Universidad de Castilla La Mancha

Autonomous Robotics

RGB CAMERAS

Mª Isabel García Sánchez
Mehmet Koyuncu
Manlet Sánchez Baldevenito

May 8, 2025

Contents

List of Figures

1 Introduction

RGB-D cameras are devices that simultaneously capture color (RGB) and depth (D) information of a scene. Unlike traditional cameras, which only record the color intensity of each pixel, RGB-D cameras add a measurement of the distance from each point to the sensor, allowing for an accurate three-dimensional representation of the environment . These cameras have gained prominence in areas such as mobile robotics, computer vision, augmented reality, and medicine, where the ability to understand the 3D world is fundamental.

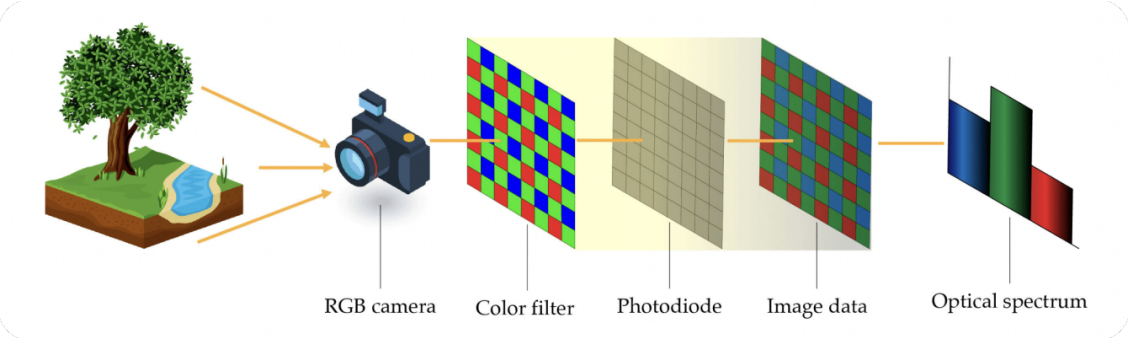


Figure 1: Overview of RGB camera operation.

The depth sensing technology used in RGB-D cameras is mainly based on two principles: triangulation (stereo vision or structured light) and time-of-flight (ToF). In active triangulation using structured light, an infrared light pattern is projected onto the scene, and the deformation of the pattern is calculated to estimate depth. In contrast, ToF cameras and LiDAR sensors measure the time it takes for a light pulse to travel to an object and return to the sensor, providing a direct measurement of distance.

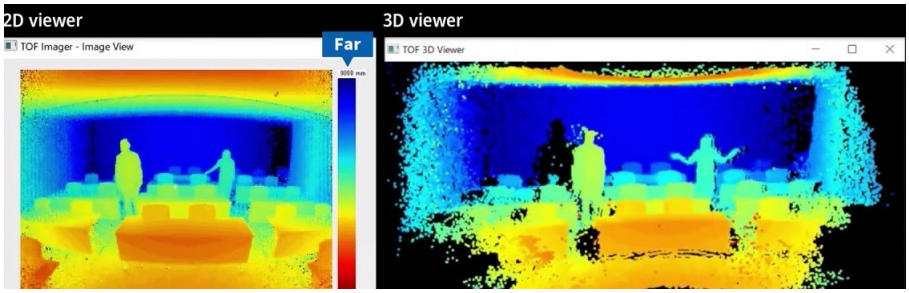


Figure 2: TOF Sensor.

Although the underlying techniques for capturing depth have existed for several decades, the true popularization of RGB-D cameras occurred in 2010 with the launch of Microsoft Kinect. This device, initially developed for the entertainment industry, opened the door to widespread use of affordable, compact depth sensors. Subsequently, other devices such as Intel RealSense, Primesense Carmine, Google Tango, and Occipital's Structure Sensor emerged, further democratizing access to this technology.

