Visual Odometry Pipeline

Robert Jomar Malate, Aron Tse Rong Choo, Rajiv Bharadwaj, Kristof Floch

I. Introduction

VISUAL odometry (VO) is the process of incrementally estimating the position and orientation of a vehicle by examining the changes that motion induces on the subsequent images of the onboard camera of the vehicle [1]. Using VO is beneficial as it provides accurate pose estimation, while it is not affected by external effects such as wheel slippage.

In this project, we have developed a monocular 2D→3D visual odometry pipeline for ground vehicles. We have implemented our pipeline using three open-source datasets:

- KITTI[1],
- Malaga [2],
- Parking [3].

The remainder of this report is structured as follows. In Sec. II gives an overview of the technical and theoretical background of monocular VO. Then Sec. III describes the overall structure of our pipeline and describes the individual components. Sec. IV presents the results of the experimental evaluation. Finally, Sec. V contains the discussion of the results, followed finally by the conclusions in Sec. VI. For the code that was developed, refer to our GitHub (**https://github.com/RobJMal/vamr_2024_project**).

II. Background

Given an image sequence, indirect VO begins with feature detection and matching which enables us to establish correspondences across frames. These correspondences are then used to estimate the relative camera pose between successive frames, thereby providing us with motion estimation. In contrast, direct VO uses the pixel intensities of successive frames instead of feature points to estimate camera motion.

Next, given the estimated camera poses, the 3D position of matched keypoints (henceforth referred to as landmarks) can be triangulated, which can then be used to estimate subsequent camera poses through 2D→3D feature correspondences. Lastly, local optimization steps such as bundle adjustment and pose-graph optimization are often undertaken to refine the estimated poses and minimize reprojection error.

Since monocular VO lacks depth information, it suffers from scale ambiguity, meaning that the camera motion and position can only be estimated up to an unknown scale factor. However, this scale can be computed by integrating IMUs as is done in Visual-Inertial Odometry (VIO) or by utilizing known object sizes such as ArUco markers.

All authors are pursuing an MSc. in Robotics Systems and Control at ETH Zürich (e-mails: rmalate@ethz.ch, archoo@ethz.ch, rbharadwaj@ethz.ch, kfloch@ethz.ch)

[1]https://rpg.ifi.uzh.ch/docs/teaching/2024/kitti05.zip

[2]https://rpg.ifi.uzh.ch/docs/teaching/2024/malaga-urban-dataset-extract-07.zip

[3]https://rpg.ifi.uzh.ch/docs/teaching/2024/parking.zip

III. Method

Our method follows the suggested Markovian structure, with the following two main components:

1) Initialization: a bootstrapping phase that searches for informative landmarks and obtains the 2D→3D correspondences via triangulation;
2) Continuous operation, where we process each incoming frame and estimate the position and orientation of the camera. This part can be further divided into 3 main parts
    a) The tracking of existing 2D keypoints between subsequent frames and then associating the tracked 2D keypoints with the 3D landmarks;
    b) The estimation of the camera position and orientation using the 3D landmarks;
    c) The identification of new, additional keypoints and triangulating the corresponding 3D landmarks.

A state object is passed through and updated in each stage of the VO pipeline. It describes the state of the current frame and consists of five items:

1) P holds the 2D keypoint locations in the current frame;
2) X holds the corresponding 3D landmark locations of the 2D keypoints in the current frame;
3) C holds the set of candidate keypoints in the current frame;
4) F holds a set containing the first observations of the track of each candidate keypoint;
5) Tau holds the camera poses at the first observation of the candidate keypoint.

A. Initialization

The first component of the visual odometry pipeline was the initialization of 2D keypoints and 3D landmarks from two selected frames at the start of the image sequence. After generating inlier keypoints from feature extraction and matching of the two frames, two-view geometry was used to estimate the relative camera pose, thereby enabling the triangulation of landmarks.

Our implementation largely followed the recommendations provided. Two frames near the start of each dataset were manually selected for bootstrapping, such that the baseline between the two initialization frames is large enough to reduce depth uncertainty (ideally at least 10% of the distance to the closest features). Frame 0, along with frame 2, 3 and 4 for the Parking, Malaga, and KITTI datasets respectively, were chosen as the two frames for bootstrapping. Next, the Shi-Tomasi corner detector was used to extract features from each of the two frames, followed by brute-force matching using SIFT descriptors to compute feature correspondences.

