3/4/2021

FUNKO POP! RECOGNITION

Dana Gorovic, Ariel Baruch
Final Project, Computer Vision Fall 2020
The Interdisciplinary Center Herzliya. Herzliya, Israel

Abstract

For our final project we decided to use the knowledge, way of thinking and techniques we learned in the course in order to tackle a visual detection problem close to heart in the area of collectibles.
Our project enables visual detection and identification of Funko POP! © collectibles. We find this project scope fascinating because we see real value in bringing this solution to eventually be used by our fellow collector community.

I. Introduction

Funko Pop! Vinyl are highly stylized bobbleheads from Funko based on the Japanese style known as Chibi. Pop! Vinyls cover a range of licensed brands and popular culture such as video games, television, movies, and more. [https://funko.fandom.com/wiki/Pop!]

Funko Pop! (we'll refer to them as 'pops') spread across a variety of different brands, which led to the establishment of a collectors society around them. There are currently **more than 10,000 unique pops** and millions of collectors worldwide.

Figure 1 – example of Funko POP!s

The Funko Pop recognition tool will allow a user to input an image containing one or more pops and get in return structured data specifying the pop(s) product details as well as suggested selling price per pop. This tool is useful for collectors who want to easily and clearly post their pop for sale on any social media or buying and selling platforms (ex: Facebook groups, reddit forums, or eBay).

The scope of this tool is to recognize the front facing side of standard size boxes (4.5x6.25 inch) and return its product aspects. A standard size box has 9 recognizable features that are pictured in figure 2. To extract the defined product data and market pricing we used a dataset of 10,000 tagged data points that include official product image and product details (including information about series, number, name, variant, and suggested price). The pre-processed input image is compared to the dataset images to find the best match from existing matches in the dataset. The output will consist of all the detect POPs in the photo with their exact position, details and suggest target price. More details in the method section

Although there are many unique POPs, the POP boxes share similar structure. Each box contains the following segments:

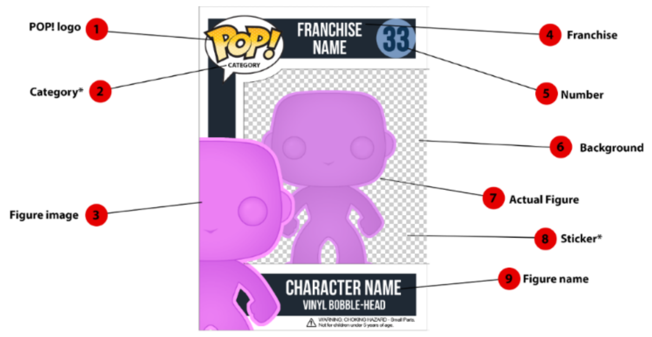


Figure 2- Front box breakdown

1. **POP! Logo:** This is the main POP! Trademark logo. This is always in the top left corner of a pop. For the purpose of identifying the extracted rectangle as a true pop we will validate the existence of this feature.
2. **Category:** The category of the pop.
3. **Figure image:** This is the main stock image of the figure, sometimes deviates from the actual figure in case of a special variant. A series of pre-processing techniques are used to crop this feature on a black background. This figure is used in the final step of matching to the dataset to narrow down from the potential matches.

4. **Franchise:** Name of the franchise / series for the current category (i.e. ‘DC Super Heros’, ‘Spider-man’). Could also be a logo image.
5. **Number**: Identifies the pop within the franchise series, usually between 0-999. Extracted with a series of pre-processing techniques and then OCR is used to return the integer value displayed. This number is used to pull the potential matches from the dataset.
6. **Background:** Box background, usually a solid color, something painted
7. **Actual figure**: The actual pop figure, may vary in size shape etc. We decided to not include this feature in the recognition process because the quality of the image here is low. It will be easier to extract information from the 2D surface than to recognize an obstructed 3D figure, which usually ha a glare when pictures.
8. **Sticker*:** Sometimes special pops come with exclusive stickers, some specifying if the pop is exclusive to a certain store, event, or rate variant. This feature will be ignored but can be valuable to enhance the recognition process in the future.
9. **Figure name:** Name of the figure sometimes with a description **(**i.e. ‘Batman’, ‘Retro Batman’, etc). This feature will be ignored but can be valuable to enhance the recognition process in the future.