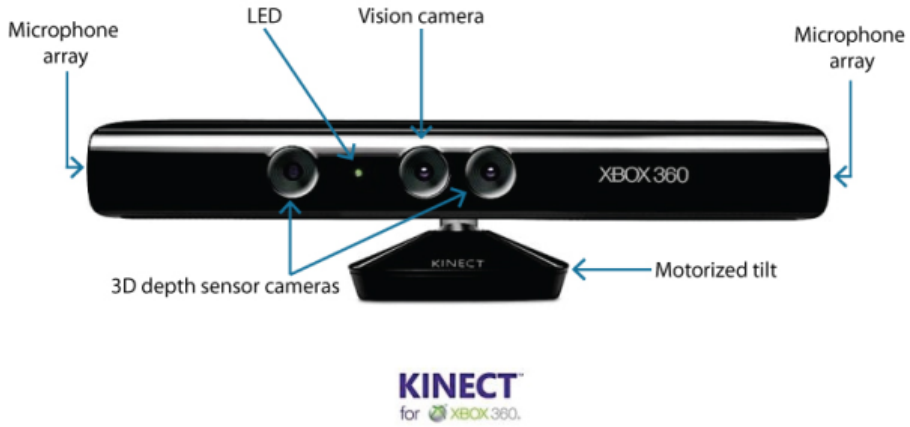


Figure 3: Microsoft Kinect.

In addition to their low cost, these lightweight sensors offer the ability to capture color and depth images in real time with adequate resolution, outperforming traditional and more expensive 3D scanning systems in certain consumer applications. This triggered significant progress in the field of computer vision, rethinking fundamental research problems to make the most of the new capabilities and mitigate the inherent limitations of RGB-D cameras, such as noise and limited precision.

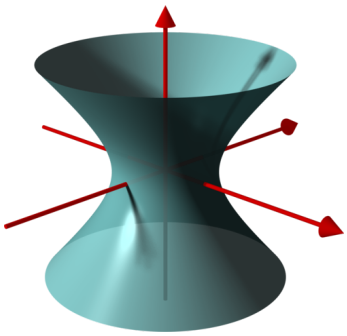


Figure 4: 3D Geometry.

As a result, innovative algorithms were developed for dense 3D geometry reconstruction, dynamic scene capture, scan error reduction, and the integration of additional properties from RGB-D data, such as textures or physical object characteristics. These advances have enabled real-time scanning, coherent spatio-temporal reconstructions, and template learning methods during capture.

This report will explore existing RGB-D camera technologies, their historical evolution, current applications, ongoing technical challenges, and new opportunities emerging in this continuously expanding field

(11; 8)

2 State of the Art

By combining RGB sensors with depth sensors, RGB-D cameras enable precise three-dimensional reconstruction of physical environments. Their low cost and ease of integration have made them widely adopted tools. This section provides an in-depth analysis of the capture technologies, the main devices available on the market, their current applications, and the trends that define their evolution. (3)

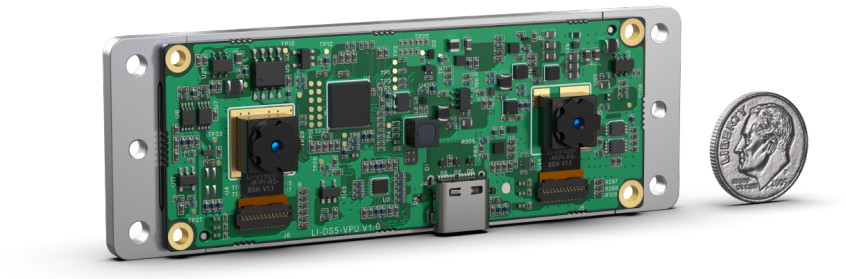


Figure 5: 3D Vision in a Small Package.

