1. Introduction
In this report, we will develop a realtime vision system based on FPGA's. The goal of this report is to demonstrate the performance of the developed FPGA-based vision system experimentally.

2. FPGA-based Vision System
In many applications including object handling and human identification, vision systems are required to process successive images in realtime. In addition, vision systems must be flexible to cope with various environments. We will develop an FPGA-based vision system to achieve both realtimeness and flexibility. Figure 1 shows a prototype of the FPGA-based vision system. This prototype consists of an FPGA, a video decoder, a video encoder, SRAM, and a PC interface.

We will introduce C/C++-based description for the design of logic circuits of vision algorithms. Description of logic circuits in C/C++ language connects the development of algorithms to the design of logic circuits seamlessly. In addition, we have to repeat the description of a logic circuit and its verification during the design of a vision processing circuit. Designing of logic circuits in C/C++ language can hasten this repetition, which reduces the time for the design. Thus, we have introduced the design of logic circuits in C/C++ language. We have introduced System-Compiler, which can translate C/C++ description into HDL description. As described in Table 1, logic circuits can be described using classes and class methods in C/C++ language.

3. Experimental Evaluation

Image Gravity Center We have implemented the logic circuit to compute the image gravity center on the FPGA-based realtime vision system. The circuit computes the image gravity center of $256 \times 256$[pixel] region at the center of an input image. FPGA consumes 2% of total gates. We will drive the circuit at 12MHz, which is equal to the frequency of image capturing. Since one division requires 32 clocks, the computation of the image gravity center finishes in about 2.7[msec] after the image capturing. Figure 2 describes a result of the computation of the center. Vertical and horizontal lines in each image indicate the position of the center. The gravity center of successive images can be computed appropriately in realtime.

Radial Projection We have implemented the logic circuit to detect the planar motion of an object on the FPGA-based realtime vision system. The circuit computes the image gravity center and the radial projection of $256 \times 256$[pixel] region at the center of an input image. Rotational angle is expressed by 144 discrete values at the intervals of 2.5°. Signed distance $\rho$ takes a discrete value between $-127$[pixel] and $127$[pixel] at the intervals of 1[pixel]. Then, the FPGA consumes 77% of total gates. The circuit can be driven at 24MHz. Since the voting for one pixel requires 2 clocks, the computation of Hough transform needs $2 \times 256 \times 256$ clocks, which corresponds to 5.46[msec]. The computation finishes simultaneously with the capturing process of an image. Fig. 4 shows the computation of Hough transform. As shown in the figure, the FPGA-based vision system can compute Hough transforms of successive images in realtime.

4. Concluding Remarks
In this report, we have described an FPGA-based realtime vision system and three vision algorithms implemented on the system. We can reduce the latency as well as the computation time by implementing vision algorithms on the FPGA-based realtime vision system. Matched filter, phase-only correlation method[1], and Radon-Fourier transformation method[2] can detect the position and the orientation of a planar motion object robustly against the change of illumination and occlusion. We will implement these algorithms on the FPGA-based vision so that they can be performed in realtime.

References
Figure 1: Prototype of FPGA-based vision system

Table 1: Description of selector in SystemCompiler

```cpp
class Selector {
public:
    uint1 reg;
    Selector ( void ) { reg = 0; }
    void run ( uint1, uint1,
               uint1, uint1, uint1, uint1& );
};
void Selector::run(uint1 clk, uint1 rst,
                   uint1 a, uint1 b, uint1 s, uint1&y)
{
    uint1 temp = s ? a : b;
    if ( infer_clock(clk) ||
         infer_reset(!rst) ) {
        if (!rst) { reg = 0; }
        else { reg = temp; }
        y = reg;
    }
}
```

Figure 2: Detection of image gravity center

Figure 3: Detection of position and orientation

Figure 4: Realtime computation of Hough transform