Object Feature Mapping and View Prediction for Same-Different Determination of 3D Polyhedrons

Peter Caruana
York University
EECS 6323 Final Project
caruana9@my.yorku.ca

Abstract

This work proposes a composite algorithm for determining if two 3D polyhedrons are the same *or* different*, using only 2D images. The approach is two-fold. First an internal model of each object is made by gathering views from various viewpoints and approximating the positions and orientations of object faces. Second, the system will use these models to make predictions about what it should see from the other polyhedron by moving to specific viewpoints. Based on the correlation between prediction and expectation the system will decide if the two polyhedrons are the same or different.*

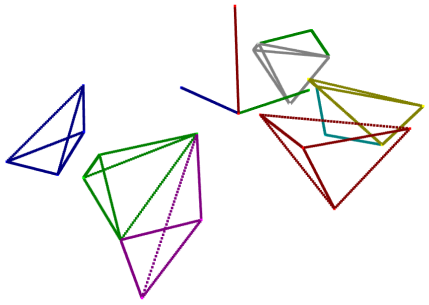

Figure 1. Visualization of mapped object features

1. Introduction

Deep learning methods, namely convolutional neural networks (CNN) have been extremely successful in a plethora of computer vision tasks. In particular, these methods are able to approach human performance for visual recognition tasks such as image classification. Given an image of an animal, most modern systems can easily determine if the image contains a cat, zebra, monkey etc. CNNs currently are the standard hammer in the vision scientists toolkit. However it must be noted that despite the immense success these networks have had, there are a number of open computer vision problems for which this approach simply has not worked. CNNs, and all feed forward neural networks for that matter, struggle to understand high-level, abstract concepts and thus cannot generalize outside of their training domain. A prime example of this is to ask the question, "Given two objects of the same class, are they the same instance of that class?", otherwise known as the same-different problem. An example would be to determine if two arbitrary shapes in an image are the same. Humans are quite good at this task, without having to have seen the shapes beforehand. CNNs on the other hand perform very poorly at this. Despite this inability to perform well on a fairly straightforward problem, contemporary machine vision research has shown very little interest, despite it being a fairly obvious question [6]. Of what work there has been within this problem domain, it is clear that feed forward neural networks are fundamentally unable to adequately learn same-different relations between images, as shown by Kim et al. [4]. Kim et al. [4] believe that feedback mechanisms such as attention, working memory, and perceptual grouping are needed to approach human levels of abstract visual reasoning.

One evident practical manifestation of this problem is in the form of adversarial attacks which exploit vulnerabilities within machine learning algorithms. These adversarial methods can generate images which are imperceptibly different to humans, however causes catastrophic results for image classifiers. An image which a network could previously classify correctly becomes unrecognizable [3][4]. This is one such example where the inability of CNNs to abstract high level concepts is clearly on display.

The method proposed in this paper aims to try and solve a same-different tasking using a more classical approach to vision, as opposed to deep learning. The method is comprised of two distinct modules: feature mapping and prediction. The approach is loosely based on how a human may

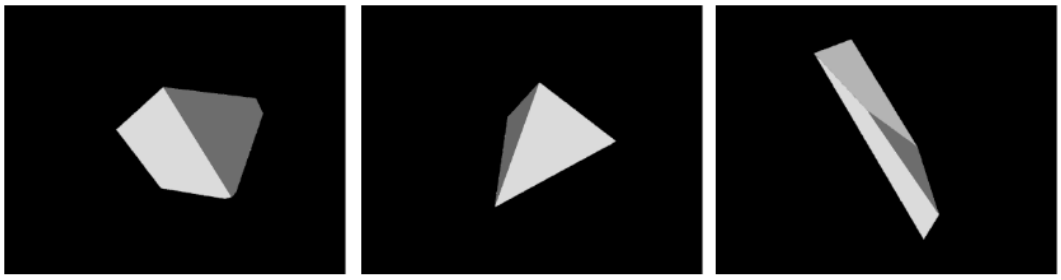

Figure 2. Enhanced view of various polyhedrons

approach solving the problem of determining if two 3D objects were the same or different. First, a person would give each object a quick look around and build up an internal model of each. With an understanding, of both objects, the next step would be to try and find matching features for each which line up. One could try to count all faces, all vertexes etc. however the fastest and most likely option someone may take is to find some views which look the same.