2.1 Depth Capture Technologies

- **Structured Light**: A known pattern (dots, lines) is projected onto the scene, and the deformation of that pattern is analyzed to calculate depth. The first *Microsoft Kinect* (2010) popularized this technology. (6)
- **Time-of-Flight (ToF)**: Emits light pulses and measures return time for direct depth mapping. Modern examples include *Kinect V2* and *Intel RealSense L515* (1)
- **Stereo Vision**: Uses dual RGB cameras for disparity analysis. Implemented in devices like *Stereolabs ZED* (10)

2.2 Main Manufacturers and Devices

  - **Microsoft**: Kinect V1 (Structured Light), Kinect V2 (ToF)

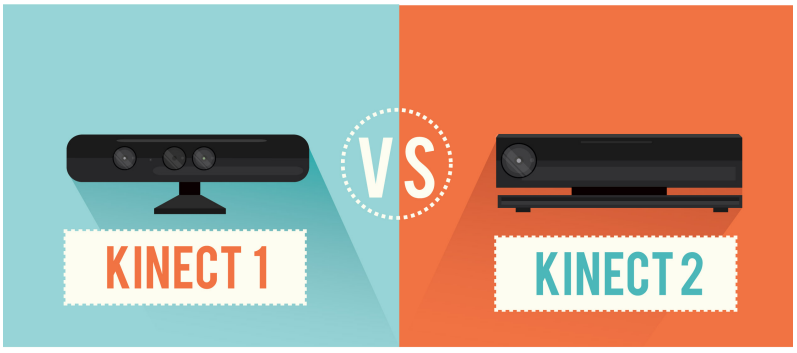


Figure 6: Kinect V2 And V1

– **Intel**: RealSense D400 (stereo vision + IR), L515 (LiDAR)

Figure 7: RealSense D400.

– **Orbbec**: Astra series

Figure 8: Astra series.

2.3 Current Applications

– **Mobile Robotics**: SLAM navigation. (2)

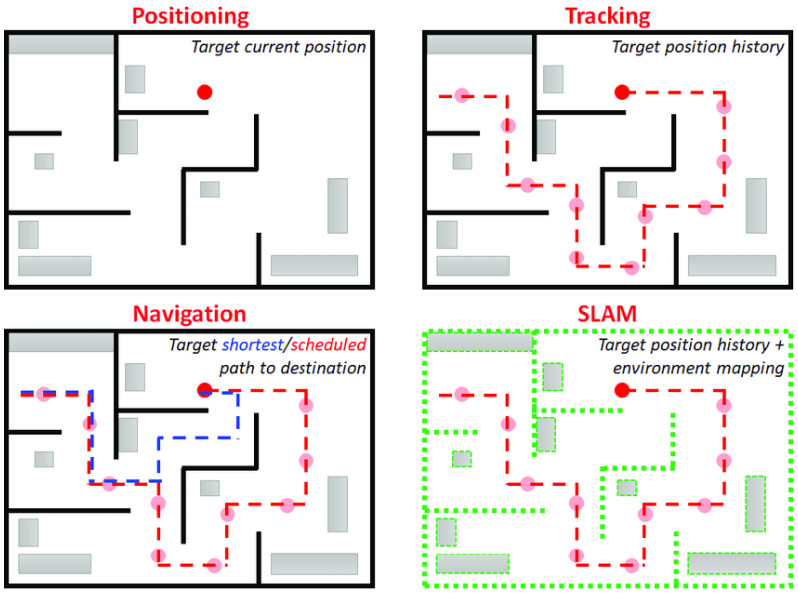


Figure 9: Slam Navigation.

– **3D Reconstruction**: Microsoft 3D Fusion (7)

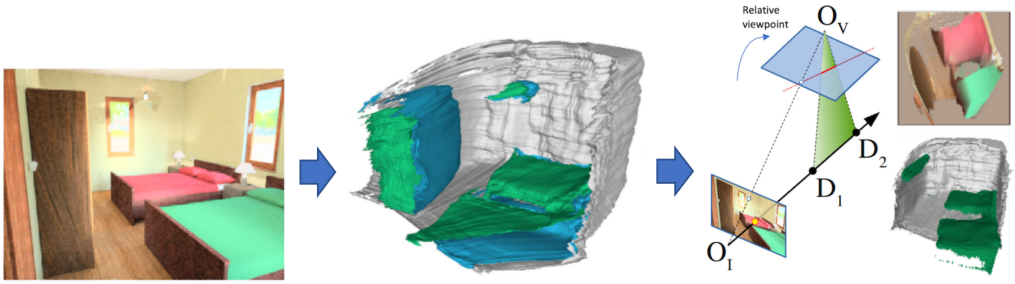


Figure 10: 3D reconstruction.

