Color-Target-Tracking Fruit Ninja Proposal

Peiheng Li(pl2978), Chengrui Li(cl4750), Yitong Bai(yb2636)

March 11, 2026

1. Project Overview

This project proposes an interactive *Fruit Ninja* style game implemented on the DE1-SoC platform using both Verilog and C. The system will use an OV7670 camera module to capture live video, detect a brightly colored target held by the player (such as a colored glove or a bright color ball), track its center position in real time, and interpret fast motion as a slashing action. The player will then use this motion to slice virtual fruits displayed on the screen.

The main idea of the project is to transform a difficult hand-gesture-recognition problem into a simpler and more practical color target tracking problem. Instead of recognizing detailed hand shapes or finger motions, the system only needs to detect and follow a distinctive color region. This reduces algorithmic complexity while preserving an intuitive and visually appealing game experience. The project's workflow is:

- **FPGA / Verilog** will handle real-time image processing tasks such as color segmentation and target localization.
- **HPS / C** will handle trajectory analysis, slash detection, game state updates, scoring, and fruit collision logic.

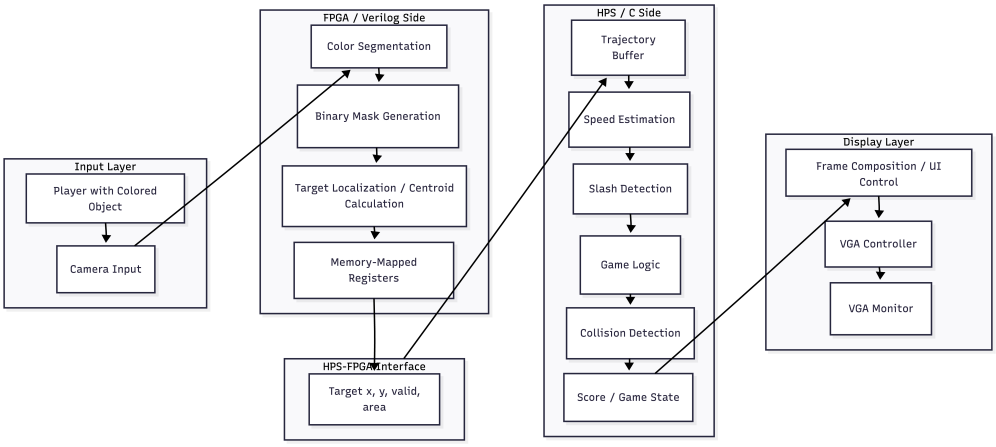

Figure 1: System Diagram

2. System Architecture

2.1 Camera Input Module

The system uses an **OV7670 CMOS camera module** as the image sensor. The OV7670 provides a parallel pixel output interface and supports resolutions up to 640×480 at approximately 30 frames per second.

The camera sends pixel data to the FPGA through an 8-bit parallel data bus along with synchronization signals including *PCLK*, *VSYNC*, and *HREF*. The FPGA captures these signals and reconstructs the video frame stream.

The OV7670 is configured through an *I2C* control interface. The FPGA initializes the camera parameters such as color format and resolution before the start of the video processing pipeline.

Once configured, the camera continuously streams pixel data into the FPGA where the image processing pipeline begins.

2.2 Color Segmentation Module (Verilog)

This module examines each input pixel and determines whether it belongs to the selected target color. The output is a binary mask indicating target pixels and background pixels.

2.3 Target Localization Module (Verilog)

This module computes the center of the detected target region. It accumulates the coordinates of valid target pixels and estimates the centroid at the end of each frame. The output includes:

- center position (x, y)
- validity flag
- optional target area information

2.4 HPS–FPGA Communication Module

This module transfers the target coordinates and status information from FPGA logic to the HPS. The C program running on the HPS reads these values through memory-mapped registers.

2.5 Trajectory Analysis Module (C)

This module stores the most recent target positions and computes motion speed and direction. If the movement exceeds a threshold, the system labels it as a slash.

2.6 Game Logic Module (C)

This module manages the state of the game, including fruit generation, object motion, slicing events, score updates, and game-over conditions.

2.7 Display Module

This module outputs the visual result to the screen. It may include the game background, fruits, score, and optionally the tracked target or slash trajectory for debugging and demonstration.

This flow allows the system to respond to user motion in real time while keeping the implementation modular.

3. Resource Budgets

The system must process camera frames in real time while tracking a colored target and updating the game state.

We assume the OV7670 camera operates at a moderate resolution of either 320×240 or 640×480. Since the FPGA processing pipeline operates on a streaming pixel basis, the entire frame does not need to be stored in memory simultaneously.

For color segmentation, only simple threshold comparisons are required per pixel. Each pixel consists of three 8-bit color channels (RGB), so each pixel requires 24 bits of input data. The segmentation module outputs a single-bit mask indicating whether the pixel belongs to the target color.

For centroid calculation, the FPGA accumulates:

- Sum of x coordinates
- Sum of y coordinates
- Number of detected pixels

Assuming a 640×480 frame, the maximum coordinate values require:

- x coordinate: up to 640 (10 bits)
- y coordinate: up to 480 (9 bits)

To avoid overflow during accumulation, approximately 32 bits are allocated for each accumulator:

$$3 \times 32 = 96bits$$

For trajectory detection on the HPS side, storing the last 16 positions requires:

$$16 \times (10 + 9) = 304bits$$

which corresponds to less than 40 bytes.

Game objects are lightweight. If up to 10 fruits exist simultaneously and each stores position, velocity, and state:

$$10 \times 64bits \approx 80bytes$$

These requirements easily fit within the available FPGA and HPS resources.

4. Hardware/Software Interface

The system uses a hardware/software co-design approach where the FPGA performs real-time image processing and the HPS manages higher-level game logic.

Communication between FPGA and HPS will use memory-mapped registers. The FPGA writes target detection results into shared registers that the HPS reads using a C program.

Each frame, the FPGA provides:

- Target center position (x, y)
- Target detection validity flag
- Optional target area information

The HPS updates the motion trajectory buffer and determines whether a slicing gesture has occurred.

If a slash is detected, the HPS updates the game state and checks for collisions between the slicing trajectory and active fruit objects.

5. Milestones

The project will be completed in several stages:

1. Implement the OV7670 camera input pipeline and verify video streaming.
2. Develop the color segmentation module to detect the colored target.
3. Implement the centroid calculation module and verify target coordinate extraction.
4. Implement FPGA–HPS communication and verify coordinate transfer.
5. Develop trajectory analysis in C for slash detection.
6. Implement the core game logic including fruit spawning and collision detection.
7. Implement the display module and render the game interface.
8. Integrate the full system and perform final testing.

6. Expected Outcome

By the end of this project, we expect to demonstrate a working prototype of a camera-based interactive game on the DE1-SoC platform. The final system should allow a player to control a virtual slicing action using a colored target in real time.

Beyond the game itself, the project will show how FPGA-based image processing and C-based application logic can be combined effectively in a single embedded system.