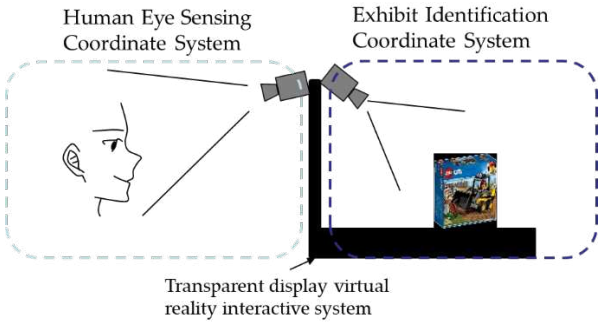


**Figure 2.** Side View of the information fusion system.

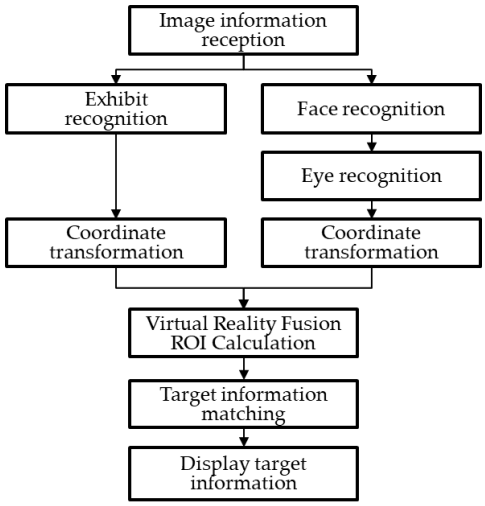


**Figure 3.** Working Flow Chart of the Information Fusion system.

The system has two essential processes: recognition and position acquisition of the fusion information. The first position acquisition process is to do the image registration between the depth and color images. The coordinates map from the depth images to the color images are according to Eq. 1, Eq. 2, and Eq. 3. Eq. 1 is the transformation from the two-dimensional depth image coordinate ($u^D$,$v^D$) to the three-dimensional depth camera coordinate ($X^D$, $Y^D$, $Z^D$). $f_x^D$ and $f_y^D$ are the focal lengths of the depth camera; $p_x^D$ and $p_y^D$ are the center position of the depth image; $d$ is the depth data of the depth image. Eq.2 is the transformation from the depth camera coordinate to the three-dimensional color camera coordinate ($X^C$, $Y^C$, $Z^C$). The $r_{11}^{DC}$ to $r_{33}^{DC}$ is the rotation parameter, and the $t_1^{DC}$ to $t_3^{DC}$ is the translation parameter. Eq. 3 is the transformation from the color camera coordinate to the two-dimensional color image coordinate ($u^C$,$v^C$).

$$\begin{bmatrix} X^D \\ Y^D \\ Z^D \end{bmatrix} = d \begin{bmatrix} f_x^D & 0 & p_x^D \\ 0 & f_y^D & p_x^D \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u^D \\ v^D \\ 1 \end{bmatrix} \tag{1}$$

 doi:10.20944/preprints202306.0863.v1



$$\begin{bmatrix} X^C \\ Y^C \\ Z^C \end{bmatrix} = \begin{bmatrix} r_{11}^{DC} & r_{12}^{DC} & r_{13}^{DC} & t_1^{DC} \\ r_{21}^{DC} & r_{22}^{DC} & r_{23}^{DC} & t_2^{DC} \\ r_{31}^{DC} & r_{32}^{DC} & r_{33}^{DC} & t_3^{DC} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X^D \\ Y^D \\ Z^D \\ 1 \end{bmatrix} \tag{2}$$

$$Z^C \begin{bmatrix} u^C \\ v^C \\ 1 \end{bmatrix} = \begin{bmatrix} f_x^C & 0 & p_x^C \\ 0 & f_y^C & p_y^C \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X^C \\ Y^C \\ Z^C \end{bmatrix} \tag{3}$$

After detecting the observer or the exhibit in the color image, the position of the observer or the exhibit in the three-dimensional color camera coordinate can be derived from the inverse transformation of Eq. 3. Afterwards; the positions are mapped to the three-dimensional transparent display coordinate ($X^M$, $Y^M$, $Z^M$) according to Eq. (4). The $r_{11}^{CM}$ to $r_{33}^{CM}$ is the rotation parameter. The $t_1^{CM}$ to $t_3^{CM}$ is the translation parameter.