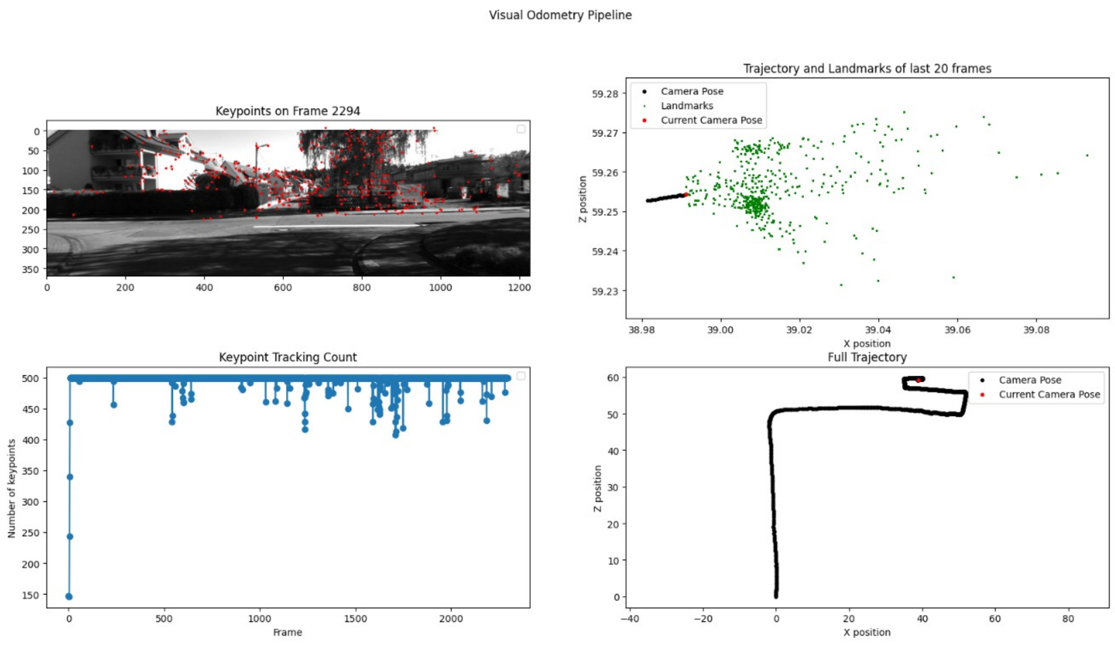


Fig. 1: Screenshot of our VO pipeline

A corner detector was chosen for feature extraction instead of a blob detector like SIFT because corners have better localization accuracy, making them better suited for VO. Then, 5-point *random sample consensus* (RANSAC) with the provided camera intrinsics was used to remove outlier correspondences and estimate the essential matrix, from which the relative pose (R, t) was recovered. The camera pose of the frame 0 was taken to be the origin of the world frame. Lastly, the provided calibration matrix, K, coupled with the computed relative pose and inlier correspondences were then used to triangulate a point cloud of 3D landmarks (X) from the inlier 2D keypoints (P). A small number of landmarks (2-3) were triangulated to be behind the camera, which were then removed from the state object along with their corresponding keypoints. The resultant state object with P and X populated was then passed to the continuous operation phase of the pipeline.

B. *Keypoint tracking*

The keypoint tracking module uses the 2D keypoints of the state object (P), the 3D landmarks (X) and the new incoming image to obtain the new keypoints and landmarks. By utilizing the Kanade-Lucas-Tomasi (KLT) tracker of the OpenCV library, the 2D keypoints from the previous to the new image can be tracked with subpixel accuracy. As we have observed a significant number of outliers during the keypoint tracking, we have utilized RANSAC to filter out the wrong matches. Furthermore, we also filtered out keypoints that moved significantly more than the mean distance travelled. For this, we utilized 3 standard deviations from the mean distance travelled by all keypoints as a threshold.

Finally, we have updated the state with the new keypoints and the 3D landmarks (X) by removing the ones corresponding to lost and filtered keypoints. The main parameters of the module are the parameters required by the KLT algorithm, i.e., window size, maximum pyramid level and termination criteria. Furthermore, the module also enforces a lower limit on the number of keypoints to track. If the number of keypoints falls under this limit the pipeline terminates. This is to ensure that subsequent modules, such as the pose estimation have sufficient data.

