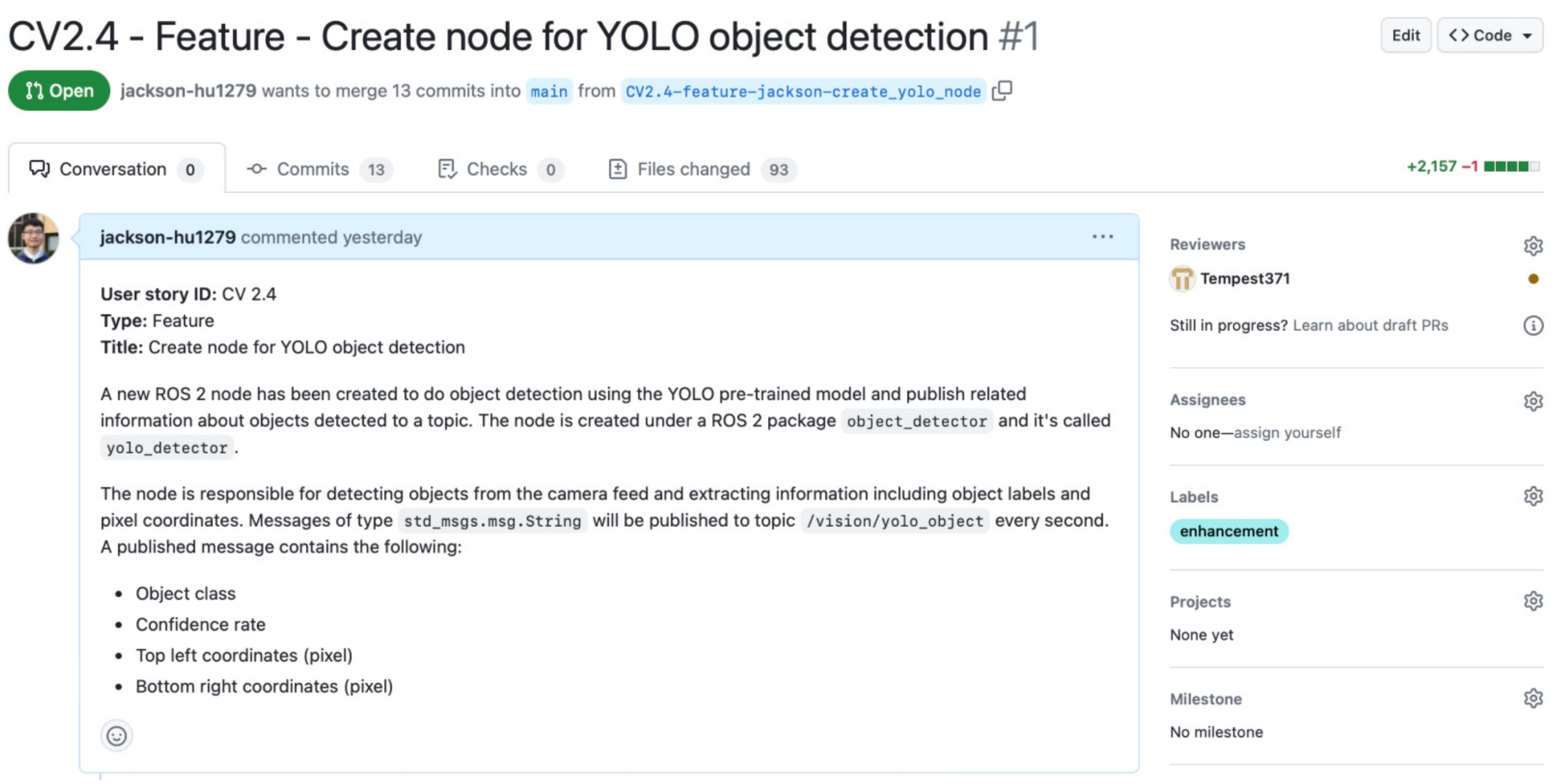
Code Review Template

Created by Renwei Hu, last modified on 20 Sep, 2023

Request Date	18 Sep 2023
Initiator	@Renwei Hu
Reviewed Date	19 Sep 2023
Reviewer	George Wang
Result	**Approve/Request Changes/Decline**