2. Related Work

2.1. Active Vision

One such idea to get around the limitations of two-dimensional images is to use an active vision approach, where the observer is an active agent which can change its position to acquire new information. Aloimonos et al. [2] demonstrate that an active observer can be much more effective at solving basic vision problems than a static observer. Active vision is more than correspondence. The problem solving method involves intelligently selecting viewpoints in order to solve a problem in the most efficient way. The proposed method in this paper achieves this with the predictor module, however the feature mapping module is more akin to simple correspondence.

2.2. Same-Different Solutions

As noted by Ricci et al. [6], relational visual reasoning is something most humans and many mammals are inherently adept at. Yet, many state of the art machine vision models fail at this very task. The problem is a feed forwards networks inability to reason properly across abstract ideas of objects, images etc. Messina et al. [5] found that for many same-different tasks this issue can be remedied by introducing residual connections. On four difficult problems from the SVRT dataset they show that architectures such as ResNet-18 and CorNet-S are able to achieve human level performance, where as older vision networks like AlexNet, and VGG are entirely incapable of doing such tasks.

3. Polyhedral Scene Generator

The polyhedral scene generator (Polyhedral) created by Solbach et al. [7] is an image generator designed for active vision experiments. Polyhedral allows for the generation of 3D scenes containing one or more polyhedrons. One can then specify camera parameters through an API to generate new views from different positions and orientations of the same scene denoted by its scene ID. For this paper, pairs of scenes were generated with each containing one polyhedron centered at the world origin. Each pair either contains the same or different polyhedrons. Even for pairs which are the same, the orientation is changed such that one cannot simply compare views from the same camera view.

4. Methods

In this section, the object feature mapping and view prediction method is introduced. Given a pair of scene IDs from Polyhedral, the goal of this method is to be able to determine if the scenes contain the same instance of a polyhedron. In separate pipelines, the system processes views of each object from various viewpoints in order to build up a spacial understanding of the object in each scene. These mappings are then passed to a predictor module which uses the understanding of one object to make predictions about what it should see in the other scene. The accuracy of these predictions are aggregated in order to make a final determination of same or different.

4.1. Problem Definition

A polyhedron is defined as a three-dimensional shape which has planar polygonal faces, straight edges and sharp vertices. The camera is represented by a position $\vec{x} = (x, y, z)$, and an orientation defined by three vectors $\vec{u}, \vec{r}, -\vec{z}$ denoting respectively: up, right, and forward. The cameras current view at position $\vec{x_i}$ is V_i. A mapping of a polyhedron is denoted as $P(F)$ where F is the set of all unique faces detected. Each face is defined as $f_i = f(P, \hat{n})$, where P is the set of 3D points belonging to that face, and $\hat{n}$ is the face normal vector, pointing away from the origin. Given two polyhedrons A and B, generate

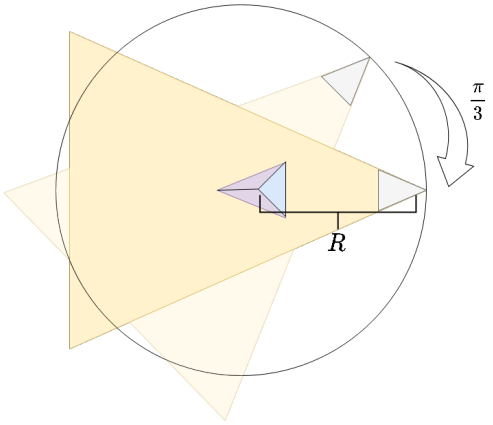

Figure 3. Camera rotation around central object. Each position is spaced $\pi/3$ from the previous one, all at radius R.

$P_A = P(F_A))$ and $P_B = P(F_B))$, the predictor $Pred$ uses these mappings to make predictions and outputs a final result $Pred(P_A, P_B) = y$ where $y \in \{Same, Different\}$.

4.2. Feature Mapping

Understanding of objects comes from combining visual information with positional information to form a coherent model. For each object, views will be processed from five positions around the a viewing sphere centered at the origin at a radius of R, seen in Figure 3. This gives sufficient coverage to have seen every face visible from that azimuth. The ideal is to capture each face in a view at least twice per face using a minimal number of views. A spacing of $\pi/3$ was chosen to accomplish this, giving five unique equiradial positions around the center. An initial view is taken at a default radius and an optimal radius is then estimated based on the size of the object in the view. At each viewpoint processing is done to detect faces, then vertices and finally depth. The following steps are repeated for each of the five views.