$$\begin{bmatrix} X^M \\ Y^M \\ Z^M \end{bmatrix} = \begin{bmatrix} r_{11}^{CM} & r_{12}^{CM} & r_{13}^{CM} & t_1^{CM} \\ r_{21}^{CM} & r_{22}^{CM} & r_{23}^{CM} & t_2^{CM} \\ r_{31}^{CM} & r_{32}^{CM} & r_{33}^{CM} & t_3^{CM} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X^C \\ Y^C \\ Z^C \\ 1 \end{bmatrix} \tag{4}$$

The positions of the observer and the exhibit in the transparent display coordinate are ($X_o^M$, $Y_o^M$, $Z_o^M$) and ($X_e^M$, $Y_e^M$, $Z_e^M$). Eq. 5 can calculate the position of the fusion information in the transparent display coordinate system. The coordinate ($X_m^M$, $Y_m^M$, $Z_m^M$) represents the position of the fusion information, as shown in Fig. 4. Finally, the position and the size transparent display coordinate are mapped to the transparent image coordinate for display, as described by Eq. (6).

$$\begin{bmatrix} X_m^M \\ Y_m^M \end{bmatrix} = \frac{Z_e^M}{Z_e^M - Z_o^M} \begin{bmatrix} X_o^M \\ Y_o^M \end{bmatrix} + \frac{-Z_o^M}{Z_e^M - Z_o^M} \begin{bmatrix} X_e^M \\ Y_e^M \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} u^M \\ v^M \end{bmatrix} = \begin{bmatrix} s_x & 0 & p_x \\ 0 & s_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X^M \\ Y^M \\ 1 \end{bmatrix} \tag{6}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 June 2023 doi:10.20944/preprints202306.0863.v1



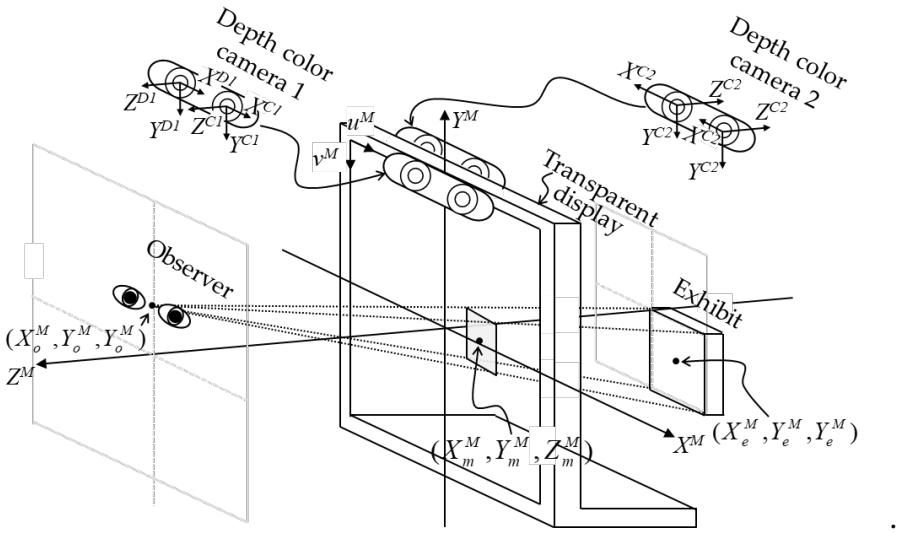


**Figure 4.** Schematic diagram and coordinate definition of the information fusion system.

## 3. Implementation

The information fusion system mainly comprised a transparent display, two depth cameras, Realsense D435i, and a Xilinx Zynq UltraScale+ MPSoC ZCU102 development board. The depth camera provides depth images and color images for object detection. The development board has an ARM-based FPGA XCZU9EG chip with a quad-core Arm® Cortex®-A53 (PS), dual-core Cortex-R5F real-time processors, Mali™-400 MP2 graphics processing unit (MGPU), and a programmable logic (PL). The PS side received image data and sent images to the PL. The image registration between the color and depth images was performed on the PL to speed up the computation. The PL has a Deep Learning Processor Unit (DPU) that executes the exhibit and the observer detection. The image registration and detection results were sent back to the PS. Afterward, the PS calculated the size and position of the displayed information on the transparent display according to the coordinates and depth values of the observer's eyes and objects. The MGPU received the display information from the PS and updated the information to the transparent display. The computation architecture of the system is shown in Figure 5.