Pull Request Details

Repository	https://github.com/COMP90082-2023-SM2/AI-RedBack-Vision
PR Title	CV2.4 - Feature - Create node for YOLO object detection
PR Link	https://github.com/COMP90082-2023-SM2/AI-RedBack-Vision/pull/1
Type	Feature/Bug-fix
Code Conflict	Yes/No



CV2.4 - Feature - Create node for YOLO object detection #1

Open jackson-hu1279 wants to merge 13 commits into `main` from `CV2.4-feature-jackson-create_yolo_node`

jackson-hu1279 commented yesterday

User story ID: CV 2.4
Type: Feature
Title: Create node for YOLO object detection

A new ROS 2 node has been created to do object detection using the YOLO pre-trained model and publish related information about objects detected to a topic. The node is created under a ROS 2 package `object_detector` and it's called `yolo_detector`.

The node is responsible for detecting objects from the camera feed and extracting information including object labels and pixel coordinates. Messages of type `std_msgs.msg.String` will be published to topic `/vision/yolo_object` every second. A published message contains the following:

- Object class
- Confidence rate
- Top left coordinates (pixel)
- Bottom right coordinates (pixel)

Reviewers
Tempest371
Still in progress? Learn about draft PRs

Assignees
No one—assign yourself

Labels
enhancement

Projects
None yet

Milestone
No milestone

Issues or Advice

The following is just an example:

Naming Conventions: It's important to adhere to our project's naming conventions consistently. In this code, I noticed that variable names like *`var1`* and *`tmp`* don't provide meaningful context. Let's consider using more descriptive names that convey the purpose of these variables. For instance, *`var1`* could be renamed to *`userCount`* for clarity. Clear and consistent naming improves code readability and maintainability.

Remove Unused Imports: I also noticed some unused imports in the code file. It's a good practice to remove them to keep the codebase clean. For example, we can safely remove the *`import datetime`* statement if it's not being used in this module. Cleaning up unnecessary imports reduces clutter and can improve compilation times.

Code Review Checklist

Category	Item	Check
1. Feature Requirement	1.1 Expected feature or functionality has been completed.	✅ / ❌
	1.2 No poorly implemented functions.	✅ / ❌
	1.3 Exceptions are appropriately handled.	✅ / ❌
2. Readability	2.1 The code is organised in its structure.	✅ / ❌
	2.2 The purpose of each block/function is clear and easy to understand.	✅ / ❌
	2.3 Appropriate comments are present in the code.	✅ / ❌
3. Maintainability	3.1 The code is easy to test and debug.	✅ / ❌
	3.2 Compatability and extendability are considered.	✅ / ❌
	3.3 Features or functionalities don't rely on deprecated libraries or functions.	✅ / ❌
4. Security Vulnerabilities	4.1 The code doesn't use tools or libraries with known security issues.	✅ / ❌
	4.2 User data is appropriately processed and stored if involved.	✅ / ❌
	4.3 No obvious vulnerabilities or known security problems.	✅ / ❌
5. Speed and performance	5.1 The code is efficient and applicable.	✅ / ❌
	5.2 No chunks of duplicated code.	✅ / ❌
	5.3 The feature or functionality will not impact the overall performance.	✅ / ❌
6. Code Documentation	6.1 Accurate and concise git commit descriptions.	✅ / ❌
	6.2 A brief explanation or description for the pull request is present.	✅ / ❌
	6.3 Documentation or user manual is prepared if necessary.	✅ / ❌
7. Coding Standards	7.1 Language naming conventions (e.g. variables, functions, comments) are followed.	✅ / ❌
	7.2 Follows the appropriate code layout.	✅ / ❌
	7.3 Code is checked with formatting tools for consistency.	✅ / ❌