II. Methods

The following includes a brief explanation of the main methods that were used:

Hough Transform: This concept is used to extract lines and circles from the given image. It is based on the idea than any point (x,y) in the picture plane is equivalent to a curve (ρ, σ) in the parametric plane. Any pair of co-linear points also correspond to a single point in parametric plane which is the intersection of their 2 curves. In the transform, the points in the picture plane are converted to parametric points and the parametric points having the highest count and above a certain threshold are the ones that represent the lines in the image. [1]

Homography: We will use the concept of homography to get a direct view of the front face of the pop. As we learned in class, two images on the same planar surface in space are related by a homography mapping, also known as an isomorphism (structure preserving mapping) of projective spaces. The homography matrix connect pairs of corresponding points between the two images. In this use case, a perspective transform matrix will transform the input image on a 2D plane (that may not be parallel to the COP plane) onto an image on the 2D plane parallel to the COP plane.

We chose a perspective transform that preserves straight lines but does not preserve angles, distance, or parallelism. Affine transform is a special case of perspective transform that preserves parallelism as well. In all our use cases the results were the same. However, there could be an extreme case where in the input image the front face of the pop does not have 2 sets of parallel lines. In this case we still wanted to extract the rectangle shape and revert back to a parallelogram. Affine transform in this case would not work well.

On a 2D plane the transform is a linear mapping therefore works excellent on the front face of the box, not including the 3D actual image inside the box that has depth. In extreme angles and when the figure is partly obscured by the box in any angle, the transform on the actual image will have bad results and for this reason we decided to ignore the 3D figure.

OCR: Optical character recognition (OCR) was used on some features for text identification. An external OCR tool provided by Microsoft was used so we will not expand on this method. Before running the features in OCR many layers of image pre-processing were performed that will be explained in the Process section below.

[1] Duda, R. O. and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Comm. ACM, Vol. 15*, pp. 11–15 (January, 1972)

SIFT feature matching and ratio test: SIFT selects features in each image that are the locally distinct points, and for each feature describes it based on its local surroundings. In order to match the extracted figure to the potential candidates from the dataset, a brute force matching with SIFT descriptors and ratio test was performed.

We make use of Brute Force Matcher to determine the matches between similar key points in both the images. The matches are evaluated on the basis of Euclidean Distance between the two key points. Euclidean distance of every keypoint in the first image is calculated with every other keypoint in the second image. The good matches are then separated by certain minimum distance criteria, known as the ratio test.

The ratio test[2] is a way to eliminate the false positive results. It looks at the first and second closest neighbor of each feature. If the distance between the feature and its neighbor is significantly less than the distance between the feature and the second closest neighbor than we can add the first closest neighbor match to the final set of corresponding points. The distance ratio proposed by Lowe is 0.8 but we will use a smaller value of 0.6 to get more accurate results.

Filtering by brightest color: Our goal is to extract the figure image keeping the figure image in the foreground and blacking out any background. In order to perform segmentation on the non-geometric foreground we make an assumption that the white outline of the figure image (in grayscale) is the lightest color in the cropped image. This assumption holds for all our use cases. To find the lightest outline we run a histogram that counts how many pixels we have of each color. Due to the noise, we will the first bin of the histogram looking from right to left that contains above 30,000 pixels. This bin with the large count will be indicative of the white outline. From there we are able to filter out the image based on this color threshold.