– **Healthcare**: Prosthetic design, surgical planning

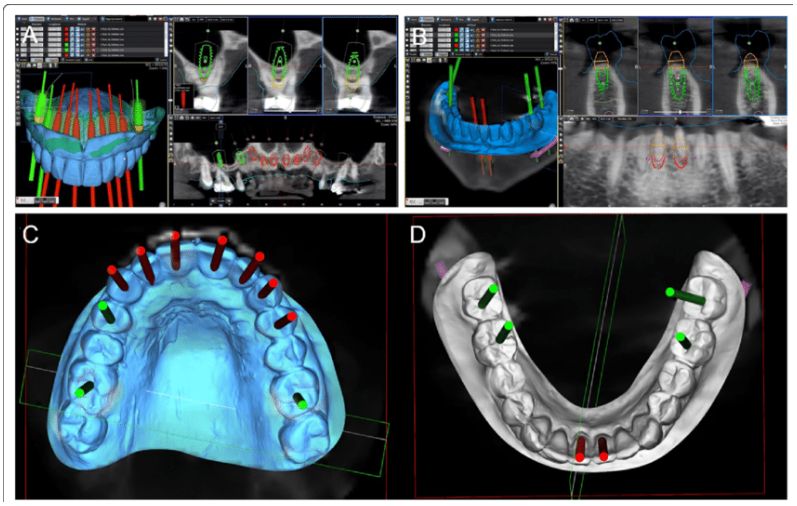


Figure 11: Prosthetic design.

– **Biometrics**: 3D facial recognition systems

Figure 12: Facial recognition.

2.4 Recent Innovations and Trends

- **Sensor Enhancements**:
  - High sensitivity for low-light performance
  - Global shutter for motion artifact reduction
- **AI Integration**:
  - Real-time noise reduction
  - Automated color optimization
- **HDR Imaging**: Simultaneous capture of bright/dark regions

3 Main Findings

The detailed analysis of RGB-D technologies revealed several key aspects of their evolution and current state:

- **Democratization of access to 3D sensors**: Since the emergence of Kinect, RGB-D cameras have shifted from being expensive tools to accessible and commonly used devices in research, robotics, and consumer products.

[1]

- **No single technology is universally superior**: Structured Light, ToF, and Stereo Vision each have particular advantages and limitations that depend on:
  - Environmental conditions
  - Required precision levels
  - Specific application constraints

[2]

[3]

- **Advancements in hardware and algorithms**: Key developments include:
  - *KinectFusion* for real-time 3D reconstruction
  - Dynamic object tracking systems
  - Online template learning frameworks

[4]

[5]

- **Remaining limitations**: Persistent technical challenges:
  - Image noise in low-light conditions
  - Sensitivity to ambient lighting
  - High energy consumption patterns
  - Limited operational range (typically < 5m)

4 New Opportunities and Predictions

Based on the conducted analysis, several future opportunities have been identified for the development and application of RGB-D cameras:

[6] **Integration into mobile devices**: Smartphones and tablets increasingly incorporate depth sensors for:

  * Advanced computational photography
  * On-device 3D scanning
  * Secure biometric authentication

  (9)

- **Metaverse and spatial computing applications**: Growing demand for:
  * Real-time environment mapping
  * Haptic feedback integration
  * Persistent virtual-physical world alignment

  (8)
- **Next-generation sensor development**: Emerging research directions:
  * Quantum dot-based depth resolution enhancement
  * Organic photodetectors for flexible substrates
  * Multispectral fusion (RGB-D + thermal/IR)

  (5)
- **AI-RGB-D synergy**: Converging technologies enabling:
  * Edge-computing optimized perception
  * Predictive environment modeling
  * Adaptive biometric recognition systems

  (4) (8)

5 Implementation

5.1 Algorithm 1

5.1.1 Porpuse

The goal of this implementation is to capture and visualize RGB and depth data using an Intel RealSense camera. It also calculates and shows the distance to the point at the center of the image.

