International Journal of Computational Intelligence in Control
Vol. 11 No. 1 (June, 2019)

MUK PUBLICATIONS
Open Access Publisher

Video Detection, Tracking and Path Planning for Ground Autonomous Systems

N. Aouf[1], M. Kharbat[2], Chun Liang, Lin[3]

[1,2]Department of Informatics and Sensors, Cranfield University
[3]Department of Electrical and Engineering, Chung Hsing University

Abstract: *This paper presents an approach of motion detection, tracking and path planning for ground autonomous vehicles using global environment visual data. Motion detection and tracking depends on the (Kanada-Lucas-Tomasi) KLT algorithm but with selecting features in a more robust manner to suit our application and our environment constraints. The proposed path planning technique is based on translating scene visual data into a graph network and optimizing it by enhancing the distribution of the nodes and edges. An optimal feasible path is then obtained by finding the best trade-off between safety and route shortness. Detailed experiments are conducted to validate the proposed methodology.*

Keywords: *Video Detection, Tracking, Path Planning, Autonomous Systems*

1. INTRODUCTION

The problem of generating a safe path for robots has drawn considerable attention in the past few years [1, 2]. The majority of the proposed algorithms assume that the environment is completely or partially known before the robot begins its traverse. Most algorithms search an environment using the distance transform [3] or heuristics [4, 5] to find the lowest cost path from the robot's start point to the target point. Cost can be defined to be distance travelled, energy spent, time in danger…etc. However, the robot may be placed to navigate in an unknown environment but equipped with sensors placed on board or remotely. Approaches in this case can be grouped into two main categories: local and global. The local approaches use on-board sensors to sense the terrain locally, and while the vehicle moves, it acquires sense of new parts of the environment, either to discover more about its current location or to discover more about the map. This way, the vehicle keeps on investigating its surrounding environment during the entire journey, and uses it as partial information to obtain a safe path. Most local approaches result in decisions which are not ideal and present many path planning risks especially in a dynamic environment. On the other hand, the global approaches use external sensors to build full information of the whole environment and create a detailed description of it so it can identify all its elements as the first step, and then search for feasible collision-free paths. Some global-like path planning techniques have been investigated in the past such as Propagating Interface Technique [6] and approximate cell decomposition [7]. The main challenge of the global approach appears in its initial step building the map. This process is often of high expense and computationally heavy which makes it unattractive to use in real-time applications and infeasible in dynamic environments. In this paper, we tackle this problem and propose an integrated, easy-to-implement and efficient methodology to interpret the environment and design a safe and short path for the robot to roam through, while keeping that robot detected and tracked. In a coordinated Air/Ground autonomous mission where aerial vehicles have a large view of the land to be investigated by the autonomous ground agents, guided ground operations could efficiently be planned from air. We introduce an integrated methodology for motion detection, tracking and path planning to guide deployed ground vehicles using aerial visual data.

The paper starts by presenting, in section 2, our motion detection and feature tracking methodology. In this methodology, we first extract motion information from real time image sequence. Then, we propose a detection/tracking technique that builds up on the original version of the texture correlation-based KLT feature tracker. Our technique is based on selecting the best feature to be detected, and then supplying it to the KLT algorithm after checking object availability in the first two frames using Hu invariant moments. In section 3, a path planner, using global information of the ground configuration, is introduced to allow the ground vehicle to move safely to any desired location. We have access to an adequate amount of information that covers the environment so we can identify its elements pre vehicle departure and classify the geometric areas whether they are free or occupied by objects (obstacles).

Using an occupancy grid-based representation and a connected graph interpretation of the world, we define an optimal route to the ground agent in order to approach its aimed destination while avoiding obstacles. Our path

planning technique is, in fact, based on translating the visual environment data to a directed graph (network), by identifying its nodes and edges using the free-space regions available, to enable the use of Dijkstra's algorithm later to find the safest path. This path is then optimised, rising up a compromise between safety and route distance. A weighting factor, based on the vehicle geometry, is proposed to obtain the best trade-off between safety and shortness.

The last section presents our methodology results based on experiments conducted on mobile robots in our labs. Testing and trials are based on a set of camera fixed on the ceiling of the lab to emulate the large view obtained from the aerial vehicles.

2. MOTION DETECTION AND TRACKING

Our detection and tracking technique consists of four main stages: (1) motion detection (2) Hu moments calculation (3) histogram-based feature selection (4) KLT tracking. Once motion is detected, an object is identified and its Hu invariant moments are calculated. Another frame is then taken and the moments are calculated again and checked against the first values, if no significant changes are found between both measures and the difference does not exceed a threshold value, it flags that we are having a moving object detected and not noises. At that point, feature selection starts to find appropriate features of the detected object and passes them to the part of the KLT algorithm in which it tracks the object's detected features. The four stages of the detection and tracking technique are shown in Fig. 1 and illustrated hereunder.

2.1. Motion Detection

Assuming that the ground vehicle of interest is the only moving part in our ground environment, the motion detection works as follows:

- While continuous image grabbing, two time-spaced snapshots are taken (Fig. 2(a) and Fig. 2(b))
- The last frame is subtracted from the previous one.
- The result image is analysed for pixels that would indicate a vehicle movement. If there are no pixels, then the two frames must be identical, thus no vehicle is in the scene view and the loop restarts taking the next two frames.

2.2. Hu Moments Calculation

The results of the step presented above could lead to detect noises instead of the vehicle entering the scene. For this reason we use some appearance-based measurements, such as Hu invariant moments, to assure that we are following the same object from a frame to another and thus detecting the vehicle of interest and not noises.

Image moments are statistics characterizing geometrical image information such as image centre of gravity and the distribution of pixels about this centre of gravity called respectively first order moment (the mean) and second order moment (the variance). In this way, if we calculate the mean and the variance of the object in the image, we also find the location, orientation and the aspect ratio of the object in the image. Using these moments, Hu derived a set of invariant moments to translation, rotation, and scale changes [8]. Hu moments use, basically, the second and third normalized central moments of an image to form the seven equations shown in Fig. 3. After the position of the detected object candidate is obtained, a region of interest on the image is set around the object in question to calculate its invariant moments. Remark that in case we do not have a priori information about our vehicle, the invariant moments are calculated for the region of interest in the first difference image frame and moments in the second difference image frame are calculated against them. However, if we could get that a priori detail of the vehicle, it would add more robustness to the detection technique we are proposing.

2.3. Feature Selection

Obviously, no feature based tracking vision system can work unless good features can be found for tracking from a frame to another. Our main challenge is to have robust features

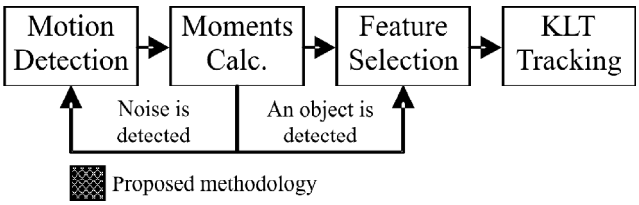

Figure 1: Motion Detection and Tracking Methodology

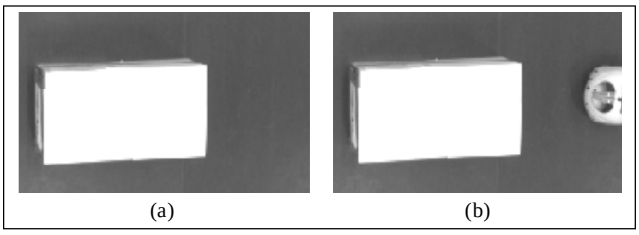

Figure 2: (a) An Obstacle in an Indoor Environment (b) New Element Enters the Environment

$$h1 = \eta 20 + \eta 02$$
$$h2 = (\eta 20 - \eta 02)^2 + 4\eta 11^2$$
$$h3 = (\eta 30 - 3\eta 12)^2 + (3\eta 21 - \eta 03)^2$$
$$h4 = (\eta 30 + \eta 12)^2 + (\eta 21 + \eta 03)^2$$
$$h5 = (\eta 30 - 3\eta 12)(\eta 30 + \eta 12)[(\eta 30 + \eta 12)^2 - 3(\eta 21 + \eta 03)^2] + (3\eta 21 - \eta 03)(\eta 21 + \eta 03)[3(\eta 30 + \eta 12)^2 - (\eta 21 + \eta 03)^2]$$
$$h6 = (\eta 20 - \eta 02)[(\eta 30 + \eta 12)^2 - (\eta 21 + \eta 03)^2] + 4\eta 11(\eta 30 + \eta 12)(\eta 21 + \eta 03)$$
$$h7 = (3\eta 21 - \eta 03)(\eta 21 + \eta 03)[3(\eta 30 + \eta 12)^2 - (\eta 21 + \eta 03)^2] - (\eta 30 - 3\eta 12)(\eta 21 + \eta 03)[3(\eta 30 + \eta 12)^2 - (\eta 21 + \eta 03)^2]$$

Figure 3: Hu Invariant Moments

consistently, and as our ground vehicles has no specific geometrical features we can rely on, the KLT auto feature selection could easily fail to detect vehicle features and/or lose tracking the vehicle in a short time. Thus, we propose an alternative method, simple yet effective, to detect and select one object feature to be tracked.

After detecting motion, an image is captured from the last frame for the vehicle and a feature is extracted from it as the following procedure:

- A threshold level technique is applied on the result image of the vehicle to convert it to a binary image. Each pixel is checked and set to one if it exceeds a specific intensity or zero if otherwise.
- Fig. 4 (b)).
- A histogram based technique is applied on the image where: each row and column is scanned for its number of pixels that have a value of 1; this number is considered as a histogram value. The feature located at the coordinates at which we have the highest histogram values corresponds to our detected vehicle feature.
- Fig. 5 shows the pseudo code for this technique.
- The detected feature should be the visual vehicle centre of gravity. This feature is input to the KLT tracker as the main and only feature to track.

2.4. KLT Tracking

The KLT algorithm is a feature-based technique which extracts local regions of interest (features) from the images and identifies the corresponding features in each image of the sequence [9]. It is based on the early work of Lucas and Kanade [10], was developed by Tomasi and Kanade [11] and was explained by Shi and Tomasi [9]. The tracking process in the original KLT algorithm can be divided into two major subtasks: feature extraction and feature tracking. Features are extracted only in the first frame and then they are searched for in the subsequent frames. If a feature is lost, the user can optionally ask for finding another one to keep the number of features constant. The main principle of KLT tracking is to align a template image $T(\mathbf{x})$ (the first frame) to an input image $I(\mathbf{x})$ (a subsequent frame). $\mathbf{x}$ is a column vector containing image coordinates $[x, y]^T$. The $I(\mathbf{x})$ could be also a small sub window within an image. The KLT defines a measure of dissimilarity that quantifies the change of appearance of the feature between the first and the current image frame. A set of translational warps $\mathbf{W}(\mathbf{x}; \mathbf{p})$, where $\mathbf{p} = [p_1, p_2]$ is a vector of translation parameters is defined as:

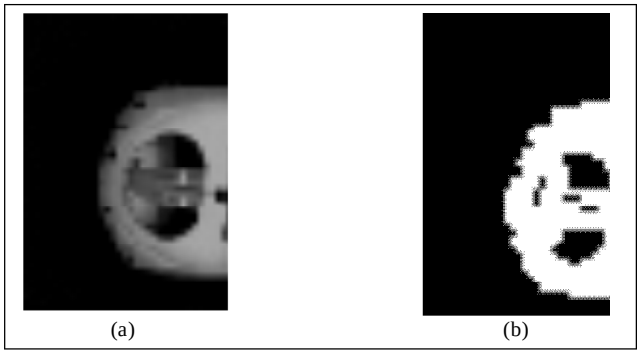

Figure 4: Image Difference and Binary Image

```
INITIALIZATION
HIST_VALUES=0;
FOR ALL ROWS
   FOR ALL PIXELS OF ROW
      HIST_VALUE[ROW]=
      PIXEL_VALUE+HIST_VALUE[ROW]
   END FOR
END FOR
```

Figure 5: Pseudo Code for Applying Histogram Technique on Rows

$$\mathbf{W}(\mathbf{x}, \mathbf{p}) = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix} \quad (1)$$

The best alignment minimises image dissimilarity:

$$\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - T(\mathbf{x})]^2 \quad (2)$$

The stability of the tracking process in the original KLT algorithm is mainly influenced by the selected features of the template $T(\mathbf{x})$. As derived and explained in [9], the criterion of a good feature is a textured path with high intensity variation in box x and y directions, such as a corner.

Denote the intensity function by $I(x, y)$ and consider the local intensity variation matrix

$$Z = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad (3)$$

A patch defined by a window (a square matrix of pixels) is accepted as a candidate feature if in the centre of the window both eigenvalues of Z : λ_1 and λ_2, exceed a predefined threshold λ (texturedness):

$$\min(\lambda_1, \lambda_2) > \lambda \quad (4)$$

2.5. Camera Calibration

To obtain vehicle real world coordinates, camera calibration is done. Assuming Camera projection is modelled with a pinhole camera model, the objective is to determine the internal camera optical characteristics (intrinsic parameters) and position and orientation of the camera frame relative to a chosen world coordinate system (extrinsic parameters).

The camera calibration is conducted using Tsai algorithm [12]. Tsai's camera model is based on the pin-hole model of 3D-2D perspective projection with 1st order radial lens distortion. The model has 11 parameters: five internal (also called intrinsic or interior) parameters, and six external (also called extrinsic or exterior) parameters.

In addition to the 11 variable camera parameters Tsai's model has six fixed intrinsic camera constants. The Internal, external, and fixed intrinsic constants are shown in Table 1, Table 2 and Table 3, respectively while our test camera values are available in the implementation and results section.

Table 1
Internal Parameters of Tsai's Camera Model

Parameter	*Description*
f	effective focal length of the pin-hole camera,
kappa1	1st order radial lens distortion coefficient
Cx, Cy	Co-ordinates of centre of radial lens distortion
sx	Scale factor to account for any uncertainty due to imperfections in hardware timing for scanning and digitisation scanline.

Table 2
External Parameters of Tsai's Camera Model

Parameter	*Description*
Rx, Ry, Rz	Rotation angles for the transform between the world and camera coordinate frames
Tx, Ty, Tz	Translational components for the transform between the world and camera coordinate frames.

Table 3
Intrinsic Constants of Tsai's Camera Model

Parameter	*Description*
Ncx	Number of sensor elements in camera's x direction (in sels)
Nfx	Number of pixels in frame grabber's x direction (in pixels)
Dx	X dimension of camera's sensor element (in mm/ sel)
Dy	Y dimension of camera's sensor element (in mm/ sel)
Dpx	Effective X dimension of pixel in frame grabber (in mm/pixel)
Dpy	Effective Y dimension of pixel in frame grabber (in mm/pixel)

The camera calibration procedure, given the above intrinsic constants, will optimise the intrinsic and extrinsic camera parameters building the camera mathematical model. This optimization requires 3D world coordinates of feature points (x, y, z) (in mm) and their corresponding coordinates in the image plane (x_f, y_f) (in pixels). The selected features used in this paper are the squares corners of a chess pattern deployed on the scene of our experimental setup.

3. PATH PLANNING

The main objective our path planning module is to get the optimal vehicle path where a best trade-off between safety and shortness is taken.

This module consists of two main parts: (1) safest path (2) optimised path.

3.1. Safest Path

In this part and inspired by cell decomposition method, the path planning technique that is proposed splits the environment into a set of grid cells representing the areas of vacant spaces distinguished from the obstacles (occupied areas).

Using this discrete environment mapping, we obtain a network representation of the ground environment, which permits to utilize the graph theory's path algorithms to achieve a safe path planning solution, as will be explained hereunder.

The shortest path problem is one of the most common problems in graph networks applications where we look after a path P, which goes through a collection of graph edges E and nodes N, to be the shortest. The total cost of P is the sum of all individual costs of those edges where the cost, c_{un}, in our case is calculated relevant to the distance from node u to node n. The shortest path problem is a special case of the min-cost flow problem where it examines transportation costs but not capacities: All edges' capacities are equal to 1, then the objective is to find the optimal sequence of edges such that:

$$\min \sum_{e \in E} C_e \tag{5}$$

Where E is the edges set, e is an individual cost, and C_e is the cost corresponding to edge e.

In the context of path planning, the environment is usually analyzed in terms of cost as mentioned earlier. Obstacles in the viewed scene can be associated with an infinite cost.

Obtaining the safest path is done by the following procedure:

- Preparing the environment snapshot :

  After detecting the vehicle of interest, a snapshot of the ground environment is saved in memory and is subject to the following four image processing techniques:

  (a) Edge detection
  Extracts the contours (detects edges) in grey-level values (Fig. 9 (a)).

  (b) Image threshold
  Convert the greyscale image into a binary one as has been explained in the feature selection method, where each pixel is checked and set to one if it exceeds a specific intensity or zero if otherwise (Fig. 9 (b))

  (c) Image filtering
  Filter the noise out of the image depending on their small size relative to the possible sizes of obstacles.

(d) Bounding the obstacle candidates by rectangles
This will simplify the process of detecting regions as will be explained below (Fig. 9 (c)).

- Detecting Regions

  After preparing the environment picture, we start scanning image pixels to identify free regions from obstacles. To simplify identifying these regions, the four corners of each obstacle candidate are found and then treated as the boundaries of our regions along with the corners of the whole picture (Fig. 6).

- Network Generation

  At this stage, our environment picture is ready to be translated into a network representation. A two-obstacle example is shown below in Fig. 7 producing a network *G* with a number of nodes *N* initially equals to 7, shown in Fig. 9 The nodes clearly represent the centres of the regions of free spaces and the directed edges represent the allowed subsequent safe motion from region R_i.

- Introducing new nodes

  As mentioned earlier, the main objective of having a network of nodes is to get a feasible path that can go through them. However, a quick look at the concept shows that some suggested paths could be infeasible and could happen to go over obstacles if straight motion between regions is adopted. Fig. 10 shows the case in more details, it is apparent that a path that goes directly between nodes R2 and R3 are not possible. To solve this difficulty, we introduce a new set of intermediate way-nodes, *N′*. The new nodes facilitate the travelling between adjacent regions that pose risks of safety. Fig. 11 shows how *N′* is obtained; horizontal and vertical lines are drawn from the regions in question.

- Creating the adjacency matrix

  Once an environment network is created, we are almost ready to make use of the appropriate graph theory algorithms to locate the paths between the different regions. A final prerequisite is to form the adjacency matrix *A* of the network, which is based on visual distance measurements. Each element A_{un} of this matrix is calculated depending on the distance between the two nodes *u* and *n* if there is a viable path that goes between them, otherwise, the cost of the element A_{un} is set to infinity.

- Dijkstra's Shortest path algorithm

  Once a graph network *G* is modelled, all possible safe regions within the environment are obtained, and all travelling costs corresponding to the graph topology are found, a graph theory algorithm is used. In this paper, we propose to use the Dijkstra algorithm to find the shortest path through the nodes and the safest practical one from the source (initial vehicle position) to the target (where the vehicle has to go). This algorithm appeared in 1959 in [13], and since then, it is probably the most widely used for graph shortest path computation.

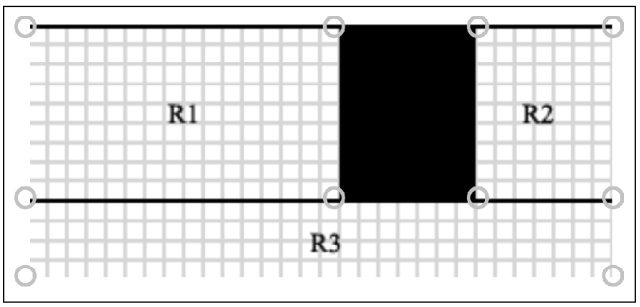

Figure 6: Detecting Regions

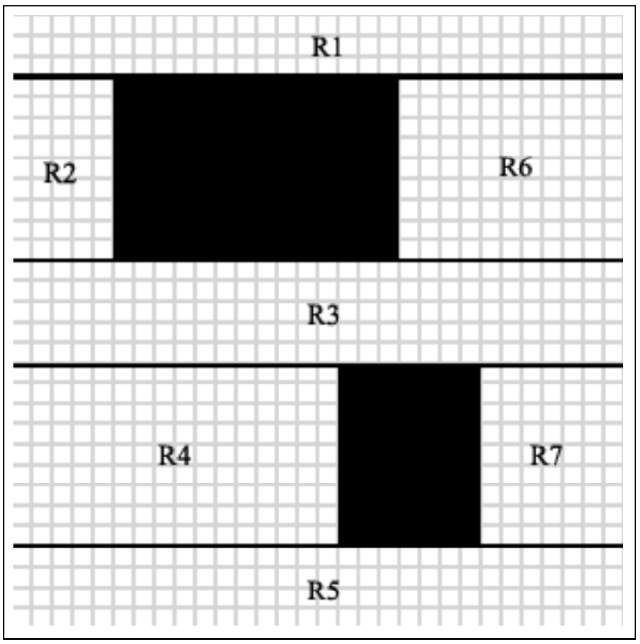

Figure 7: Regions Detected of an Environment of two Obstacles

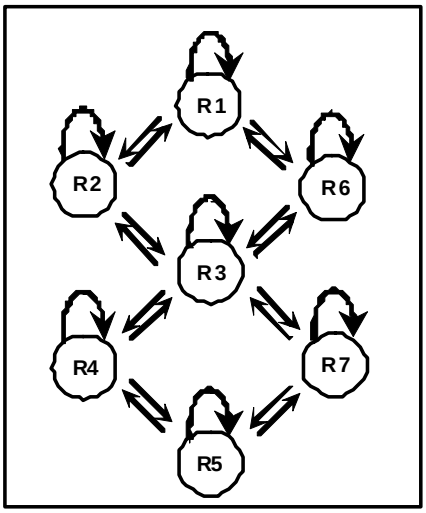

Figure 8: A Network G which Represents the Environment in the Previous Figure

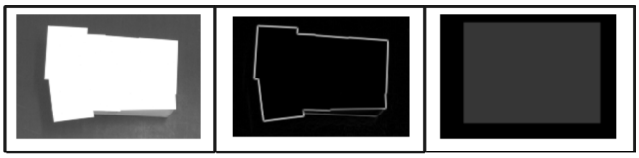

Figure 9: Preparing the Environment Snapshot

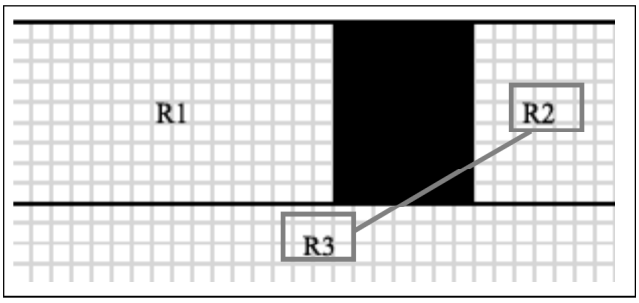

Figure 10: Unsafe Path

Suppose we have a network *G* with *N* nodes and *E* edges. Every edge has two ends represented by the notation (*u*, *n*) and has a value assigned to it, a cost, represents the cost of moving directly from vertex *u* to vertex *n*. The cost of an edge can be thought of as the distance between those two vertices. The cost of a path between two vertices is the sum of costs of the edges in that path. For a given pair of vertices *s* and *t* in *N*, the algorithm finds the path from *s* to *t* with lowest cost.

The basic concept of the algorithm is edge relaxing [14]. It works by traversing all edges going out of a certain node and selects only the edge of the lowest cost to be part of the path. By doing that for all nodes in an iterative way, the shortest path would be obtained. The operation of Dijkstra's algorithm can be summarized by the following steps:

- While the algorithm is running, we have two sets grouping our nodes: *U* and *S*. As a start, we set *U* contains all nodes and *S* is empty. When the shortest path from an initial node to a subsequent node towards the desired location is obtained, the initial node is moved from set *U* to set *S*. The algorithm keeps on running until all way nodes, required to reach the destination node, move to set *S*.
- When Dijkstra's algorithm visits a way node, it relaxes it; that means it examines all edges going out of it to choose the right subsequent way node to use in the shortest path towards the desired destination. A simple case shown in Fig. 12, where we want to find the shortest path to go from a start node to a target node, illustrates this concept. In figure (a), the start node is relaxed, there are two edges going out from it having the costs 10 and 5 and thus the edge of the lowest cost 5, in this case, is chosen. In figure (b) the start node has been moved to S and the second node is chosen to be relaxed, before it is moved to S in the third graph (c) where we reached the target.

Fig. 13 shows the result of the shortest path method for safe regions applied on our set of nodes generated in Fig. 11 Starting from R1, the vehicle is trying to go to R2 through R3. The generated additional intermediate two nodes assist the vehicle to go through a path that is completely safe.

3.2. Path Optimization

The path we have achieved so far, as a result of the Dijkstra's algorithm, proves to be the safest path to manoeuvre through regions since it goes in the centre of area of each free-space region. However, such a path might not be the most selected one in some cases when the safety factor can be flexible. The path planning methodology optimises the path to reduce the cost of it, which would let us achieve an optimum solution in terms of shortness.

The methodology proposed here gives the flexibility to modify the safest path already obtained to go closer to the obstacle so it saves distance to get a shorter path as shown in Fig. 14. However, since the technique here is designed for a point object, a minimum clearance from the obstacle is required and equals to maximum effective radius of the vehicle. The optimization steps can be summarized as follows:

- The first step is to detect if there is an obstacle between the two main nodes (centres of regions), if there is no such obstacle, the path formed is simply a direct line between the two nodes avoiding passing by any added intermediate node.
- If an obstacle exists around, its nearest corner is found and a directive vector V_D is created connecting the starting point with this corner to work as a guide line for our desired optimum path as shown in Fig. 16.
- Safest path vector V_{SP}, obtained from the Dijkstra's algorithm, is identified and labelled on Fig. 15.
- Final vector, dashed in the figure, is calculated by a linear combination with a weighting factor $0 \le w \le 1$ applied to the two vectors: safest path vector V_{SP} and the directive vector V_D, as follows:

$$\bar{R} = w.V_{SP} + (1-w).V_D \qquad (6)$$

The resultant, then, is controlled by the weighting factor; it has a maximum magnitude equals to the safest vector's magnitude when $w = 1$ and a minimum equals to the directive vector's magnitude when $w = 0$.

The optimised path should guarantee that the vehicle does not hit an obstacle while moving on it, which necessarily means that the weighting factor in (4) should always grant the path enough clearance for the vehicle away of all obstacles based on the vehicle effective radius R_V. Calculation

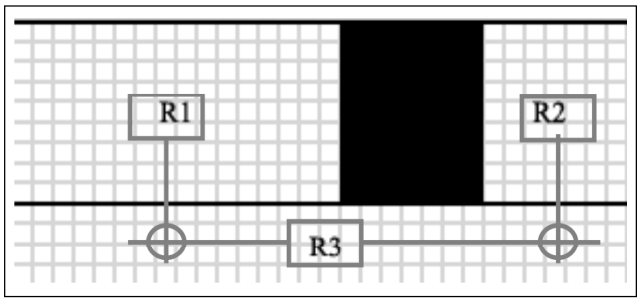

Figure 11: Introducing New Nodes

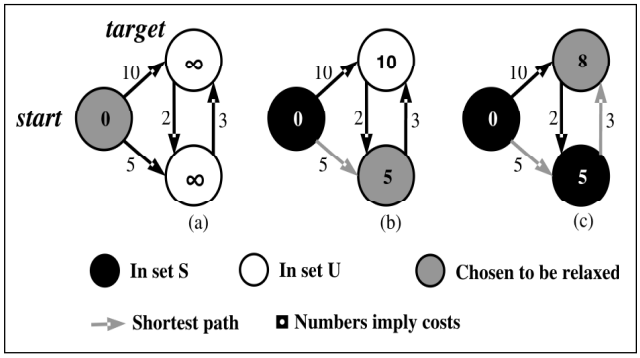

Figure 12: The Execution of Dijkstra's Algorithm

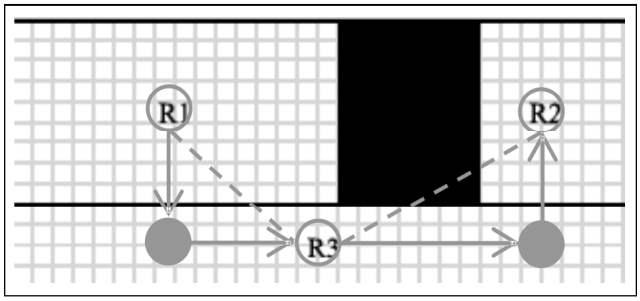

Figure 13: Dashed Lines for the Original Unsafe Path while Solid Ones for the Safe Path

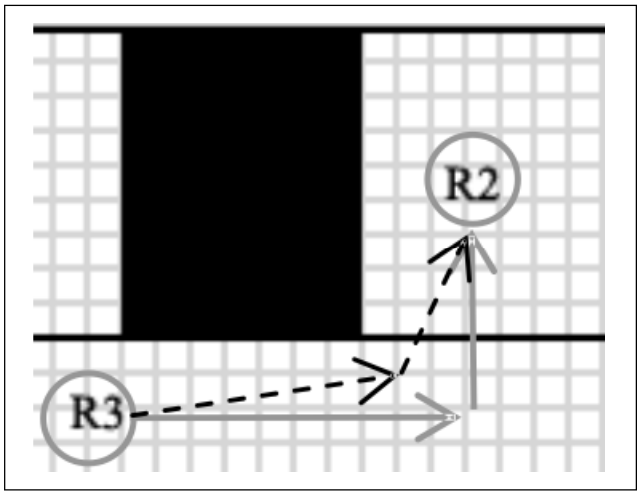

Figure 14: Optimised Path

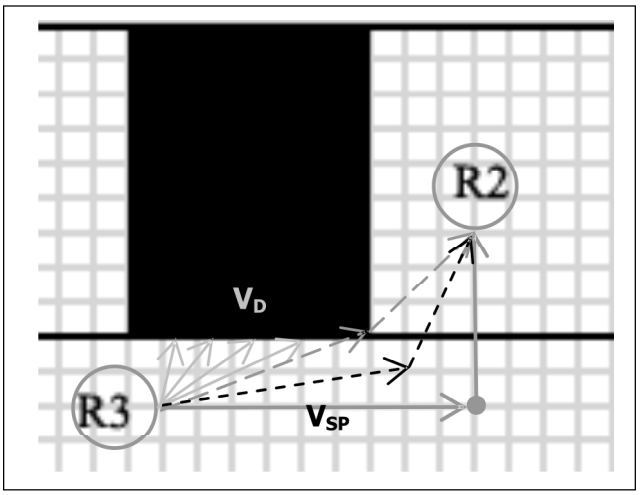

Figure 15: Optimised Path

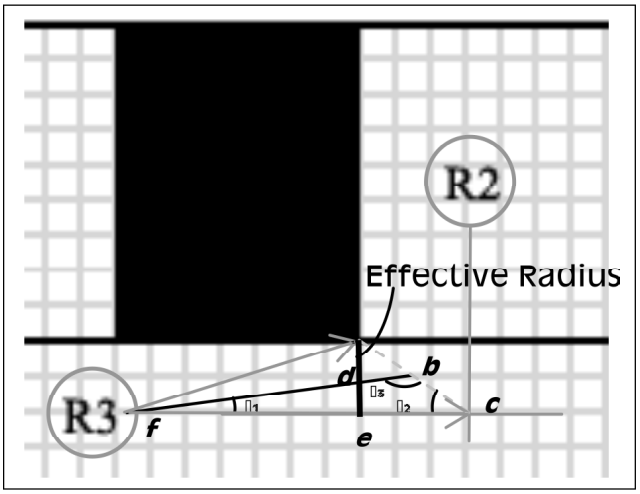

Figure 16: W_{min} Calculation

of the minimum value of the weighting factor W_{min} can be done by obtaining the distances $\overline{ab}$ and $\overline{ac}$ where

$$W_{min} = \frac{\overline{ab}}{\overline{ac}} \tag{7}$$

$\overline{ab}$ and $\overline{ac}$ are found using the triangular calculations in Table 4.

Table 4
Triangular Calculations Needed to Compute the Weighting Factor

$\theta_1 = \tan^{-1}\frac{\overline{de}}{\overline{ef}}$	$\theta_2 = \tan^{-1}\frac{\overline{ae}}{\overline{ec}}$
$\theta_3 = \Pi - \theta_1 - \theta_2$	$\frac{\overline{fc}}{\sin\theta_3} = \frac{\overline{bc}}{\sin\theta_1}$

4. IMPLEMENTATION AND RESULTS

This section presents results of actual experiments that we held in our laboratory for the presented methodology. The environment configuration for our experiments, shown in Fig. 17, emulates an aerial view, using cameras fixed on lab ceiling, providing discrete frames of the scenes on land. The ground vehicle, which is a mobile robot, starts its journey following the path provided and, hence, avoiding obstacles to its desired location.

The first part of our experiments shows success in detecting and tracking the object. Fig. 18 shows two views with different orientations of our vehicle taken from two consecutive frames. Table 5 presents the results of the Hu invariant moments calculation of both views and shows no significant difference in both values giving a sharp indication that we are dealing with the same object.

Table 5
Hu Moments for the Orientations in Fig. 16

	*h1** *10-3*	*h2** *10-7*	*h3** *10-10*	*h4** *10-11*	*h5** *10-22*	*h6** *10-16*	*h7** *10-21*
(a)	1.74	1.40	1.92	3.43	-9.35	9.42	-2.62
(b)	1.72	1.40	1.91	3.08	-7.22	1.2* 10-15	-2.65

The tracking part of the KLT algorithm was successfully adapted to our application as shown, in Fig. 19, where two successive frames are tracked using the selected vehicle feature marked with a red dot.

Fig. 20 shows the detected regions of the aerial environment scene with a number on the centre of each region. Fig. 21 presents the binary reading of the frame followed by Fig. 22 which introduces the new way nodes beside the original ones. Fig. 23 shows the path between two selected nodes with different values of the weighting factor, while Fig. 24 shows the final path used by the vehicle.

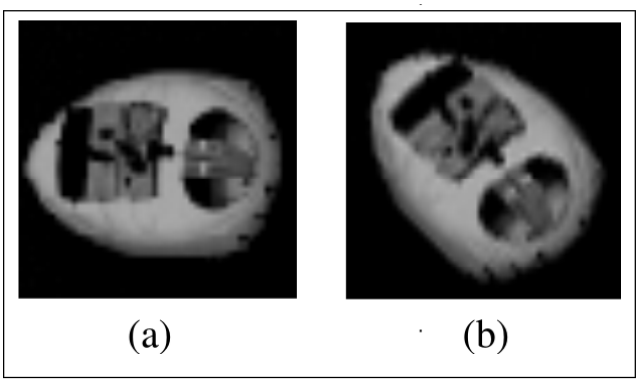

Figure 18: Vehicle's Orientation in Two Frames

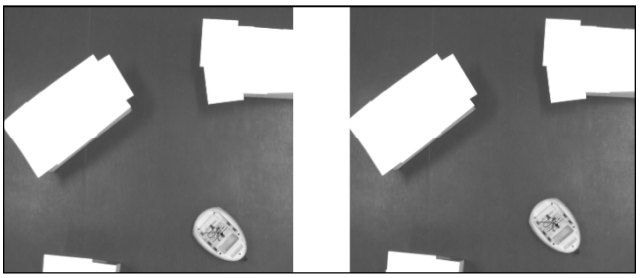

Figure 19: Successive Frames with KLT

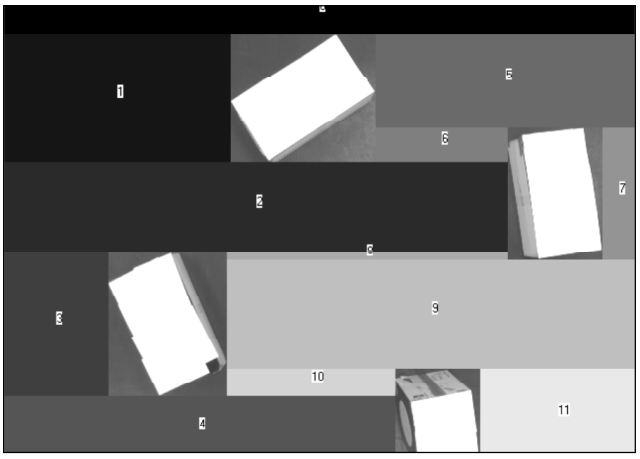

Figure 20: Detected Regions of the Environment

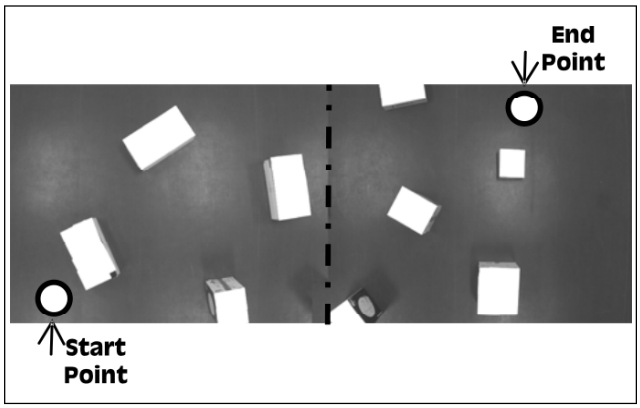

Figure 17: Our Test Environment

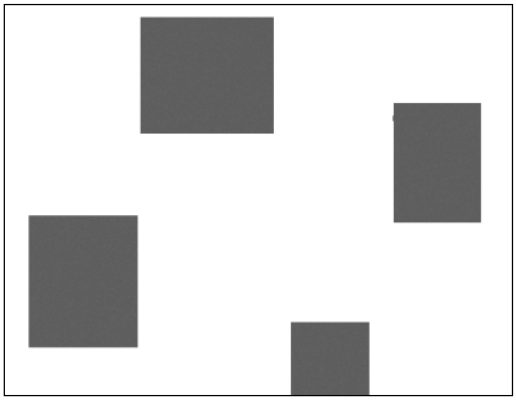

Figure 21: Binary Reading of the Environment

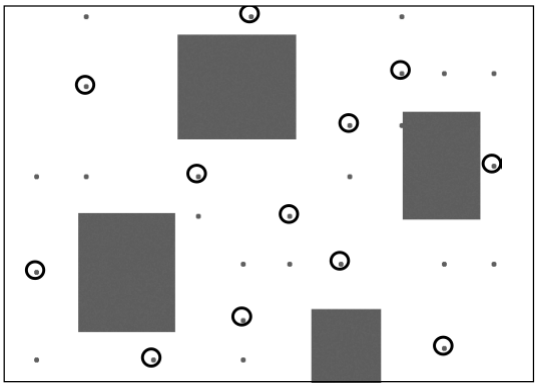

Figure 22: Nodes Generated for the Path, the Original Nodes are Circled

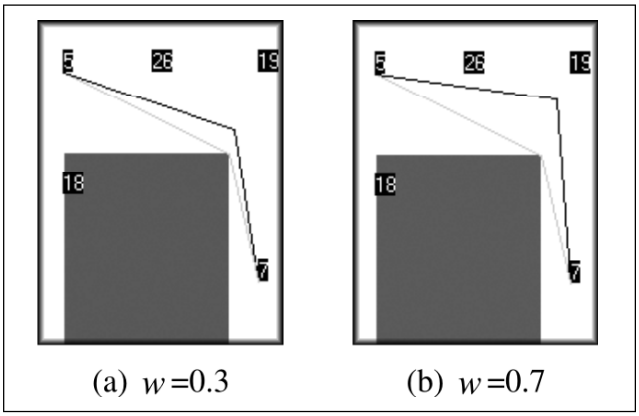

Figure 23: Path Obtained between Two Nodes with

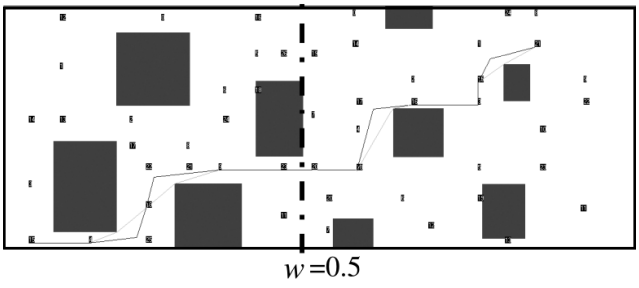

Figure 24: Final Path Followed by the Vehicle

5. CONCLUSIONS

In this paper, we have presented an integrated methodology for robot's detection, tracking and path planning in an unknown environment. External video cameras on cooperating aerial vehicles are deployed to provide a large view over the ground. The robot agent is first detected and then tracked throughout its journey. A fast, easy-to-implement, and efficient path planning algorithm is introduced to afford an optimised short and safe route from the agent's start point to its desired target.

The contributions of the paper are divided into two sub parts. The first one is of detection and tracking. The tracking technique builds on the KLT feature tracking algorithm which has a solid theoretical foundation. The feature extraction process is done in a new, simple yet effective, way that proved high reliability in the experiments. The second contribution part is in providing an optimal route for the robot to traverse in the sense that the final path reflects a combination of short distance and low risk of colliding with obstacles. After the environment is interpreted into a graph network, Dijkistra algorithm can find the shortest path but the output result has to be optimised to take the geometry of the robotic into consideration.

The methodology presented in this paper provides a set of feasible and practical algorithms for a global solution of path planning. The method's performance limitation is mainly related with the number of moving objects in the scene. The future avenues of this work include investigation of computing safe and short paths for multiple robotic agents simultaneously.

References

[1] J. Barraquand, B. Langlois, and J. C. Latombe, "Numerical Potential Field Techniques for Robot Path Planning," in *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR.*, 1991, **2**, 1012-1017.

[2] A. Stentz, "Optimal and Efficient Path Planning for Unknown and Dynamic Environments," Carnegie Mellon University 1993.

[3] R. A. Jarvis, "Collision-free Trajectory Planning using Distance Transforms," *Mechanical Engineering Trans. of the Institution of Engineers*, Vol. ME10, 1985.

[4] P. Khosla and R. Volpe, "Superquadric Artificial Potentials for Obstacle Avoidance and Approach Superquadric Artificial Potentials for Obstacle Avoidance and Approach," in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, 1988, 1778-1784, **3**.

[5] O. Khatib, "Real-time Obstacle Avoidance for Manipulators and Mobile Robots." **5**: Sage Publications, Inc., 1986, 90-98.

[6] H. Kao-Shing and J. Ming-Yi, "Global Path Planning of Mobile Robots Based on Propagating Interface Technique Global Path Planning of Mobile Robots Based on Propagating Interface Technique," in *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, 1999, 662-667, **4**.

[7] B. Faverjon, "Object Level Programming of Industrial Robots Object Level Programming of Industrial Robots," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, 1986, 1406-1412.

[8] M. K. Hu, "Visual Pattern Recognition by Moment Invariants Visual Pattern Recognition by Moment

Invariants," *Information Theory, IEEE Transactions on*, **8**, 179-187, 1962.

[9] J. Shi and C. Tomasi, "Good Features to Track," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, 1994.

[10] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *IJCAI81*, 1981, 674-679.

[11] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features," Carnegie Mellon University 1991.

[12] R. Y. Tsai, "An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision.," in *IEEE Conference on Computer Vision and Pattern Recognition* Miami Beach, 1986.

[13] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, **1**, 269-271, 1959.

[14] T. H. Cormen, *Introduction to Algorithms*. Cambridge: MIT press, 2001.