4.2.1 Face Detection

Raw views taken from Polyhedral do not have very clear distinction between faces and edges. Shown in Figure 4, views are enhanced by maximizing the differences between the colours of faces. This is achieved by first separating the object from background through simple thresholding, and determining which pixels correspond to the polyhedron in the view. The entire image is squared element-wise which exacerbates the differences in colour. This is further refined by adding a term proportional to the pixel brightness for all

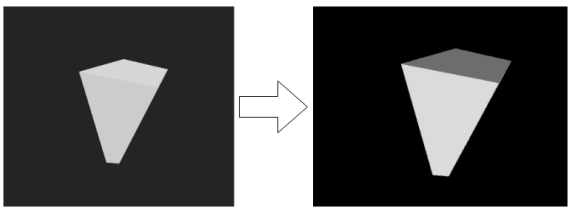

Figure 4. Enhancement of raw view from Polyhedral

pixels belonging to the polyhedron,

$$c'_{i,j,} = c_{i,j,} + c_{i,j,} \cdot \frac{255 - c_{i,j}}{255} \tag{1}$$

Where $c_{i,j,}$ is the colour of the pixel as location i, j, between 0 and 255. This is done twice, and finally the image is squared one final time. This gives all faces the two properties of being all far from the background colour, and all visually distinct from each other. Each face is then segmented by grouping all similar non-zero colour values.

4.2.2 Vertex Detection

For each segmented face in the view, a bounding polygon is found which best approximates the shape. Each point defining the polygons corners is a likely vertex. The criteria for the polygon is that it must have more than three points, and it must approximate the area of the face by at least $\pm 10\%$. Because some vertices may belong to multiple faces, all the detected vertices in the entire view are gathered into clusters where the maximum distance between all points in a given cluster C is 30 pixels. For each cluster C_i the true vertex v_i is found as the weighted least squares centroid of all points $p \in C_i$ is given by:

$$\mu_i^{(c)} = \frac{1}{\sum w_n} \sum_{n=1}^{|C_i|} \frac{p_n}{w_n} \tag{2}$$

Where w_n is the weight of point p_n. For μ as the average position of all points in cluster C_i, w_n is then determined by the formula:

$$w_n = \frac{1}{||p_n - \mu||_2^2} \tag{3}$$

The final result is the detection of all likely vertices within the current view, seen in Figure 5.

4.2.3 Stereo Point Depth

Depth cannot be easily inferred from one image. The stereo depth algorithm from Adi and Widodo [1] is applied in order to accurately determine the positions of detected vertices. The camera is shifted in the $-\vec{r}$ direction by a small amount b determined as a function of the radius,

$$b = 0.05 \cdot (R/2) \tag{4}$$

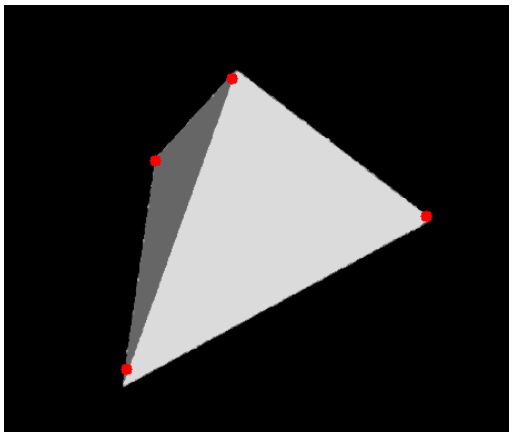

Figure 5. Global vertices detected across entire view

With this new view from a slightly different position the previous steps of face and vertex detection are repeated. Since the direction of motion is known, vertices are between views by seeking the nearest vertex to the right of each point detected in the original view. The horizontal distance between each pair of points (in pixels) is the disparity, d. The depth for each vertex pair v_1, v_2 is then given by,

$$Z = \frac{b \cdot f}{d} \tag{5}$$

Here f is the apparent focus of the camera in pixels. The camera in Polyhedral uses has a 35mm focus, 32mm sensor size and outputs at a resolution of 1920×1080. The apparent focus is then $(35/32) \cdot 1920 = 2100$. From the depth, the coordinates of the real vertex $v = (x, y, z)$ are

$$\begin{aligned} x &= \frac{Z \cdot (1080/2 - v_1.x)}{f} \\ y &= \frac{Z \cdot (1920/2 - v_1.y)}{f} \\ z &= R - Z \end{aligned} \tag{6}$$

Now with the real positions of each point (relative to the camera), the vertices are re-correlated with the faces found in the original view. Figure 6 shows the process of re-correlation, simply based on the vertices overlapping each face. By knowing the camera orientation, the coordinates can be transformed to world coordinates through multiplication with the inverse of the camera transformation matrix. Each face is now defined by a set of vertices in real coordinates.