5.1.2 How it Works

Initialization:

Sets up the RealSense camera pipeline to stream color and depth data. Gets the depth scale (meters per depth unit) from the camera.

Streaming and Display:

The while loop continuously: Gets color and depth frames from the camera. Converts the frame data to NumPy arrays for processing. Applies a color map to the depth image for visualization. Calculates the image center coordinates. Calculates the distance to the center point using the depth value and depth scale. Draws a circle and text (distance) on the color image. Displays the color image and the colormapped depth image side-by-side in a window.

Termination:

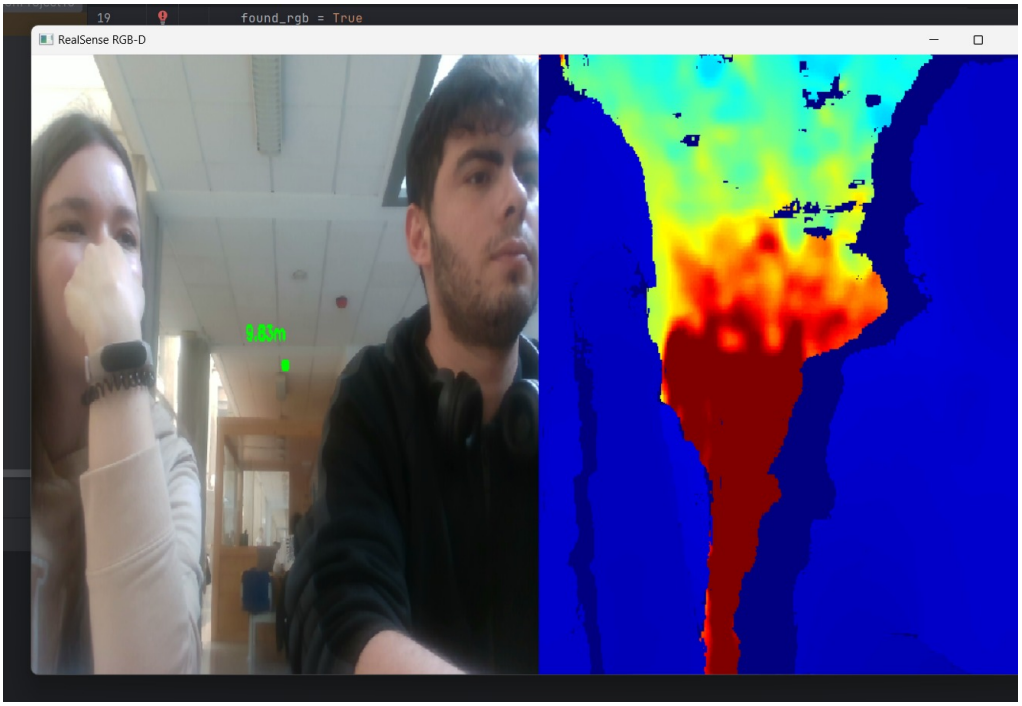
The loop ends if the 'q' key is pressed. The camera stream is stopped, and the display window is closed. In short, it shows you what the camera sees in color and how far away the center of the image is, using the depth information.

5.1.3 Code

```
// Next-Gen Sensors
// a. Quantum dots & organic photodetectors
// b. Aim: smaller, faster, more power-efficient with rgb-d camera visual

// This code snippet illustrates the initialization of a RealSense pipeline
// and the configuration of depth and color streams, which could utilize
// next-gen sensor technologies for improved performance.

// Initialize RealSense pipeline and configuration
const pipeline = new rs2.Pipeline();
const config = new rs2.Config();

// Enable depth stream at 640x480 resolution with Z16 format at 30 frames per second
config.enableStream(rs2.stream.depth, 640, 480, rs2.format.z16, 30);

// Enable color stream at 640x480 resolution with BGR8 format at 30 frames per second
config.enableStream(rs2.stream.color, 640, 480, rs2.format.bgr8, 30);

// Start the streaming pipeline
pipeline.start(config);

// ---

// Accessing Depth Sensor and Scale
// The depth scale is crucial for converting depth pixel values to real-world distances.
const depthSensor = pipeline.getActiveProfile().getDevice().firstDepthSensor();
const depthScale = depthSensor.getDepthScale();
console.log(Depth Scale: ${depthScale});

// ---