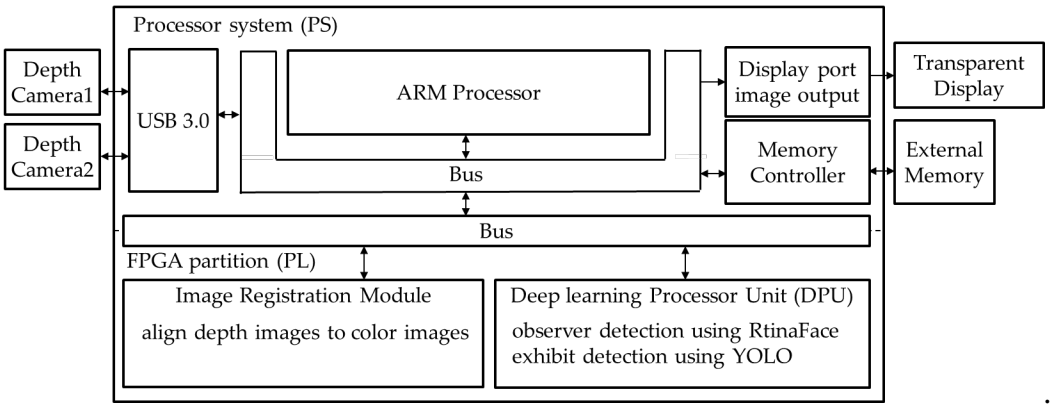


**Figure 5.** Computation system block diagram.

The tasks of the PS were implemented in a multi-thread way. A total of four threads, including DecodeThread, DpuThread, SortingThread, and GuiThread, were used. The DecodreThread received the color and depth images and sent the images to the PL for image registration. Then, the DecodreThread received the image registration results and pushed the color images into a queue. The DpuThread popped the images in the queue to the DPU for the observer and exhibit detection and received the detection results, including the coordinates, sizes, and labels. The SortingThread

 doi:10.20944/preprints202306.0863.v1



sorted the images in the queue according to the index of frames. The GpuThread displayed images at a specific position on the transparent display.

The information fusion system employed the RetinaFace model to detect the face of the observer and the YOLO model to detect the exhibit. The RetinaFace model was pre-trained and supported by the Vitis AI, and it could quickly apply the RetinaFace model on the DPU. The YOLO model for detecting LEGOs was self-trained, and the self-trained model needs to be converted into a DPU-executable xmodel format. The conversion flow is shown in Figure 6. The first step is to adjust the parameter of the YOLO model, as shown in Table 1. The input image size of the DPU was 512 by 512 pixels. The DPU didn't support mish activation function; Thus, the activation function was changed to leaky rectified linear unit activation function. The max-pooling sizes were reduced due to the maximum maxpool size of the DPU was limited to eight. Furthermore, the computing number format of the model was converted from the floating-point format to the integer format by VITIS AI quantizer for acceleration.

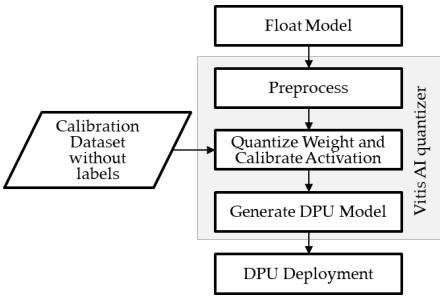


**Figure 6.** Conversion flow for applying a self-trained model to the Deep Learning Processor Unit in Xilinx Zynq UltraScale+ MPSoC Chip.

**Table 1.** Parameter adjustment of the self-trained YOLO model for Xilinx Deep Learning Processor Unit.

| Parameters | Origin | For Xilinx Deep Learning Processor Unit |
|---|---|---|
| Input image weight | 608 | 512 |
| Input image height | 608 | 512 |
| Activation function | mish | leaky |
| Max-pooling size 1 | 9 | 6 |
| Max-pooling size 2 | 13 | 8 |

The image registration function was programmed in C++ code and used the Vitis™ High-Level Synthesis tool to synthesize the function into the registration level (RTL) module for PL implementation. The code is listed in Table 2. The input image size is 640 by 480 pixels. The input parameter includes the intrinsic and intrinsic parameters of the color and depth camera. Lines 10 to 19 of Table 2 describe the coordinate transformation computation from depth camera coordinate to color camera coordinate. Lines 20 and 21 of Table 2 describe the depth data alignment in color camera coordinates.

