

# VisionAgent Workflow and Results

I am truly amazed by the planning and actions taken by VisionAgent...

**Inputs: Image given below**

**Query: Did you see anything unusual in this image?**



Vision Agent (VA) is thinking and planning to execute the assigned task.

Followings are the detailed steps VA takes:

Step 10: First, I need to load the image and get a general understanding of what might be unusual in it. I'll use claude35\_vqa to get a description and ask for suggestions on how to analyze the image for unusual elements.

Code to Execute:

===== Code =====

```
1
2 image = load_image('tmpibfda2c1.jpg')
3 claude35_vqa('Can you describe this image in detail? Do you notice anything unusual or out of place?', [image])
4 suggestion('How can I identify unusual or anomalous elements in this image?', [image])
5
```

Code Execution Output (32.72s): ----- stdout -----

[claude35\_vqa output]

The image shows an elderly man on the floor of what appears to be a home interior. He is wearing a white polo shirt and seems to be in a position of distress, with his hands planted on the wooden floor as if he has fallen or is struggling to get up.

Near him on the floor is a walking cane, which suggests he may use it for mobility assistance. The setting appears to be a bedroom or living area, with a white cabinet or nightstand visible, as well as part of what looks like a bed with a blue and white patterned cover.

The unusual aspect of this image is the concerning position of the man on the floor, which implies he may have fallen and be unable to get back up easily. The presence of the cane nearby adds to this interpretation, suggesting a possible accident or medical issue.

This scene highlights potential safety concerns for elderly individuals living alone and the importance of having systems in place to call for help in such situations. It's a sobering reminder of the challenges some older adults face with mobility and independence.

[end of claude35\_vqa output]

----- stderr -----

Step 9: Based on Claude's description, the unusual thing in the image is an elderly person who appears to have fallen on the floor with a walking cane nearby. This is a potential medical emergency situation. Let me get a tool recommendation for detecting people in distress or unusual poses.

Code to Execute:

===== Code =====

```
1
2 get_tool_for_task('Detect a person in an unusual position or fallen on the floor', [image])
```

Code Execution Output (91.05s): ----- stdout -----

[get\_tool\_for\_task output]

The owl\_v2\_image tool provided the most accurate detection for this request. It correctly identified a 'person in unusual position' with a bounding box covering the fallen man.

Other tools like countgd\_object detection and florence2\_sam2\_image detected the person but didn't capture the 'unusual position' aspect as well. The owl\_v2\_image prompt of 'person in unusual position, fallen person' worked well to detect this specific scenario.

Tool Documentation:

owl\_v2\_image(prompt: str, image: numpy.ndarray, box\_threshold: float = 0.1, fine\_tune\_id: Optional[str] = None) -> List[Dict[str, Any]]:

'owl\_v2\_image' is a tool that can detect and count multiple objects given a text prompt such as category names or referring expressions on images. The categories in text prompt are separated by commas. It returns a list of bounding boxes with normalized coordinates, label names and associated probability scores.

Returns:

List[Dict[str, Any]]: A list of dictionaries containing the score, label, and bounding box of the detected objects with normalized coordinates between 0 and 1 (xmin, ymin, xmax, ymax). xmin and ymin are the coordinates of the top-left and xmax and ymax are the coordinates of the bottom-right of the bounding box.

Example

-----

```
>>> owl_v2_image("car, dinosaur", image)
[
  {'score': 0.99, 'label': 'dinosaur', 'bbox': [0.1, 0.11, 0.35, 0.4]},
  {'score': 0.98, 'label': 'car', 'bbox': [0.2, 0.21, 0.45, 0.5]},
]
```

[end of get\_tool\_for\_task output]

----- stderr -----

Step 8: Now that I have confirmed the unusual situation - an elderly person fallen on the floor - I will use owl\_v2\_image to detect this specifically and visualize the detection to confirm.

Code to Execute:

===== Code =====

```
1
2 # Detect fallen person and related objects
3 detections = owl_v2_image("fallen person, walking cane", image)
4
5 # Visualize the detections
6 image_with_boxes = overlay_bounding_boxes(image, detections)
7 save_image(image_with_boxes, 'result.jpg')
8
9 # Print detection details
10 for det in detections:
11     print(f"Detected {det['label']} with confidence {det['score']:.2f}")
12
```

Code Execution Output (3.29s): ----- stdout -----

Detected fallen person with confidence 0.26

Detected walking cane with confidence 0.31

----- stderr -----

**Step 7:** Based on the analysis so far, I can finalize my findings about the unusual situation in the image. The tools have confirmed what appears to be a medical emergency - an elderly person who has fallen on the floor. The detection system identified both the fallen person and their walking cane with reasonable confidence scores. This is indeed an unusual and concerning situation that requires immediate attention.

**Finalizing Plan:** The analysis has successfully identified the unusual situation in the image: an elderly person who has fallen on the floor (detected with 26% confidence) with their walking cane nearby (detected with 31% confidence). This represents a potential medical emergency situation. The unusual aspects include:

1. The person's position on the floor
2. The presence of a fallen walking cane
3. The person's posture suggesting they are in distress

This kind of situation highlights the importance of fall detection and emergency response systems for elderly individuals living alone.