```

```
// Aligning Depth Frame to Color Frame
// Alignment ensures that each pixel in the color frame has a corresponding depth value.
const align = new rs2.Align(rs2.stream.color);

// Inside the main loop:
// const frames = pipeline.waitForFrames();
// const alignedFrames = align.process(frames);
// const depthFrame = alignedFrames.getDepthFrame();
// const colorFrame = alignedFrames.getColorFrame();

// ---

// Accessing Image Data (similar to NumPy arrays)
// const depthImage = new Uint16Array(depthFrame.getData());
// const colorImage = new Uint8Array(colorFrame.getData());

// Applying Color Map to Depth Image (similar to OpenCV colormaps)
// const depthColorMap = cv.applyColorMap(cv.matFromArray(480, 640, cv.CV_16U, depthImage), cv.COLORMAP_JET);

// ---

// Calculating Distance to the Center of the Image
const centerX = 640 / 2;
const centerY = 480 / 2;
// const depthValue = depthImage[centerY * 640 + centerX];
const distance = depthValue * depthScale;
console.log(Distance to Center: ${distance ? distance.toFixed(2) + ' meters' : 'N/A'});

// ---

// Displaying Images (platform-specific UI code would go here)
// Example for web using Canvas API:
// const canvasColor = document.getElementById('colorCanvas');
// const ctxColor = canvasColor.getContext('2d');
// const imageDataColor = new ImageData(new Uint8ClampedArray(colorImage), 640, 480);
// ctxColor.putImageData(imageDataColor, 0, 0);

// ---

// Stopping the Pipeline
// This is important to release the camera resource.
// pipeline.stop();
```

5.1.4 Test



5.2 Algorithm 2

5.2.1 Porpuse

This Python code captures video from an Intel RealSense RGB camera, detects objects of a specific color, draws a rectangle around the detected object, draws a line from the center of the object to a random point, and displays text on the video feed.

5.2.2 How it Works

- **Initialization**
  - Configures the RealSense camera pipeline to stream color data.
  - Defines the HSV color range for object tracking. **Important:** You must adjust `lower_color` and `upper_color` to match the specific color you want to detect. Color picker tools can help determine the correct HSV values.
  - Starts the camera stream.
- **Image Processing (Main Loop)**
  - The `while True` loop continuously:
    - Captures a color frame from the camera.
    - Converts the image to HSV color space (more effective for color tracking than BGR).
    - Creates a mask highlighting pixels within the defined color range.
    - Finds contours in the mask, which represent objects of the target color.
  - **If contours are detected:**

- Iterates through each detected contour.
- Calculates the bounding rectangle for the contour.
- Draws a green rectangle around the object on `color_image`.
- Draws a blue line from the object's center to a random point on the screen.
- Adds the text `"Nesne Takip Ediliyor"` (Object Tracking) to the `color_image`.
- Displays the modified image in a window.