**Table 2.** C++ code of image registration for High-level RTL synthesis.

```
1  void align_depth_to_color_2(float intrinsics_color[4], float intrinsics_depth[4], float extrinsics_depth_to_color[12],
   float depth_scale ,unsigned short int depth[307200], float depth_of_color[640*480]) {
2  float pd_uv[2];
3  float Pd_uv[3];
4  float Pc_uv[3];
5  unsigned short int pc_uv[2];
6  for(int i =0; i<480; i++) {
7  //#pragma HLS pipeline II = 1
```

 doi:10.20944/preprints202306.0863.v1



```
for(int j=0;j<640;j++) {
//#pragma HLS pipeline II=1
pd_uv[0] = j;
pd_uv[1] = i;
Pd_uv[0] = (pd_uv[0]-intrinsics_depth[2]) * depth[i*640+j]*depth_scale / intrinsics_depth[0];
Pd_uv[1] = (pd_uv[1]-intrinsics_depth[3]) * depth[i*640+j]*depth_scale / intrinsics_depth[1];
Pd_uv[2] = depth[i*640+j]*depth_scale;
Pc_uv[0] = extrinsics_depth_to_color[0] * Pd_uv[0] + extrinsics_depth_to_color[1] * Pd_uv[1] +
extrinsics_depth_to_color[2] * Pd_uv[2] + extrinsics_depth_to_color[3];
Pc_uv[1] = extrinsics_depth_to_color[4] * Pd_uv[0] + extrinsics_depth_to_color[5] * Pd_uv[1] +
extrinsics_depth_to_color[6] * Pd_uv[2] + extrinsics_depth_to_color[7];
Pc_uv[2] = extrinsics_depth_to_color[8] * Pd_uv[0] + extrinsics_depth_to_color[9] * Pd_uv[1] +
extrinsics_depth_to_color[10] * Pd_uv[2] + extrinsics_depth_to_color[11];
pc_uv[0] = (unsigned short int)((intrinsics_color[0] * Pc_uv[0] / Pc_uv[2] + intrinsics_color[2]));
pc_uv[1] = (unsigned short int)((intrinsics_color[1] * Pc_uv[1] / Pc_uv[2] + intrinsics_color[3]));
if(pc_uv[0] < 640 && pc_uv[0] >= 0 && pc_uv[1] < 480 && pc_uv[0] >= 0 )
depth_of_color[pc_uv[1] * 640 + pc_uv[0]] = Pd_uv[2];
}
}
}
```

## 4. Experimental setup and results

The image registration model used 4.46% of the look-up-table (LUT), 3.64 % of the register (REG), and 2.94% of the digital signal processing (DSP) unit in the XCZU9EG. Two DPU units were used for the RetinaFace and YOLO models. The two DPU units used approximately 38% of the LUT, 38% of REG, 62 % of Block RAM (BRAM), and 55% of DSP. Most resources were obviously used for DPU. The total utilization of the information fusion accounted for 50% to 60% of the whole resource of the XCZU9EG. The utilization of the XCZU9EG is listed in Table 3.

**Table 3.** Utilization of the XCZU9EG.

| Name | LUT | LUTAsMem | REG | BRAM | DSP |
|---|---|---|---|---|---|
| DPUCZDX8G_1 | 51,082 | 5,680 | 97,940 | 257 | 690 |
| | 19.69% | 3.98% | 18.86% | 30.82% | 27.38% |
| DPUCZDX8G_2 | 51,308 | 5,680 | 98,128 | 257 | 690 |
| | 19.78% | 3.98% | 18.90% | 30.82% | 27.38% |
| Image registration | 11,568 | 981 | 18,923 | 15 | 74 |
| | 4.46% | 0.69% | 3.64% | 1.80% | 2.94% |
| sfm_xrt_top | 9,638 | 540 | 8,142 | 4 | 14 |
| | 3.71% | 0.38% | 1.57% | 0.48% | 0.56% |
| Total Utilization | 138,219 | 14,139 | 251,993 | 612 | 1,468 |
| | 50.00% | 10.00% | 46.00% | 67.00% | 58.00% |