C. *Pose Estimation*

From the landmarks and their corresponding keypoints in the image, we then determine the pose of the camera with respect to the world frame. To do so, we created our pose estimation module. The inputs to this are the following:

- The 2D keypoints (P).
- The 3D landmarks (X).
- The intrinsic matrix K.
- The initial pose. This is used to provide an initial estimate that the solver can build on.
- The ID of the frame with which we are working. This is primarily used for plotting purposes.

From these inputs, we utilize them to get an estimate of the pose. We utilize the *perspective-n-points* (PNP) RANSAC solver from the OpenCV library to solve the PNP problem using RANSAC ([2], [3]). The outputs that the solver returns are the following:

- The return value (indicating success or failure to find a solution).
- The rotation vector with respect to the camera.
- The translation vector with respect to the camera.
- The indices of the inliers.

To further refine the solution, we then perform a non-linear optimization step using the Levenberg-Marquardt minimization scheme to minimize the reprojection error using the

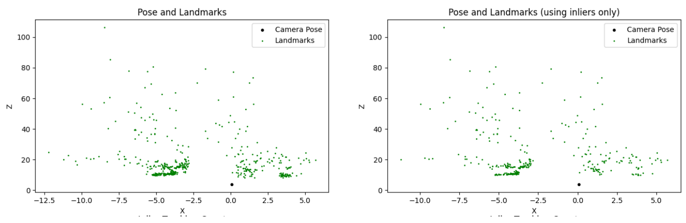


Fig. 2: PNP solving and pose refinement. Plot of left uses the data to solve the initial PNP problem with RANSAC. The plot on the left showcases the PNP refinement and the inlier points that were used.

solvePnPRefineLM function from OpenCV [4]. In addition to the rotation vector and the translation vector outputted from the previous step, we provide the 2D keypoints and 3D landmarks that have been filtered as inliers from the previous RANSAC solver. A comparision between the points used for solving PNP and refining the output can be found in Fi

After the solver returns the optimized rotation vector and translation vectors, the rotation vector is converted to a rotation matrix using the *Rodrigues* function from OpenCV. The feature then returns the following output:

- The return value (indicating success or failure to find a solution).
- Rotation matrix (world with respect to camera frame).
- Translation vector (world with respect to camera frame).

D. New Landmark Tracking

To ensure continuous operation of the odometry pipeline, the new landmark tracking component constantly evaluates possible candidate keypoint-landmark pairs that can be added to P and X respectively that can provide high quality information to the pipeline. For a given state, C holds the 2D candidate keypoints corresponding to the latest image frame. F holds the location for first observations of the same keypoint. Tau holds the pose corresponding to the camera frame of the first observation.
We perform this task in the following steps

1) Continuous Tracking of Candidate Keypoints: Using the same methods mentioned in III-B, we filter out candidate keypoints from the previous state that can no longer be tracked.

2) Triangulation and Evaluation of a Landmark: For a given camera frame i, a candidate keypoint $c \in C^i$, we use the current camera pose T^i_{WC}, state variables $f(c) \in F$, $\tau(c) \in Tau^i$ to triangulate a candidate x_C in the world frame (Fig. 3).

Due to the possible accumulation of inaccuracies in each of the variables used for triangulation, the set of all candidate landmarks X^i_C goes through a filtering process to reject landmarks triangulated behind the camera, and also rejects outliers based on the current set of active landmarks X^i. We also calculate the angle $\alpha(c)$ as the angle between the two bearing vectors between the $(f(c), \tau(c))$ and (c, T^i_{WC}). This angle acts as a basis for declaring a landmark to be significant as discussed in III-D4

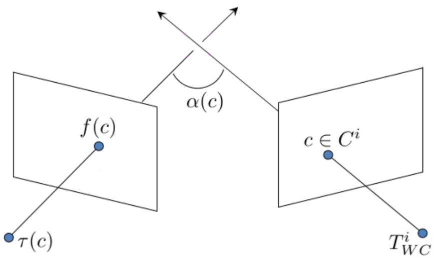


Fig. 3: Landmark Triangulation Method
(Figure obtained from the spec)

3) Finding new candidate keypoints: We apply the same methods as in III-A to identify new keypoints using the previous and current image frames. To improve the quality of new candidate keypoints, we ensure that they are sufficiently far enough from existing keypoints in P. For a new candidate keypoint, its location is saved to both C_i and F_i, and the current camera pose T_i is saved to Tau_i.

4) Putting it all together: A keypoint-landmark pair is considered to be eligible to put into P and X respectively when the triangulation angle exceeds a preset threshold $\alpha(c) > \alpha_t$. This ensures that the keypoint has been tracked for a sufficient time period (from III-D1) and there's a sufficient baseline distance between the two camera poses to reduce the depth estimation uncertainty.

IV. Experimentation and Results

To help measure the quality of our pipeline, we performed experiments on the data sets provided. The screencasts of the experiments for the three datasets are available at **https://www.youtube.com/playlist?list=PLKhwq5S3PpRlKr-pcsuEx62bNuZnPCTZc**.

In our experimentation, except for the frames that were selected for boostrapping, we use the same set of hyperparameters for all datasets. The frames selected are discussed in their respective sections.

A. Parking

During development, this baseline often served as a quality check, since it involved only motion in a single direction (to the right), and ground-truth data are provided. The bootstrapped frames are the 1st (index and image name 0) and 3rd frame (index and image name 2).

We wanted to highlight some notable observations when we were experimenting with this dataset:

- **Scale Ambiguity**: The scale ambiguity phenomenon can be observed here. As was discussed in class, this occurs when we use monocular vision since we do not have a reference that constrains the scaling factor. Our VO pipeline shows that the camera is moving in the same direction as the ground truth, but it is moving to the right at a much greater magnitude.
- **Increasing Depth Uncertainty**: When the camera passes through the driveway, the estimated position z increases significantly. This can be attributed to the fact that the

error estimate of the depth increases quadratically with respect to the depth value (as shown in the class). However, when the camera starts to pass through the parked cars, the estimated z-position starts to decrease.

B. KITTI

For this dataset, the bootstrapped frames are the 1st (index and image name 0) and 5th frame (index and image name 4). When experimenting the dataset, we note the following observations:

- **Dropping Keypoints**: Around frame 2340, our pipeline started to drop keypoints. In this scene, the camera stops moving while at the same time, other cars are passing by. The camera loses keypoints when it stops, and losees even more with the cars driving by. We believe that this keypoint drop is caused by two factors: (1) the occlusions from the cars passing by covering the previous keypoints and (2) the camera being stationary and not picking new keypoints.

C. Malaga

For this dataset, the bootstrapped frames are the 1st (index and image name 0) and 4th frame (index and image name 3). When experimenting the dataset, we note the following observations:

- **Loss of Keypoints**: The scale ambiguity phenomenon can be observed here. As was discussed in class, this occurs when we use monocular vision since we do not have a reference that constrains the scaling factor. Our VO pipeline shows that the camera is moving in the same direction as the ground truth, but it is moving to the right at a much greater magnitude.
  Because of the limited computational resources, we needed to reduce the number of keypoints that we are tracking.

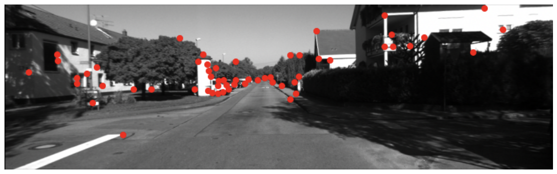
V. Discussion

During development, there were several notable challenges that we encountered.

A. Quality of Keypoint and Landmark Selection

*1) **Identifying Useful Keypoints**:* During development, we found that using features detected by the Shi-Tomasi detector performed significantly better than the Harris detector or using SIFT features. These provided points that were more robust and enabled more accurate pose estimation over time. See figure 4

*2) **Limiting the Number of Keypoints and Landmarks**:* We aimed to store a low number of keypoints and landmarks in both the main set P, X and candidate sets C, F, Tau to ensure that our pipeline did not slow down as we progressed through a dataset. We set a limit of 500 keypoints and landmarks that can be available in the sets P and X respectively, and an incremental limit of 100 within P, X from C that can be added per iteration, and 500 that can be added to C itself. These thresholds were empirically determined to ensure the balance


(a) Harris Keypoints

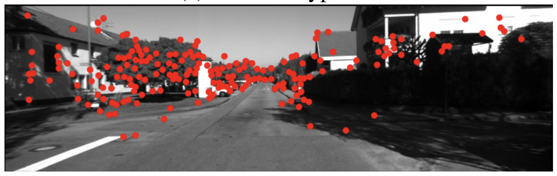

(b) Shi-Tomasi Keypoints

Fig. 4: Keypoint Detection Comparison during Bootstrapping

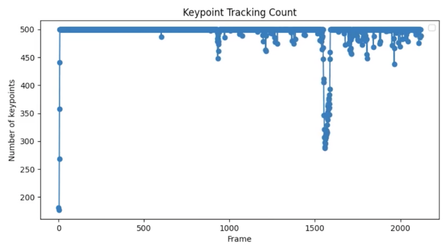


Fig. 5: Number of Keypoints Tracked with the Malaga Dataset

between having a sufficient number of candidates available to track, while not bloating the main state variables, see Figure 5.

*3) **Improving Keypoint and Landmark Diversity**:* By deciding to reduce the number of keypoints and landmarks, we had to figure out a way to improve the diversity of the keypoint and landmark locations that were tracked in C, and subsequently added to the sets P, X. To achieve this, we added a feature to track a new candidate keypoint c if it is at least at a distance of 8 pixels away from a currently tracked keypoint in P. This number was empirically determined.

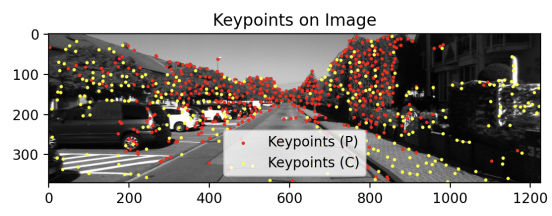
Similarly, if we pick a set of C_e, X_e^C eligible candidate keypoints and their respective landmarks to add to P, X respectively, we only add the best n candidates based on a ranking metric that depends on both the triangulation angle $\alpha(c), c \in C_e$ and the distance of the landmark $x \in X_e^C$ from the currently tracked set of landmarks X.

Figure 6 shows the search diversity with the KITTI dataset

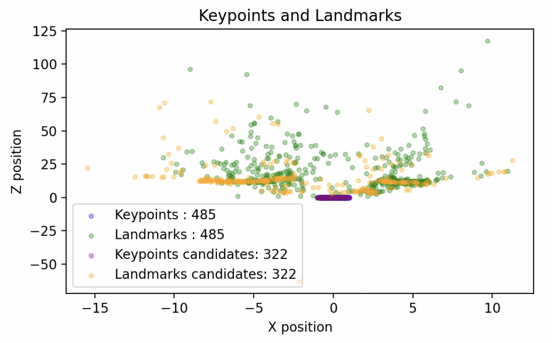
B. Filtering Out Outliers

Since pose estimation was both an output of a given step i and an input to the next step $i + 1$, the pipeline was very sensitive to outliers present in the data passed around in the state variable, often failing due to an accumulation of such errors.

*1) **Using RANSAC for Inlier Filtering**:* As was mentioned in the lecture and what we encountered during development, outliers have a significant effect in the estimates that our pipeline produces. To filter out these, we use RANSAC to remove and reject outliers when performing Keypoint Tracking, Pose Estimation, and Candidate Keypoint Tracking.



(a) Evaluated Keypoints vs Existing Keypoints



(b) Evaluated Candidates vs Existing Candidates

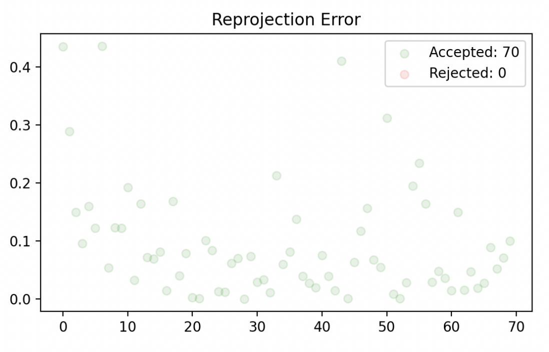
Fig. 6: Diversity of Candidate vs Tracked Keypoints and Landmarks in the KITTI dataset

2) ***Using Reprojection Error to Optimize Triangulation Accuracy:*** Based on the lecture, since the triangulation of 2D points to 3D Landmarks involves solving a linear least-squares problem, we attempted to perform non-linear optimization using the `scipy.optimize.least_squares` method using Levenberg-Marquardt (LM) algorithm. However, we noticed the bulk of the datapoints already had a low reprojection error < 1.0 pixels, with large outliers appearing in certain iterations, see Figure 7. In favor of speeding up each triangulation iteration, and considering that candidate keypoints are not contributing to the main pipeline, we decided to implement a simpler rejection mask that rejects any triangulated landmarks that have a reprojection error ≥ 2.0 pixels.

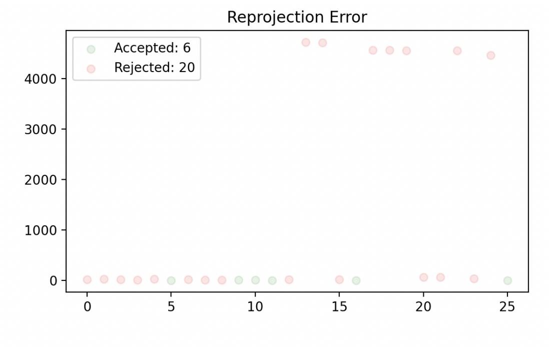
3) ***Removing Outliers based on Z-Score:*** Another outlier rejection method we implemented was to remove the landmarks that were located 3 standard deviations away from the mean. Since the computation of the Z-score was done for all three dimensions, we took the average across all of them instead. A possible improvement could be to consider a weighted mean by the dimension.

C. Verifying Reference Frames

Once development of individual components was done by each member, we realized that there was a lack of standardization and understanding of the best coordinate frames to evaluate and pass data. Initally, we believed it was an error with the landmark triangulation and keypoint addition. Upon further inspection, we realized that the reference frames were mixed up between the camera frame and the world frame. After correcting this, we were able to observe outputs that align better with the ground truth.



(a) Accepted Landmarks



(b) Rejected Landmarks

Fig. 7: Plot showing the acceptance and rejections of triangulated landmarks based on reprojection error

The error was quite subtle. Looking at the trajectory plot when compared to the ground truth, it initially would appear that the trajectory aligns with the ground truth. However, after multiple frames (around 20), our pipeline provides an estimate that deviates strongly from the ground truth. Once this happens, the performance of the pipeline significantly declines and is unable to fix itself.

VI. Conclusion

In this project, a monocular 2D$\rightarrow$3D VO pipeline has been developed and and implemented. Our pipeline utilized an indirect, feature based method that can be separated into two main part: initialization, where informative features and landmarks are selected for VO; and a continuous operation where the features are tracked across consecutive frames, the camera pose is determined, and new landmarks are added. The developed pipeline has been evaluated using three open source dataset: KITTI, Malaga, and Parking. From the experimental results, we can conclude that our pipeline performed adequately, albeit with some expected challenges due to the inherent limitations of monocular VO. The observed scale ambiguity and increasing depth uncertainty highlight the need for integration with additional sensors, such as IMUs, or the use of known references to improve accuracy and robustness.

VII. Author Contributions

Aron was responsible for the initialization module, while Kristof, Robert, and Rajiv contributed to the continuous VO module, namely keypoint tracking, current pose estimation, and new landmark triangulation respectively. All team members contributed equally to developing visualizations, running experiments, and debugging.

References

[1] D. Scaramuzza, “Lecture notes on vision algorithms for mobile robotics,” University of Zürich, Tech. Rep., 2024.
[2] OpenCV, *OpenCV: solvePnP*, 2023, accessed: 2025-01-06. [Online]. Available: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html
[3] ——, *OpenCV: Camera Calibration and 3D Reconstruction*, 2023, accessed: 2025-01-06. [Online]. Available: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga50620f0e26e02caa2e9adc07b5fbf24e
[4] ——, *Calibrate Camera — cv::calibrateCamera*, 2025, accessed: 2025-01-06. [Online]. Available: \url{https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga650ba4d286a96d992f82c3e6dfa525fa}