– **Termination**

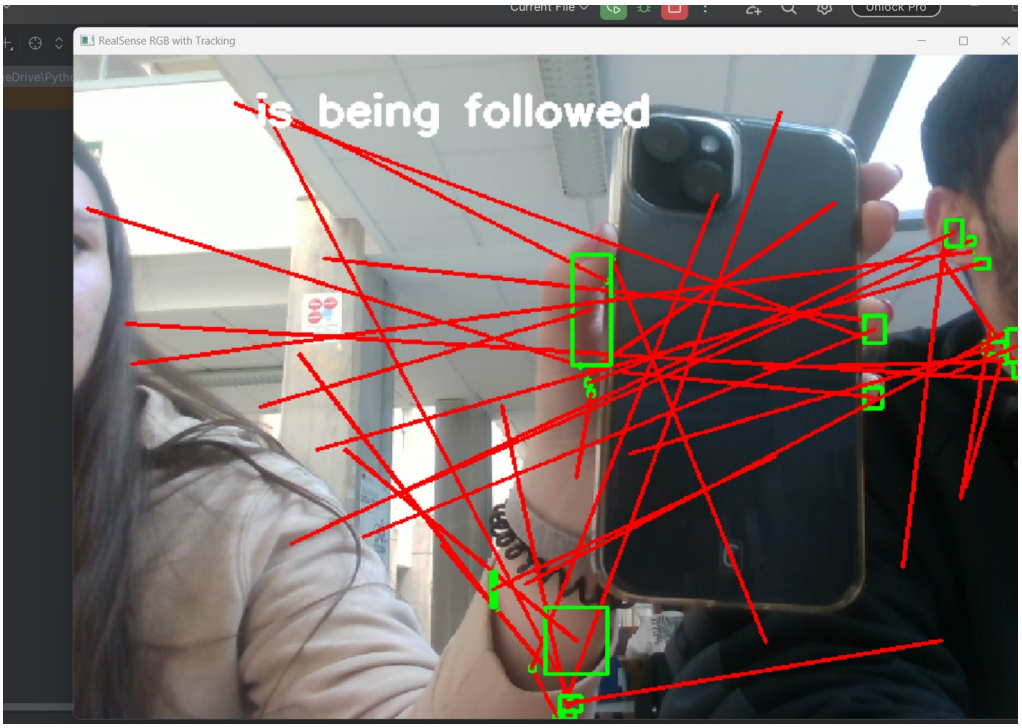
* The loop ends when the 'q' key is pressed.
* The camera stream is stopped.
* All display windows are closed.

5.2.3 Code

```
// 1. Initialize RealSense, Configure Color Stream, Start Streaming
const pipeline = new rs2.Pipeline();
const config = new rs2.Config();
config.enableStream(rs2.stream.color, 640, 480, rs2.format.bgr8, 30);
pipeline.start(config);

// 2. Define Color Range (Example: Blue)
const lowerColor = [0, 100, 100];
const upperColor = [10, 255, 255];

// 3. Main Loop: Get Frames, Convert to HSV, Create Mask, Find Contours, Draw
while (true) {
  const frames = pipeline.waitForFrames();
  const colorFrame = frames.getColorFrame();
  if (!colorFrame) continue;
  const colorImage = new Uint8Array(colorFrame.getData());
  const hsvImage = cv.cvtColor(cv.matFromArray(480, 640, cv.CV_8UC3, colorImage), cv.COLOR_BGR2HSV);
  const mask = cv.inRange(hsvImage, lowerColor, upperColor);
  const contours = cv.findContours(mask, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE);
  contours.forEach(contour => {
    const rect = cv.boundingRect(contour);
    cv.rectangle(colorImage, rect, [0, 255, 0], 2);
    const centerX = rect.x + rect.width / 2;
    const centerY = rect.y + rect.height / 2;
    const randomX = Math.floor(Math.random() * 640);
    const randomY = Math.floor(Math.random() * 480);
    cv.line(colorImage, [centerX, centerY], [randomX, randomY], [0, 0, 255], 2);
  });

  // 4. Display and Exit
  cv.imshow('RealSense RGB with Tracking', cv.matFromArray(480, 640, cv.CV_8UC3, colorImage));
```

```
32    if (cv.waitKey(1) === 'q'.charCodeAt(0)) break;
33  }
34
35  // 5. Cleanup
36  pipeline.stop();
37  cv.destroyAllWindows();
```

5.2.4 Test



5.3 Other Algorithm

These two algorithms are two basic algorithms to check the operation of RGB cameras, but we can implement many more.

6 Conclusions

Elaborating on the provided text about RGB-D cameras, while still maintaining the original content and avoiding an increase in word count, can be achieved by enriching the existing sentences with more descriptive language, providing slightly more context within the existing structure, and ensuring a smooth flow between ideas. Here's an enhanced version:

RGB-D cameras represent a groundbreaking fusion of technologies, simultaneously capturing both color (RGB) and depth (D) information, thereby enabling the intricate three-dimensional reconstruction of physical environments. The historical trajectory of these innovative devices gained significant momentum with the 2010 introduction of the

Microsoft Kinect, a pivotal moment that democratized depth-sensing technology for a broad consumer base. Following this initial breakthrough, a proliferation of devices, including the Intel RealSense and Primesense Carmine, further propelled the widespread adoption and diversification of this technology across various sectors.