The Linux system on the ARM side executes the application program to complete the functions of image reading, detecting object frame selection, marking the name, and image output. The interface includes the opening advertisement video, selecting the target object to watch, and jumping to the relevant introduction interface after clicking the display. According to the front and rear depth cameras, the position of the human eye and the exhibit are recognized, respectively. The position of the object on the screen seen by the eyes is calculated through the above relative position formula. The directional touch function was realized in the ARM-based FPGA platform and shown in Figure 7. After clicking the target, the corresponding introduction interface is displayed, as shown in Figure 8.

 doi:10.20944/preprints202306.0863.v1




**Figure 7.** Touch the screen to select the exhibit behind the transparent display.


**Figure 8.** Transparent display shows the introduction overlapped on the exhibit in the view of the observer.

## 5. Conclusions and Discussion

This research uses ARM-based FPGA to implement the computation of virtual-real fusion hardware systems. It also evaluates the acceleration effect of virtual-real fusion execution speed after connecting related hardware acceleration and provides a reference for real-time information fusion system design. This system received image data through the Linux system built on the PS and fed to the PL to execute the deep learning model for object detection. In addition, the alignment module on the PL performed the coordination alignment of color images and depth images. The execution time of the image registration performed by the PL and PS were approximately 3 ms and 1.1 s, respectively. The execution time of the image registration using PL was faster than that using PS. The system calculated the size and position of the display information on the transparent display according to the pixel coordinates and depth values of the human eye and the exhibit. The fusion work took about 47ms to execute on the PC with a GPU RTX2060 and 25ms on the ZCU102 development kit. The

acceleration is nearly 168%. Finally, the results were successfully transplanted into a retail display system for demonstration.

**Supplementary Materials:** This research has no supplementary materials.

**Author Contributions:** Conceptualization, Y.-H.T. and M.O.-Y.; methodology, M.-H.H.; software, M.-H.H.; validation, M.-H.H. and T.-Y.Y.; resources, Y.-H.T. and M.O.-Y.; data curation, M.-H.H.; writing—original draft preparation, T.-Y.Y.; writing—review and editing, Y.-H.T., Y.-J.Y. and M.O.-Y.; visualization, T.-Y.Y; supervision, Y.-H.T., Y.-J.Y. and M.O.-Y.; project administration, Y.-H.T., Y.-J.Y. and M.O.-Y.; funding acquisition, Y.-H.T. and M.O.-Y.; All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data sharing is not applicable.

**Acknowledgments:** This paper was particularly supported by grant MOEA 112-EC-17-A-24-171 from the Ministry of Economic Affairs, Taiwan, the Industrial Technology Research Institute, Taiwan, and National Yang Ming Chiao Tung University, Taiwan.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Y. T. Liu, K. Y. Liao, C. L. Lin, and Y. L. Li, "66-2: Invited Paper: PixeLED display for transparent applications," In Proceeding of SID Symposium Digest of Technical Papers 49(1), 874-875 (May 2018).
2. P. Mohr, S. Mori, T. Langlotz, B. H. Thomas, D. Schmalstieg, and D. Kalkofen, "Mixed reality light fields for interactive remote assistance," In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, 1-12, (April, 2020).
3. A. Shawahna, S. M. Sait, and A. El-Maleh, "FPGA-based accelerators of deep learning networks for learning and classification: A review," in IEEE Access, 7, (2018), pp. 7823-7859.
4. Lauro Rizzatti, "A Breakthrough in FPGA-Based Deep Learning Inference," in EEWeb, 2018
5. C. Farabet, C. Poulet, J. Y. Han, and Y. LeCun, "Cnp: An FPGA-based processor for convolutional networks," in Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on. IEEE, 2009, pp. 32–37.
6. E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in Advances in neural information processing systems, 2014, pp. 1269–1277.
7. Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in Advances in neural information processing systems, 1990, pp. 598–605.
8. S. J. Hanson and L. Y. Pratt, "Comparing biases for minimal network construction with back-propagation," in Advances in neural information processing systems, 1989, pp. 177–185.
9. B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in Advances in neural information processing systems, 1993, pp. 164–171.
10. S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv preprint arXiv:1510.00149, 2015.
11. S. Gupta et al., "Deep learning with limited numerical precision," in Proc. 32nd Int. Conf. Mach. Learn.(ICML), Jun. 2015, pp. 1737–1746.
12. K. Guo et al., "Angel-Eye: A complete design flow for mapping CNN onto embedded FPGA," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 37, no. 1, pp. 35–47, Jan. 2018.
13. S. Migacz. 8-Bit Inference With TensorRT. Accessed: May 2017. [Online]. Available: https://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf