3D Reconstruction using Underwater Stereo Vision & Scene Classification

Assignment Task 2 Report

University of Sydney

Vishant Prasad - 470416309

AMME4710 – Computer Vision & Image Processing

Contents

Abstract & Background

The following document highlights the process of constructing two overall programs and carrying out investigations utilizing these two systems. This includes a **3D Reconstruction using Underwater Stereo Vision**, whereby, given stereo calibration and camera pose data is utilized alongside comparison to true terrain data. The second program includes a **Scene Classification/Recognition** system that can identify and classify specific objects in given images and assign semantic labels that describe the context of the environment.

Keywords: Stereo Vision

3D Reconstruction using Underwater Stereo Vision & Scene Classification

Question 1 – 3D Reconstruction using Underwater Stereo Vision

The following section highlights the investigation of utilizing Stereo Vision techniques in order to 3D reconstruct a region of underwater environment. More specifically, utilizing a collection of left and right perspective images captured by a camera.

Introduction

Stereo Vision is the calculation of depth or extraction of 3D information from digital images. This is calculated through binocular disparity between the images of an object via both left and right perspectives or eyes of the camera. The method requires the matching of the two perspectives – identifying common features of the left and right perspectives. These are the identification of common points more specifically in a visual scene. The applications of Stereo Vision include industrial inspection for 3D objects (quality control, deformation analysis, food inspection and defect analysis), scanning electron microscope, 3D sensing tools (three dimensional measurement of objects – *Figure 1*) and surveillance (motion tracking and object tracking to measure paths), 3D growth monitoring and radar.[5]

Figure 1- SceneScan Product Line (3D Sensor - Stereo Vision Technology)

(Industrial Applications – Nerian Vision Technologies, n.d.)

Methodology

Stereo Calibration & Loaded Data

The methodology of the program includes firstly configuration of the two perspectives and sets of images collected from the camera – left and right images from respective eyes. This would encompass the calibration of the camera through geometric models – more specifically, a set of equations that will allow for th prediction of expected pixel locations for a certain common point that lies on the image. This prediction would include gaining information on the given point's location in a 3D space, the camera's orientation and position in 3D space and parameters describing the camera's optical system. The stereo calibration has been collected and compiled into a file for ease of loading into the program alongside a reference of the real terrain for comparations and the camera pose data. The model uses a pinhole assumption to represent the undistorted image coordinates (u, v):

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} Z^c = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X^W \\ Y^W \\ Z^W \\ 1 \end{bmatrix}$$

Equation 1 – Camera Pinhole Projection Model

This is followed by the parametric model for radial and tangential distortion to arrive at the real, distorted pixel coordinate in the image (u_d, v_d):

$$u_d = f_x x_d + u_0$$

$$v_d = f_y y_d + v_0$$

Equation 2 & 3 – Distorted Pixel Coordinates

$$x_d = x(1 + k_1 r^2 + k_2 r^2 + k_3 r^6) + 2k_4 xy + k_5(r^2 + 2x^2)$$

$$y_d = y(1 + k_1 r^2 + k_2 r^2 + k_3 r^6) + k_4(r^2 + 2y^2) + 2k_5 xy$$

Equation 4 & 5 – Distorted Normalized Image Coordinates (x & y)

$$x = \frac{X_c}{Z_c} = \frac{u - u_0}{f_x}$$

$$y = \frac{X_c}{Z_c} = \frac{v - v_0}{f_y}$$

$$r^2 = x^2 + y^2$$

Equations 6-8 Radial Positions of Objects w/respect Optical Centre of Camera

Using these equations, camera calibration process is completed as the focal lengths (f_x, f_y), principle point (u_0, v_0) and lens distortion coefficients $(k_1, k_2, k_3, k_4, k_5)$. Furthermore, with stereo-imaging systems, the extrinsic relative rotation matrix and translation vector (R, t) between the two cameras is also estimated.

Furthermore, calibration was performed using comparations with differing fundamental matrix computations using a function that has been created and included within *Appendix Section A.*

Feature Extraction

The program design for the feature extraction involves detecting and matching features (interest points) across stereo pairs for use in the triangulation algorithm. Within feature extraction the program finds *Speeded Up Robust Feature (SURF)* points in an image and returns descriptor vectors association with these detected feature points. Once feature descriptors have been extracted, a set of matches between two sets of features in two images can be computed. The feature extraction can also compute descriptors for a range of other interest point feature types that work in a similar fashion to SURF points. The matches found using feature descriptors typically contain outliers – pairs of corresponding points that are not referring to the same object in 3D space. A method of detection and rejection of outliers is through maintaining only the set of matching points that align with epipolar geometry (or triangular algorithm) between the two image perspectives. The program utilizes therefore, a robust estimation of the fundamental

matrix $\boldsymbol{F}$, representing the relative camera geometry using a RANSAC-style algorithm(s) to estimate $\boldsymbol{F}$ from a set of points correspondences when finding the set of inlier correspondences.

Feature Types

The investigation encompassed two feature detection types including **SURF Detection** as well as **Harris Feature Detection**. Through experimentation that is documented in the Results & Analysis section of this document, a feature type that resulted in more valid results was selected. The SURF detection algorithm implemented begins by looking for potential features by building a succession of images at increasing rate of blurred images looking at differences in the image intensities between subsequent blurring levels. Therefore, feature extraction using these methods was selected as it has been shown to be robust to noise, viewing angle and is distinct to the local image intensity information.[9]

Methods for Computing Fundamental Matrix

The investigation experimented with two model fitting methods or algorithms including **Random Sample Consensus (RANSAC), Least median of Squares (LMeds) Method** and **M-estimator Sample Consensus (MSAC)**. These models are preferrable and this overall *Fundamental Matrix* is included as repeating the sampling computation for a certain large number of iterations, the chances of detection of the correct and incorrect points is of higher probability. The procedure of the model fitting includes drawing N points/pixels uniformly at random, fitting the model to these N points – computing the line parameters for the number of points N, finding the inliers to this line among the remaining points – points whose distance from the line is less than a defined threshold within the program (t), decide if there are more inliers, accept the line and possibly refit using all inliers (via least-squares approach) and repeat the procedure from one to N times. In summation, the general method is suitable to application such

as this investigation, is easy to implement, calculate failure rate and therefore, would be desirable to use either RANSAC, LMeds or MSAC. Finally, these methods are able to deal with models that have higher-dimension without lots of memory/computational time, such as that necessary in a 3D space.[1]

Triangulation

The triangulation and epipolar geometry between the two views and a set of stereo camera parameters utilizes computation of the 3D coordinate at a point seen at a pixel location (u_1, v_1) in the first image and at the pixel location of a second image (u_2, v_2). Furthermore, the algorithm utilizes the stereo parameters that are loaded through the camera data compiled earlier – contains the parameters for the image pair. Within the program, the triangulation is 3D points with respect to the left camera's frame.

World Frame Conversion

The program requires translation and rotating of the XYZ coordinate system of the camera's frame to the world coordinate reference frame. This is accomplished by implementing an algorithm that processes the rotation and translation (R, t) of the X^W points using an equation derivation defined below:

$$X^C = C_w^C(X^W - X_{Cw}^W) \rightarrow X^C = RX^W + t$$

$$X^W = \frac{1}{R}(X^C - t)$$

Equation 9 – Derivation of World Frame Transformation

where, X^C is the X coordinate in the camera frame, X_{Cw}^W is the relative position of the camera with respect to the world frame and $C_w^C = R$ is the direction cosine (or rotation) matrix which transforms coordinates from W to C frame.

Visualization & Comparison

The outputted points or world frame coordinates stored can be plotted against the provided terrain model data through mesh plots and 3D point cloud which contains the locations, intensities and RGB colors to render the point cloud. The plot itself requires the inverse of the *Z* values to make the plot close to the given terrain illustration. Comparison of results within this investigation has been performed solely through overlaying and visually differentiating the calculated coordinates to the given terrain.

Methodology Evaluation

The features have been successfully extracted through testing both methods of **SURF Detection** as well as **Harris Feature Detection**. In the final program that has been developed for this investigation, the SURF Detection method provided greater extraction features with the final result of the terrain illustration in point cloud being more accurate to the given terrain data. Contrast to SURF Detection being desirable for object detection, this method was still more applicable as it provided feature descriptors that can be used to compare key points when performing the fundamental matrix computation. The Harris Feature Detection was more simplified defining the corner, edge, and flat regions of an image through detection of change in directions.

The final world-referenced point cloud was constructed as highlighted in *Figure 2* and *Figure 3*, using the individual triangulations for each stereo pair.

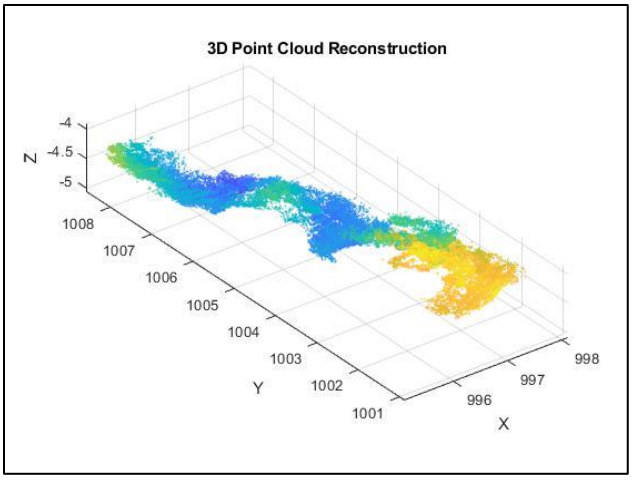


Figure 2 - 3D Point Cloud Reconstruction

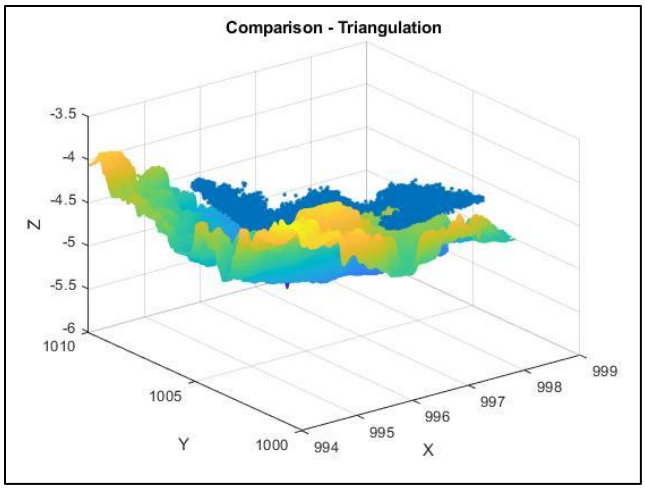


Figure 3 - Given Terrain with Reconstruction Overlay

Results & Analysis

As discussed earlier, the SURF Detection method was favored over the Harris Detection method. The Harris Detection method involves computation of derivatives (gradients) at each pixel, computation of second moment matrix (M) in a window around each pixel using gradients, compute corner response function (M_C) from M, find points with large corner response $(M_C > threshold)$ and take the points of local maxima of M_C. In comparison with SURF where there is an approximate Gaussian second derivative mask applied to an image at many scales with feature detector masks along each axis at 45 degrees to the axis making it more

robust to rotation than the Harris feature or corner detection method. Therefore, SURF denotes more viable results due to its usefulness for object detection – can detect a series of interest points in both a source image and template image and look for our object by finding matching sets of correspondences between images. The results or comparison between the two methods are illustrated below in *Figure 4* and *Figure 5*.