4.2.4 Face Pruning

Faces are likely to be detected from multiple views. To eliminate redundancy as well as improve the accuracy of

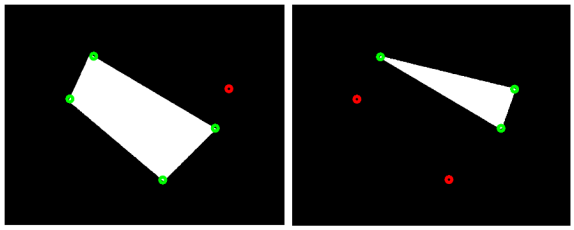

Figure 6. Vertices being re-correlated to faces in the view. Green means the vertex belongs to the face, red means it does not.

the detected features, faces are grouped by having similar area, as well as similar normals. Faces which are determined to be similar are then averaged. Area of a polygon is calculated using the formula,

$$A = \frac{1}{2}(x_1 \cdot y_2 - y_1 \cdot x_2 + x_2 \cdot y_3 - y_2 \cdot x_3 + ...) \tag{7}$$

Where each (x_i, y_i) are the planar coordinates of every point i in the face. Since a polyhedrons face is assumed to be planar, the normal can be calculated using three points A, B and C as,

$$\hat{n} = \frac{AB \times AC}{|AB \times AC|} \tag{8}$$

4.3. Predictions

With a set of mapped features assembled for each object, they are then passed on to the predictor. Simply choosing views at random from each object scene would be a poor way to evaluate sameness visually. Having an understanding of each object allows the predictor to intelligently select views to compare. The predictor first looks through each set

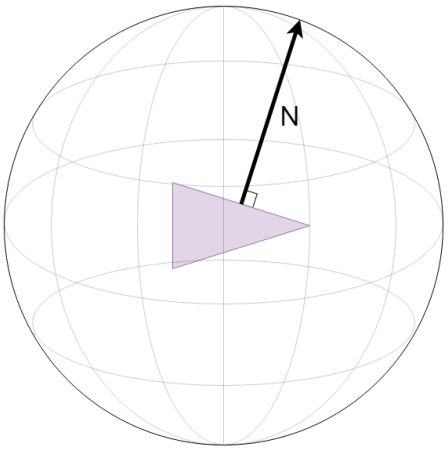

Figure 7. Visual representation of how the camera aligns itself to look head on at a face with normal N by finding where N intersects the viewing sphere

of faces and finds the ones which have a similar area. Ideally more information would be used to select similar faces however, in practice the vertices in each face were not accurate enough to make good comparisons. For a pair of faces f_A, f_B from scenes A and B, the camera in each scene is oriented to where the normal of each face intersects the viewing sphere of radius R shown in Figure 7.

The camera positions itself at that intersection point and looks at the center, giving it a view of that face head on. Consider the view of face f_A as the prediction, and the view of face f_B as the observation. If the objects are the same, it is to be expected that the views should be very similar. For each likely similar face, a bounding polygon is calculated, and its centroid. The centroids for each shape are aligned together at (0,0). An optimization algorithm attempts to align the views together, with the goal of minimizing congruency score. Congruency is defined as the weighted sum,

$$C = \alpha \cdot A_s + \beta \cdot P_s \tag{9}$$

Where the area ratio score, A_s, is the ratio of how much area overlaps. This is easiest seen in Figure 8, where A_s would be calculated as the sum of red and blue areas, over the purple area. P_s is the point distance score, which is the sum of all the shortest distances from a point in f_A to any point in f_B. α and β are the weight coefficients between $(0, 1)$. The idea behind this weighted sum is to reward having a similar size, but also reward alignment. The best (lowest) value is found for each pair of faces. The congruency score for the two objects is given by the average congruency across all predictions. Another metric is simply the ratio of each shape's perimeters denoted by λ. The final output is a 2D point $S = (C, \lambda)$. Experimentally it was determined that the optimal region for predicting sameness was $C < 0.3$ and $\lambda > 0.7$

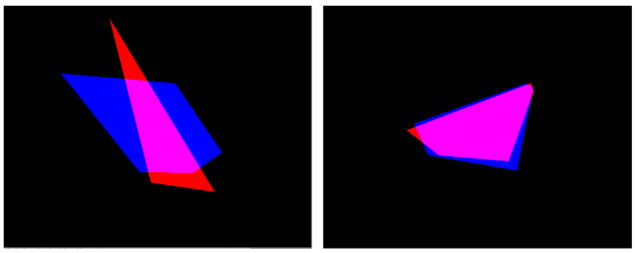

Figure 8. Bad congruency (left); Good congruency (Right). The blue area corresponds to the predicted shape, the red area to the observed shape, and purple is area which overlaps.