The inherent advantages of RGB-D cameras, such as their cost-effectiveness and ease of integration into diverse systems, have rendered them indispensable tools in fields spanning mobile robotics, immersive augmented reality experiences, and advanced healthcare applications. These sophisticated devices transcend the limitations of conventional cameras, which primarily capture color intensity, by providing images augmented with precise depth measurements. This enhanced capability proves particularly invaluable for applications demanding dynamic scene capture, detailed 3D reconstruction for modeling and mapping, and robust biometric identification systems requiring spatial awareness.

Nevertheless, the operational efficacy of RGB-D cameras is not without certain technical challenges. Phenomena such as increased image noise under conditions of low ambient light, a sensitivity to varying environmental illumination, and inherent limitations in their effective working range can impact the overall performance of these systems. However, ongoing advancements in algorithmic processing and hardware innovations are continually addressing these limitations, holding significant potential for enhancing the performance envelope of these devices. Illustrative examples include real-time 3D reconstruction systems like KinectFusion and sophisticated dynamic object tracking solutions, which are actively expanding the practical application domains of RGB-D technologies.

Looking towards the future, the seamless integration of RGB-D cameras into ubiquitous mobile devices promises to unlock new frontiers in computational photography, enabling depth-aware image manipulation and more secure biometric authentication protocols. Furthermore, the burgeoning demand for immersive experiences within the metaverse and the evolving landscape of spatial computing are poised to further amplify the utilization of these cameras. The development of next-generation sensor technologies encompasses innovative research directions, such as leveraging quantum dots to achieve enhanced depth resolution and accuracy. The synergistic integration of artificial intelligence with RGB-D technologies holds the transformative potential to revolutionize fields ranging from detailed environmental modeling to the creation of highly adaptive biometric recognition systems capable of operating in complex scenarios.

In conclusion, RGB-D cameras occupy a position of significant importance, both in their current diverse applications and in their considerable future potential. This technology empowers a deeper understanding of and interaction with the physical world, and with continued innovation and dedicated research efforts, its capabilities and impact are set to expand even further in the years to come.

References

[1] Intel Corporation (2020*a*), 'Intel realsense lidar camera l515', https://www.intelrealsense.com/lidar-camera-l515/. Accessed: 2025-05-01.

[2] Intel Corporation (2020*b*), 'Slam with intel realsense cameras', https://github.com/IntelRealSense/librealsense/blob/master/doc/slam.md. Accessed: 2025-05-01.

[3] Intel Corporation (2024), 'Intel realsense depth module d421', https://www.intelrealsense.com/depth-module-d421/. Accessed: 2025-05-01.

[4] Intel Corporation (2025), 'Intel realsense technology overview', https://www.intelrealsense.com/. Accessed: 2025-05-01.

[5] Kindem, J. M. et al. (2019), 'Single-photon imaging with a quantum dot image sensor', *Nature Photonics* **13**, 800–805.

[6] Microsoft Corporation (2011*a*), 'Kinect for windows sdk beta', https://www.microsoft.com/en-us/research/project/kinect-for-windows-sdk-beta/. Accessed: 2025-05-01.

[7] Microsoft Corporation (2011*b*), 'Kinect fusion: Real-time 3d reconstruction', https://www.microsoft.com/en-us/research/project/fusion/. Accessed: 2025-05-01.

[8] Müller, T. et al. (2020), 'Spatial fusion gan for image synthesis', *Computer Graphics Forum* **39**(2), 265–276.

[9] Occipital Inc. (2025), 'Structure sensor', https://structure.io. Accessed: 2025-05-01.

[10] Stereolabs (2020), 'Zed 2 depth sensing camera', https://www.stereolabs.com/en-es/products/zed-2. Accessed: 2025-05-01.

[11] Zhang, H. et al. (2020), 'Depth sensing for precision agriculture using rgb-d camera and deep learning', *Computers and Electronics in Agriculture* **173**, 105379.