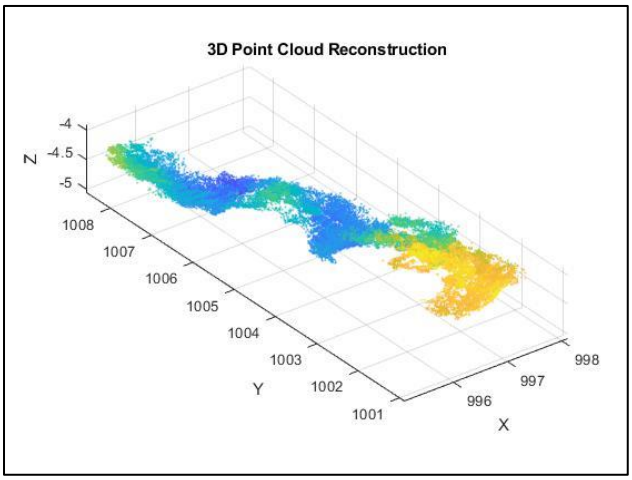


Figure 4 - Surf Detection

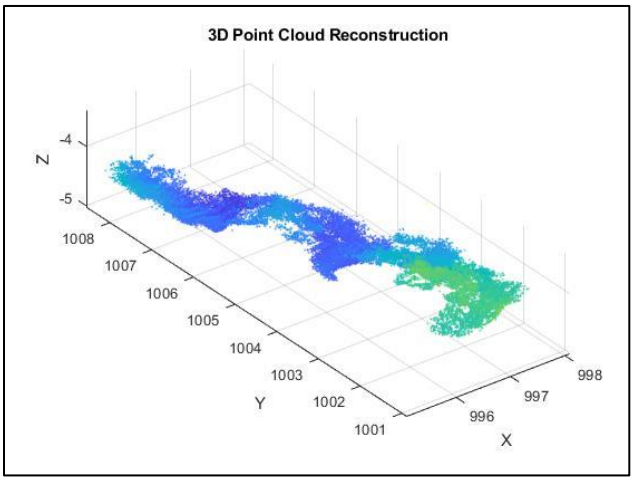


Figure 5 - Harris Detection

With testing different extraction parameters, specifically with different methods of computing the fundamental matrix, the MSAC, LMeds and RANSAC were decided to be the most appropriate to test with. LMeds and MSAC were found to be the most viable to employ into

this program as LMeds can be used since at least 50% of the matched points detected are inliers. MSAC can also be employed since, we can set the distance threshold for the inlier as mentioned earlier in the program methodology. Furthermore, MSAC proved to be a method that converges quicker than the RANSAC method. In comparison between LMeds and RANSAC, LMeds seeks for the narrowest stripe while RANSAC seeks for the most densely populated stripe of the user data. LMeds also requires no threshold, improving computation time for determining the inlier. Finally, MSAC would be desirable over RANSAC as RANSAC does not consider the amount of error within bounded threshold and hence, worse accuracy than the other two methods. Overall, through the comparison, all three methods are quite similar. The program currently and finally was recommended to utilize MSAC as it is the most accurate among the three estimators.

Discussion & Conclusions

The final effects of these various different types of methods and features that were employed are illustrated in the figures as mentioned. The final accuracy had a slight increase with MSAC method in comparison with the LMeds and RANSAC (referenced from various sources and through this investigation to be minute - $2-5\%$ increase in accuracy from comparison.[1]

The common issue with the triangulation found, to be common amongst all differing methods of computation and feature detection is the y-axis plot or coordinate within the overlay as seen in *Figure 3*. The results also denote that the trajectory is the most similar.

Improvements to the performance of the program, therefore, can be made using further research and investigation into further fundamental matrix computation techniques and methods such as **Normalized Eight-Point Algorithm** and **Least Trimmed Squares** (LTS). Furthermore,

points accuracy can be improved by coincidingly implementing *Dense Stereo* techniques allowing for specifically locating 3D depth for each and every pixel in one of the two camera images via *Dense Stereo Triangulation*. Alike, computation time of the program can be improved through implementation of **Stereo Rectification**, with reduced implementation complexity than searching along a non-horizontal line.

Question 2 – Scene Classifier

The following section highlights the design and implementation of a **Machine Learning Algorithm** that can perform scene classification based on RGB images. The investigation will include development of an algorithm through pre-processing data, feature extraction, model training, validation and testing and parameter selection.

Introduction

The term scene classification refers to classifying a scene image to one of a predefined scene category by comprehending the entire image. This has been a large and fundamental standing issue or investigation as apart of computer vision. Scene image recognition provides valid context information with an application in boot target recognition, re-texturing of video games and graphics, fields of research in remote sensing, aerial scene classification and in fields of computer vision and robotics. In robotics, in order to determine the robot's position and orientation in its environment in real time is an example of scene classifications application.[8]

As part of the process of designing such a program, **Semantic Segmentation** is utilized along with training of a model which in this case investigates the **Support Vector Machines** (SVM), **K-Nearest Neighbor** (KNN) and **Decision Trees** model approaches. Within image classification, the algorithm itself can be summarized by processing an input image, extracting

features, generating an outputted feature vector that can be sent to the classifier and the class category is outputted.

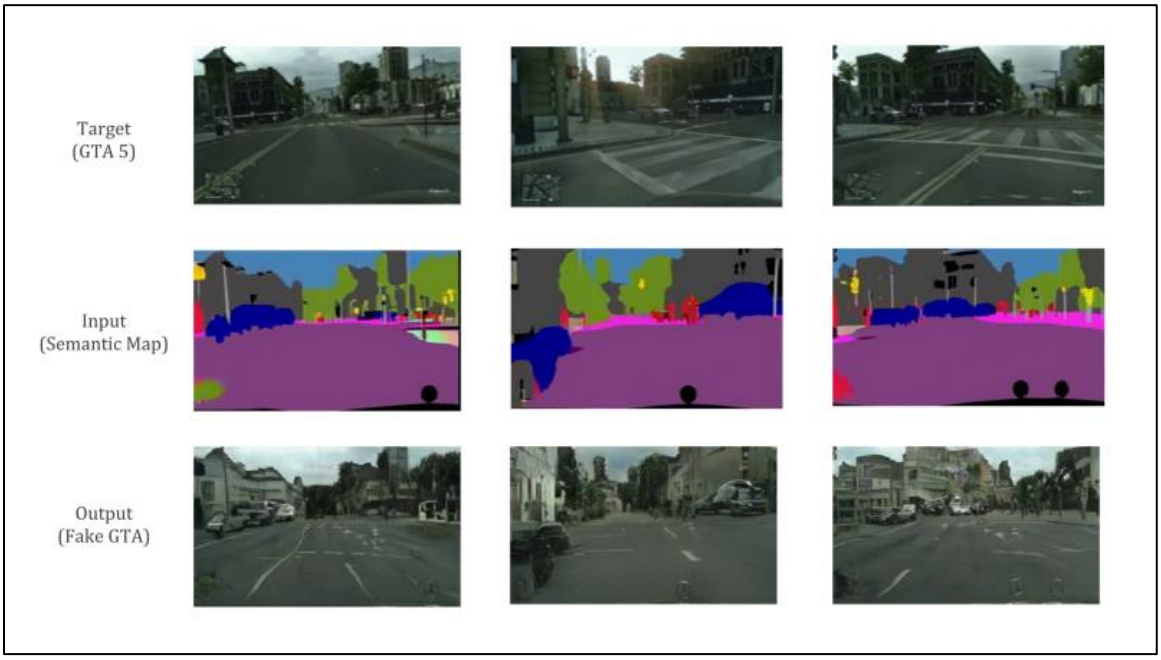


Figure 6 - Example of Video Game Scene Recognition & Graphics Adjustment

(Trivedi, 2018)

Methodology

Preprocessing of Images

The initial setup of the program involves setting the primary classifications or labels that will be assigned to images. In total there are **seven** classes listed as *ball pit, desert park, road, sky, snow* and *urban*. Therefore, the overall objective is to properly assign a label to an image that is inputted into the program. The assumption of this investigation is to aim to ideally have one correct outputted text label that correctly identifies the image. The pre-processing of the image within the function that has been created for the classification of the images is included within the feature extraction.

Feature Extraction

The feature extraction utilizes graying of the image and applying to histogram equalization in order to gather properties of the image for generation of the feature vector. Furthermore, the function blurs the image through a 2D filter applied with adjustable factors. Moreover, the function gathers the average RGB values of this blurred output for this feature vector. Finally, we generate a color space, whereby the MATLAB color threshold application is used to isolate the red, blue, green, white, and gray color proportions. Ideally within these isolations and color manipulations, the aim is to test response to different filters. The function then involves detection of the circular objects in the image using morphological structing element and Hough transform for the feature vector generation. Furthermore, the function utilizes Harris Corners feature detection with adjustable parameters, specifically experimenting with the quality of the corners that are to be passed through the function. In order to gain the textural properties, the function has been designed to average texture histograms using the gray blurred images.

Model Development

The model development methodology for this program was primarily referenced from MATLAB itself in order to save and generate a SVM trained model. SVM involves kernel type, lambda, and c-value within hyperparameter tuning, allowing for the control of the algorithm functions. The optimal parameters were determined by the cross-validation performance with the final test set performance and evaluation on the best model defined to be with a value of **four** defining the integer dictating how many folds are to be used during validation. This was decided through repeated experiments of K fold values and the values as increased denoted that **three** to

four denoted an optimal performance with low training error and test error – *Figure 7-8* displays an example with testing K-fold values and results still requiring tunning.

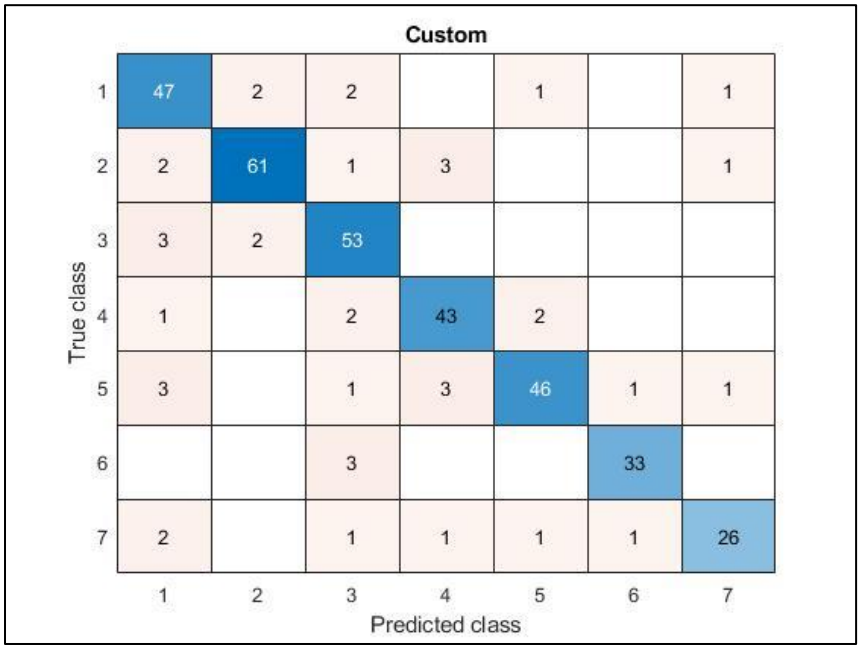


Figure 7 - Low K-Fold Values Experimental Results

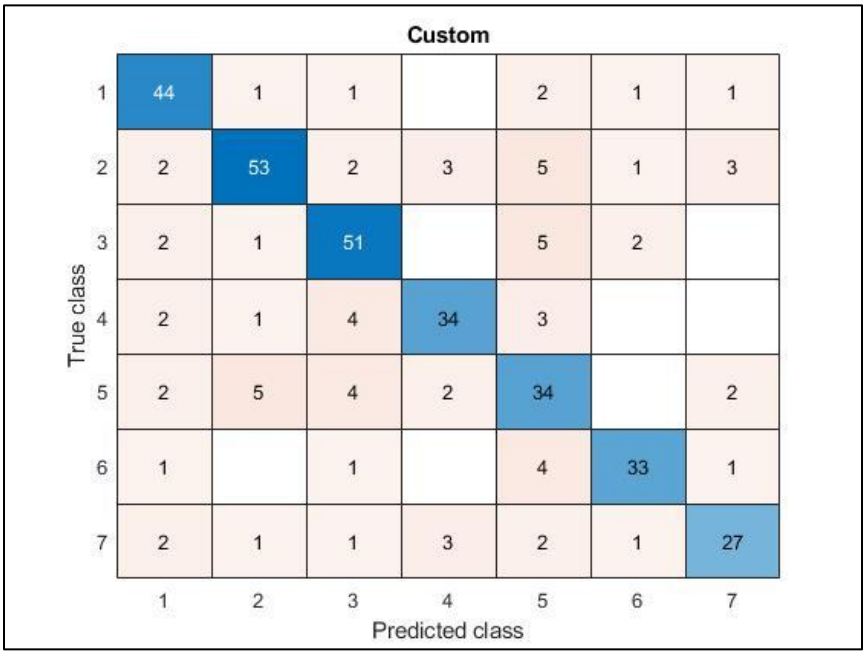


Figure 8 - High K-Fold Values Experimental Results

When designing the analysis model, the confusion matrix, accuracy, precision, recall, and score is utilizing matrices with **seven** classes representing the differing environments that are to be detected. Through comparisons of resulting predicted labels from different models, including as mentioned KNN, Decision Tree and SVM plots in *Figure 9-11* denote the results. Furthermore, by investigating SVM to have a great accuracy in predicted the models as the confusion chart denotes valid results with the lowest errors or misclassifications.

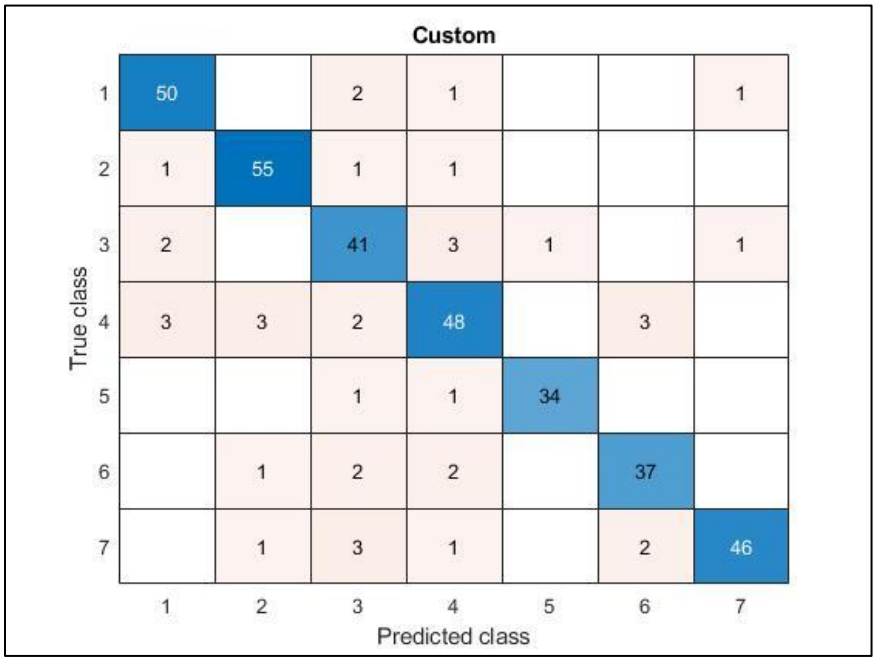


Figure 9 - SVM

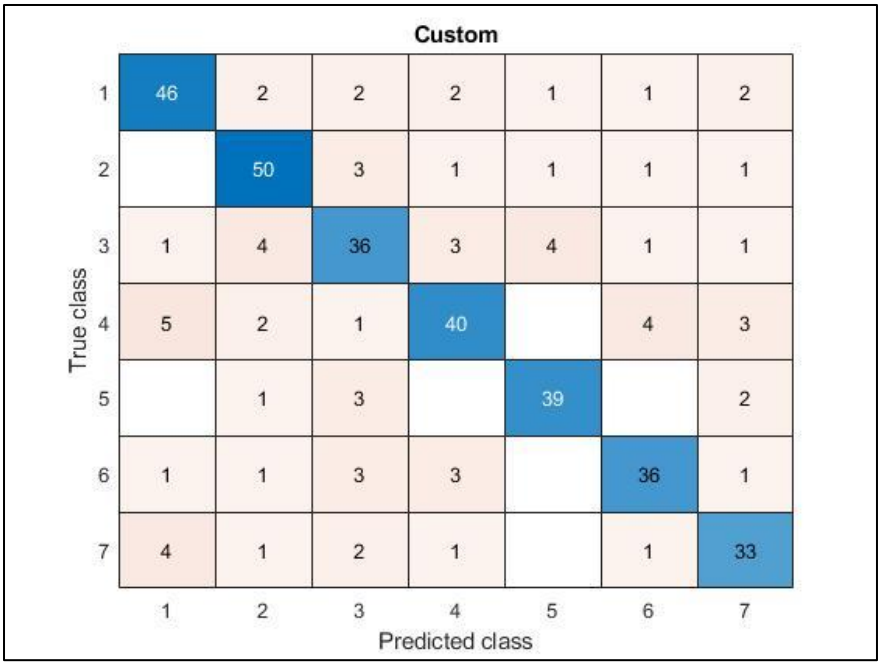


Figure 10 – KNN

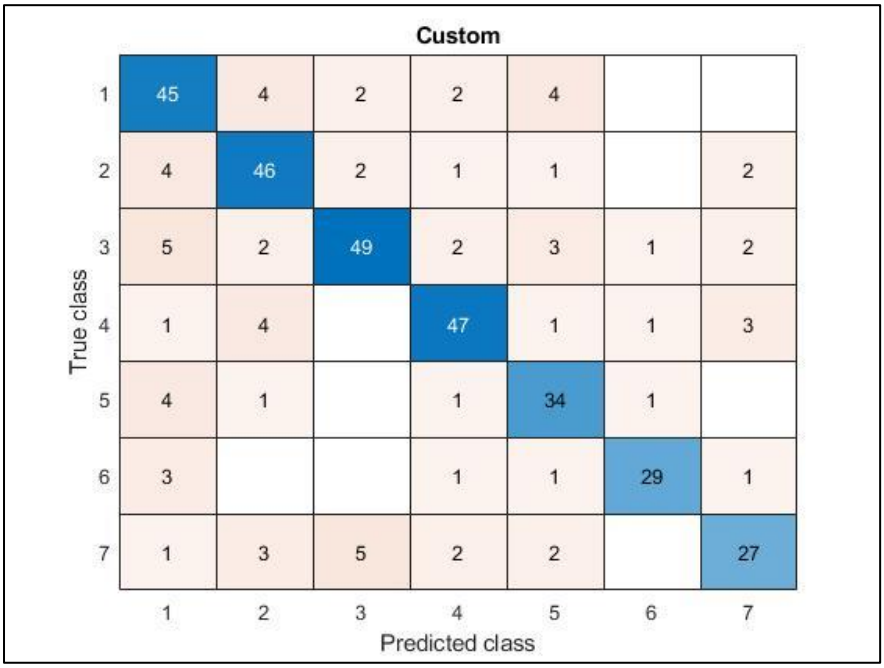


Figure 11 - Decision Tree

Metric Decisions

The metric decisions include a selection of Harris Corners and Hough Transforms when performing feature extraction. This was tested alongside a combination of SURF Detection (with

Harris Feature Detection) which denoted that the former decision provided a greater result with feature extraction and hence, overall results in the model creation. Furthermore, within the analysis model, a binary classifier performance evaluation is applicable since we have either a (0/1) if the classification passes. Therefore, we use the following code segment to represent the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) over the classes:

```
TP = confmat(k,k);
FP = sum(confmat(k,:))  - TP;
FN = sum(confmat(:,k))  - TP;
TN = sum(confmat,'all') - TP - FN - FP;
```

Therefore, resulting accuracy, recall, score, and precision utilize a top-7 as the competition dataset metrics for this problem or investigation is reasonably large. With the cross validation of K folds, multiple splits of the training/testing data is performed with the metrics averaged over the K folds. The measured total expected performance suitable, as mentioned, was of K folds approximately **three** to **four**. Precision and recall were metrics that were tuned for the algorithm to output a reduced false positive and false negative value by investigating the trade-off effects between the two.

Results & Analysis

The investigation displays that the function for scene classification utilizing the pre-trained SVM prediction model is effective with results denoting its performance in the form of a Confusion Matrix with specific scenes detected and errors minimized through the selection of pre-processing images (utilizing color space and histogram equalization). Furthermore, the final program includes the feature extraction methods of Harris Corner Detection and Hough Transform – finding circles. The results of training and performance are denoted in *Figure 12*-

13. More results upon the differing scenes and classifications are evident within the *Appendix Section B.*

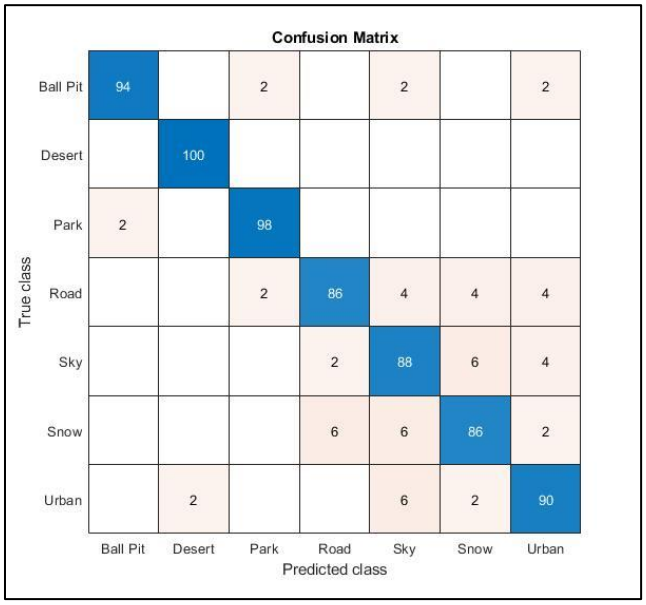


Figure 12 - Confusion Matrix - SVM (Final Performance)

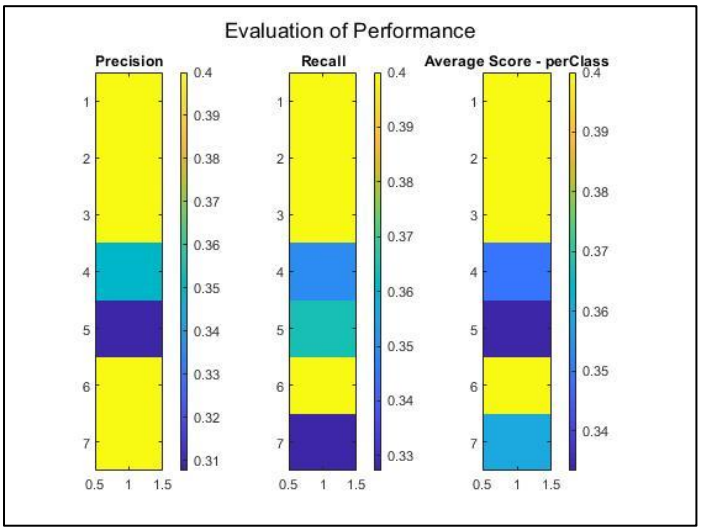


Figure 13 - Evaluation of Performance (SVM)

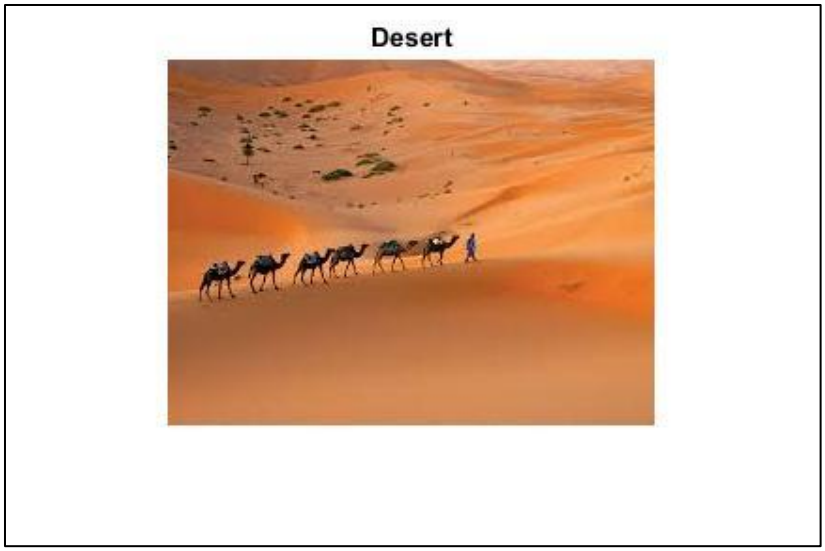


Figure 14 - Example of Function Output - Scene Classification

Discussion & Conclusions

Misclassifications are evident with specific images and data that has been tested within this investigation, with clear reasoning being due to the inclusion of different aspects of a range of some of the datasets being prevalent in a single image. Therefore, a collision takes place or overlap, whereby, the model is predicting a combination of environments or is choosing a scene over the precedence of another (seen the program has been designed to output only one label for an image). Furthermore, some images that have unknown objects or figures can act as noise and hence, through preprocessing and feature extraction, these objects are mistaken, cause collisions or disrupt the prediction model causing the scene to be misclassified.

The primary difference between Machine Learning algorithms investigated through this program and those of state-of-the-art design is the inclusion of such higher quality programs and algorithms utilizing a combination of Neural Networks and Artificial Intelligence, coinciding with their Deep Learning and Machine Learning algorithms. More specifically, Artificial Intelligence for example is a technique that enables the machines to mimic closer human classifications and hence, greater accuracy with a deeper resemblance of scenes that would be

closer to human environments. Furthermore, tying this to a Neural Network would mean ideally the algorithm is constantly improving with greater datasets to train from – ultimately, they possess greater computation power and are larger in algorithm scale.[7]

To increase the accuracy and performance of the investigation undertaking within this report (with relevant and realistic scope), feature extraction can be improved with the use of various other color spaces and investigations – HSV & LAB manipulation for example.

References

[1] Choi, S., Kim, T. and Yu, W., 2009. *Performance Evaluation of RANSAC Family*. [online] bmva.org. Available at: <http://www.bmva.org/bmvc/2009/Papers/Paper355/Paper355.pdf> [Accessed 10 October 2021].

[2] Goswami, R., 2021. *COMPARISON OF STATE-OF-THE-ART MACHINE LEARNING BASED DATA DRIVEN AND MODEL UPDATING METHODS AGAINST SHALLOW AND DEEP CONVOLUTIONAL NEURAL NETWORKS METHODS OF STRUCTURAL DAMAGE DETECTION*. [online] Research Gate. Available at: <https://www.researchgate.net/publication/348236914_COMPARISON_OF_STATE-OF-THE-ART_MACHINE_LEARNING_BASED_DATA_DRIVEN_AND_MODEL_UPDATING_METHODS_AGAINST_SHALLOW_AND_DEEP_CONVOLUTIONAL_NEURAL_NETWORKS_METHODS_OF_STRUCTURAL_DAMAGE_DETECTION> [Accessed 10 October 2021].

[3] Gupta, S., 2020. *Natural Scene Recognition Using Deep Learning*. [online] Towards Data Science. Available at: <https://towardsdatascience.com/natural-scene-recognition-using-deep-learning-91b6ba86bad5> [Accessed 10 October 2021].

[4] Jiafa, M., Weifeng, W., Yahong, H. and Weiguo, S., 2019. *a scene recognition algorithm based on deep residual network*. [online] Taylor & Francis. Available at: <https://www.tandfonline.com/doi/full/10.1080/21642583.2019.1647576> [Accessed 10 October 2021].

[5] nerian.com. n.d. *Industrial Applications – Nerian Vision Technologies*. [online] Available at: <https://nerian.com/applications/industrial-applications/> [Accessed 10 October 2021].

[6] Pingel, J., 2019. *Scene Classification Using Deep Learning*. [online] MathWorks. Available at: <https://blogs.mathworks.com/deep-learning/2019/11/25/scene-classification-using-deep-learning/> [Accessed 10 October 2021].

[7] Sevakula, R., Au-Yeung, W., Singh, J., Heist, K., Isselbacher, E. and Armoundas, A., 2021. *State-of-the-Art Machine Learning Techniques Aiming to Improve Patient Outcomes Pertaining to the Cardiovascular System*. [online] Journal of the American Heart Association. Available at: <https://www.ahajournals.org/doi/10.1161/JAHA.119.013924> [Accessed 10 October 2021].

[8] Trivedi, C., 2018. *How to create realistic Grand Theft Auto 5 graphics with Deep Learning*. [online] freeCodeCamp. Available at: <https://www.freecodecamp.org/news/how-to-create-realistic-grand-theft-auto-5-graphics-with-deep-learning-cc092c4a69f0/> [Accessed 10 October 2021].

[9] Woodruff, C., 2013. *Feature Detection and Matching*. [online] courses.cs.washington.edu. Available at: <https://courses.cs.washington.edu/courses/cse576/13sp/projects/project1/artifacts/woodrc/index.htm> [Accessed 10 October 2021].

[10] Zeng, D., Liao, M., Tavakolian, M., Guo, Y., Zhou, B., Hu, D., Pietikäinen, M. and Liu, L., 2021. *Deep Learning for Scene Classification: A Survey*. [online] arxiv.org. Available at: <https://arxiv.org/abs/2101.10531> [Accessed 10 October 2021].

Appendix

Section A

Code for Calibration – Fundamental Matrix Computation:

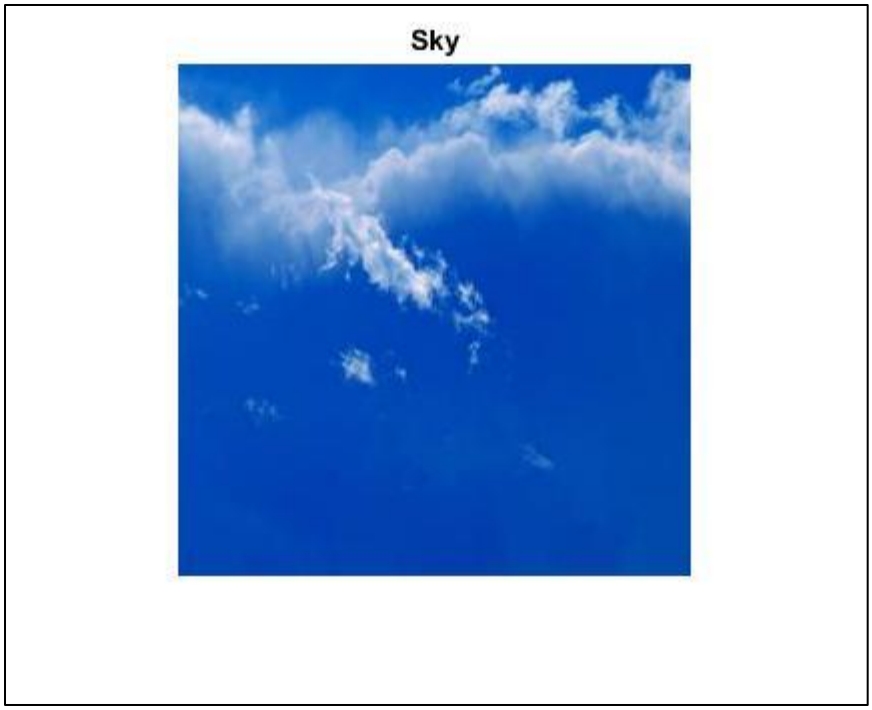
```
%Function that is used for testing and comparing metrics and parametres
%involved within the generation of a point cloud 3D reconstruction.
function [pts] = imgEvalWorldRefPtCloud(i,stereoParams,cameraPose)

    %Setting of Directories & Image Formatting
    im = im2double(imread(['images_left',filesep,...,
        char(cameraPose.left_images(i))] ));
    im_ref = im2double(imread(['images_right',filesep,...,
        char(cameraPose.right_images(i))] ));

    %Convert to Grayscale Image
    im = rgb2gray(im);

    %Storing Undistorted Image
    [~, ~] = undistortImage(im, stereoParams.CameraParameters1);
    [~, ~] = undistortImage(im_ref, stereoParams.CameraParameters2);

    %Extract interest points - using extraction methods.
    pts1 = detectKAZEFeatures(im);
%     pts1 = detectSURFFeatures(im,'MetricThreshold',500,'NumOctaves',...,
%     3,'NumScaleLevels',4);
    [fts1, ~] = extractFeatures(im, pts1);

    pts2 = detectKAZEFeatures(im_ref);
%       pts2 = detectSURFFeatures(im_ref,'MetricThreshold',500,...,
%       'NumOctaves',3,'NumScaleLevels',4);
    [fts2, ~] = extractFeatures(im_ref, pts2);

    %Calculations of corresponding pairs.
    indx_pairs = matchFeatures(fts1,fts2);

    mtched_pts1 = pts1(indx_pairs(:,1),:);
    mtched_pts2 = pts2(indx_pairs(:,2),:);

    %Displaying the matched pairs.
%     figure(1);
%     ax = axes;
%     showMatchedFeatures(im,im_ref,mtched_pts1,mtched_pts2,...,
%     'montage','Parent',ax);

    %Implementing outlier rejection - undistorting of points.
    undist_pts1 = undistortPoints(mtched_pts1.Location,...,
        stereoParams.CameraParameters1);
    undist_pts2 = undistortPoints(mtched_pts2.Location,...,
        stereoParams.CameraParameters2);

    %Calculating set of inlier correspondences based on this set of initial
```

```
    %matches.
    [~,inliersIndex] = estimateFundamentalMatrix(undist_pts1,...,
        undist_pts2,'Method','MSAC');

    %Sets of inlier correspondences.
    mtched_pts1 = mtched_pts1(inliersIndex);
    mtched_pts2 = mtched_pts2(inliersIndex);

    %Produce a Point Cloud - World Reference Frame
    %Compute of 3D locations corresponding to points located in two images.
    pts = triangulate(mtched_pts1,mtched_pts2,stereoParams);
end
```

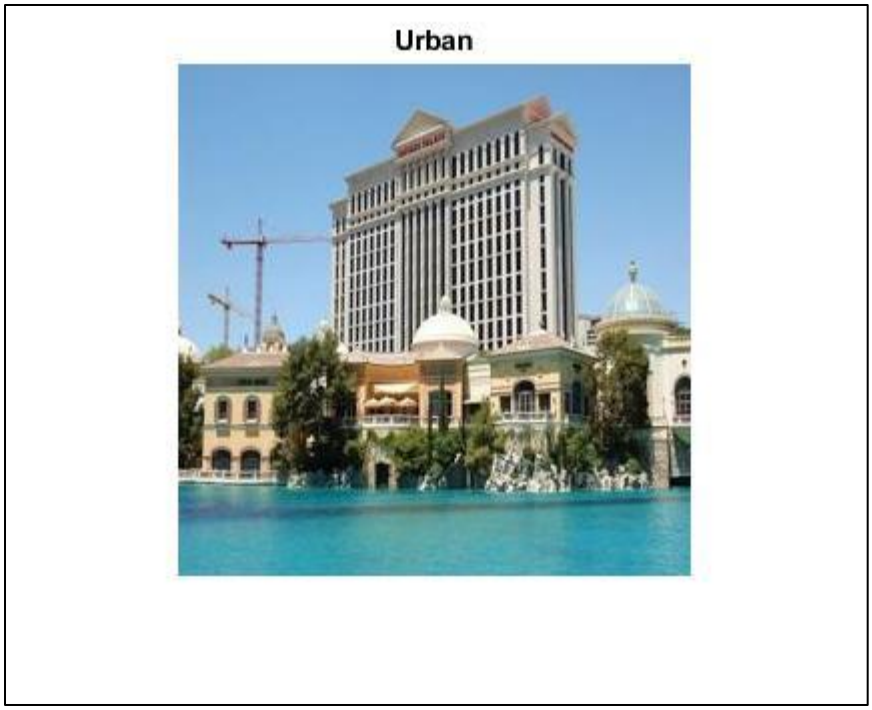
Section B

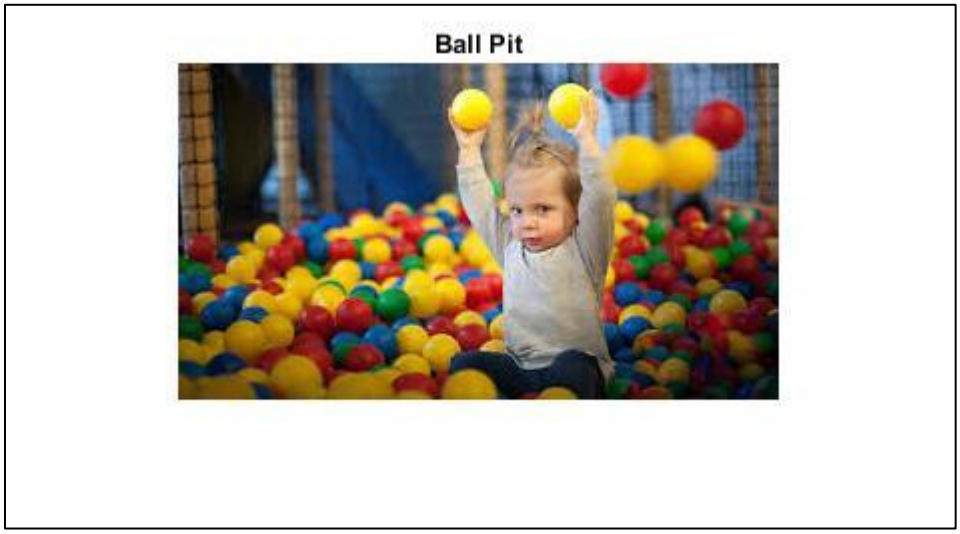
Park

Road


Sky


Snow


Urban


Ball Pit