**Final Plan:** The plan is to detect unusual elements in an image, specifically identifying a fallen elderly person and their walking cane. The plan uses the owl\_v2\_image tool to detect and visualize these elements, confirming the unusual situation of a potential medical emergency.

Plan

#### Instructions

Load the target image using load\_image()

Use owl\_v2\_image tool to detect 'fallen person, walking cane' in the image

Visualize the detections by overlaying bounding boxes on the image

Save the resulting image with detections

Print detection confidence scores for verification

#### Plan Code

```
===== Code =====
1
2 # Load the image
3 image = load_image('tmpibfda2cl.jpg')
4
5 # Detect fallen person and related objects
6 detections = owl_v2_image("fallen person, walking cane", image)
7
8 # Visualize the detections
9 image_with_boxes = overlay_bounding_boxes(image, detections)
10 save_image(image_with_boxes, 'result.jpg')
11
12 # Print detection details
13 for det in detections:
14     print(f"Detected {det['label']} with confidence {det['score']:.2f}")
15
```

Code:

===== Code =====

```
1
2 from typing import Dict, List, Tuple
3 import numpy as np
4 from vision_agent.tools import load_image, owl_v2_image, overlay_bounding_boxes, save_image
5
6 def detect_fallen_person(image_path: str, output_path: str = 'result.jpg') -> Tuple[List[Dict], np.ndarray]:
7     """
8     Detects unusual situations in an image, specifically focusing on fallen persons and walking canes.
9
10    Parameters:
11        image_path (str): Path to the input image file
12        output_path (str, optional): Path where the annotated image will be saved.
13                                   Defaults to 'result.jpg'
14
15    Returns:
16        Tuple[List[Dict], np.ndarray]: A tuple containing:
17        - List of detections, where each detection is a dictionary with 'label',
18          'score', and 'bbox' keys
19        - Annotated image as a numpy array with bounding boxes overlaid
20
21    Example:
22        >>> detections, annotated_image = detect_fallen_person('input.jpg', 'output.jpg')
23        >>> for det in detections:
24            >>> print(f"Detected {det['label']} with confidence {det['score']:.2f}")
25    """
```

```
25    """
26    # Load the image
27    image = load_image(image_path)
28
29    # Detect fallen person and related objects
30    detections = owl_v2_image("fallen person, walking cane", image)
31
32    # Visualize the detections
33    image_with_boxes = overlay_bounding_boxes(image, detections)
34
35    # Save the resulting image
36    save_image(image_with_boxes, output_path)
37
38    # Print detection details
39    for det in detections:
40        print(f"Detected {det['label']} with confidence {det['score']:.2f}")
41
42    return detections, image_with_boxes
43
```

**Test:**

===== Code =====

```
1
2 def test_detect_fallen_person_basic_functionality():
3     """
4     Test case to verify the basic functionality of detect_fallen_person function.
5     This test checks:
6     1. If the function processes the given image successfully
7     2. If the output follows the expected format (tuple with list and numpy array)
8     3. If the detections list contains dictionaries with required keys
9     """
10    # Use the image provided in the user instructions
11    image_path = "tmpibfda2cl.jpg"
12    output_path = "result.jpg"
13
14    # Run the detection function
15    detections, annotated_image = detect_fallen_person(image_path, output_path)
16
17    # Print the detection results
18    print("\nDetection Results:")
19    print(f"Number of detections: {len(detections)}")
20    for det in detections:
21        print(f"Detected {det['label']} with confidence {det['score']:.2f}")
22
23    # Print the shape of the annotated image
24    print("\nAnnotated Image Info:")
25    print(f"Image shape: {annotated_image.shape}")
26    print(f"Image dtype: {annotated_image.dtype}")
27
28    return detections, annotated_image
29
30 # Run the test
31 test_detect_fallen_person_basic_functionality()
32
```

# Final Output

```
Code execution result: ----- stdout -----
Detected fallen person with confidence 0.26
Detected walking cane with confidence 0.31
```

```
Detection Results:
Number of detections: 2
Detected fallen person with confidence 0.26
Detected walking cane with confidence 0.31
```

```
Annotated Image Info:
Image shape: (408, 612, 3)
Image dtype: uint8
```

```
Annotated Image Info:
Image shape: (408, 612, 3)
Image dtype: uint8
```

```
----- stderr -----
```

```
----- Intermediate output-----
```

```
None
```

```
----- Intermediate output-----
```

```
None
```

```
----- Intermediate output-----
```

```
<PIL.Image.Image image mode=RGB size=612x408>
```

```
----- Final output -----
```

```
([{'label': 'fallen person',  
  'bbox': [0.349, 0.288, 0.849, 0.896],  
  'score': 0.26},
```

```
{ 'label': 'walking cane',  
  'bbox': [0.163, 0.739, 0.405, 0.947],  
  'score': 0.31}],
```

```
array([[[194, 177, 161],  
        [194, 177, 161],  
        [194, 177, 161],
```

```
....,
```

```
[168, 142, 125],
```

```
[168, 142, 125],
```

```
[167, 141, 124]]], shape=(408, 612, 3), dtype=uint8))
```