4.4. Results

Testing was done on nine pairs of same objects, and nine pairs of different objects (18 total). Figure 9. shows the joint perimeter-ratio congruency scores for each prediction. It is clear that pairs of same objects tend to have predictions with perimeter ratios closer to one, and much lower congruency values (lower is better). Taking the average prediction scores per object pair yields an even more pronounced division between same and different, these results are shown in Figure 10. By thresholding the bottom right region (high perimeter ratio, low congruency), this method can correctly determine sameness 16 out of 18 times. While this data is linearly separable meaning an an accuracy of 100% is possible, in reality a predictor which scores perfect on this data would perform worse on other datasets.

These results appear to indicate that the general approach for solving this problem is sound, however could benefit from refined accuracy in feature mapping. Manually analyzing views predicted from face normals shows that a non-insignificant portion of views were not directly facing a polyhedrons face. Considering the performance in spite of occasional half-baked predictions and that 3D mapping from corresponding 2D images is a well explored problem,

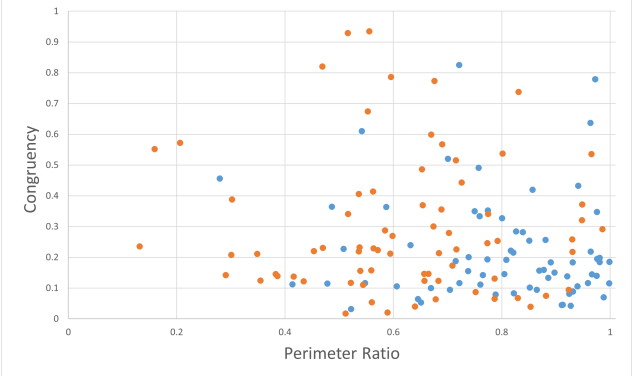

Figure 9. Scores for each view prediction for same pairs (Blue); different pairs (Orange). The further to the bottom right, the stronger the prediction

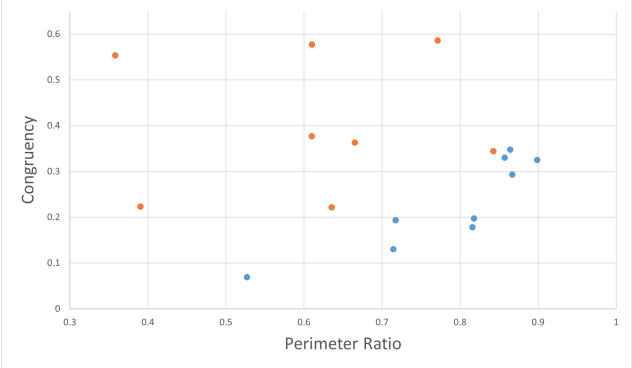

Figure 10. Average prediction scores for same pairs (Blue); different pairs (Orange). The further to the bottom right, the stronger the prediction

it is not difficult to imagine this approach with more accurate sub-algorithms.

5. Conclusion

The method proposed in this paper is to first build understanding a pair of objects, and then use that understanding to make predictions on what it would expect to see if they are indeed the same. By using an active vision approach to prediction, the system is able to utilize minimal views to come to an answer of reasonable accuracy. Problems with the vertex detection and point localization lead to somewhat inaccurate results for features. This affects performance of view prediction, however results are still quite promising. It is believed that improving the accuracy of feature mapping would clearly lead to much higher levels of performance utilizing the same prediction algorithm and comparison metrics.

References

[1] K. Adi and C. Widodo. Distance measurement with a stereo camera. *International Journal of Innovative Research in Advanced Engineering*, 4(11):24–27, 2017.

[2] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *International journal of computer vision*, 1(4):333–356, 1988.

[3] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58, 2011.

[4] J. Kim, M. Ricci, and T. Serre. Not-so-clevr: learning same–different relations strains feedforward neural networks. *Interface focus*, 8(4):20180011, 2018.

[5] N. Messina, G. Amato, F. Carrara, C. Gennaro, and F. Falchi. Solving the same-different task with convolutional neural networks. *Pattern Recognition Letters*, 143:75–80, 2021.

[6] M. Ricci, R. Cadène, and T. Serre. Same-different conceptualization: a machine vision perspective. *Current Opinion in Behavioral Sciences*, 37:47–55, 2021.

[7] M. D. Solbach, S. Voland, J. Edmonds, and J. K. Tsotsos. Random polyhedral scenes: An image generator for active vision system experiments. *arXiv preprint arXiv:1803.10100*, 2018.