Filtering by most common color: In this use case we want to extract the number feature and pre-process it before sending it to OCR. Each number feature is displayed in a colored circle in the top right corner of the pop face. A circular mask displaying only the number inside the circle and a black background is output. Then in order to find the color of the number which is the most common color in the region we focus in on a smaller radius where a histogram analysis of the colors per pixel are performed. The largest bin signifies the color of the numbers and the color (+/- 50) that the rest of the crop will be filtered for.

Find contours: Retrieves contours from a binary image using the algorithm of Suzuki & Be. [3]

III. Recognition Pipeline

We've designed the recognition pipeline to consist of the following steps:

1. **Funko box detection:**

This task requires finding the corners of each Funko box in the given photo. For simplicity we will discuss the process for a single box corner detection.

First thing we did is to find straight lines in photo. To do so we used Hough transformation (discussed in methods) to find lines. As a preprocessing step we've turn the image into grayscale, blurred the image using a Bilateral Filter[4] and then used an Adaptive Binary Threshold[5]. The goal for preprocessing is to strength and distinguish the lines in the photo. The way Hough transform works is to look at all possible lines and count how many points in the image are a part of each line – this will be the line's weight in the Hough space. We

[2] David G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints" (2004)

[3] Suzuki, S. and K. Abe. "Topological structural analysis of digitized binary images by border following." *Comput. Vis. Graph. Image Process.* 30 (1985): 32-46.

[4] Tomasi, C; Manduchi, R (1998). *Bilateral filtering for gray and color images*(PDF). Sixth International Conference on Computer Vision. Bombay. pp. 839–846. doi:10.1109/ICCV.1998.710815

[5] https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html

restricted the angles to nearly horizontal and vertical lines. To illustrate (Figure 3):

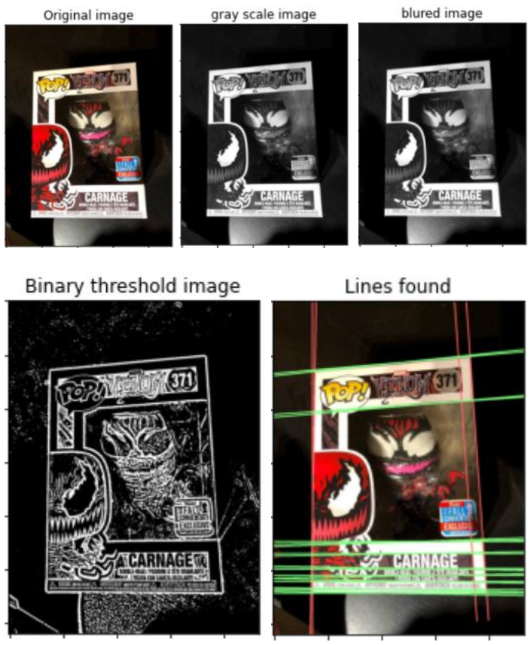


Figure 3 – Illustration of detecting lines

Note – we also explored using edge detection (canny) and Harris corner detection which gave less accurate results. After we have the lines we calculate intersection between the vertical and horizontal lines we found. These intersecting points will be used to determine possible box rectangle candidates.

Considering all the intersection points we found – there could be a huge amount of rectangles to consider, up to $\binom{N}{4}$ combinations. Here we applied heuristics to filter out candidates and only consider real possible box candidates. We got to an average of < 100 candidates using the following constraints:

- Natural order between corner points – top left is left of the top right corner etc.
- Consider horizontal/vertical points that shared a detected line - i.e. top left was on the same detected line as top right.
- Proportion between the width of the top edge and the bottom edge
- Proportion between the height of the right edge and left edge
- Box proportions – we know what to expect scale wise.
- Angles of lines and inner angles or corners.
- Surrounding patch colors – we should have a bright color near the corners.
- Filter 'close' points by distance – No need to take more than 1 point for a very close distance.

Once we have the box candidates, we consider each candidate by transforming using homograph (explained later) and performing a verification to check if it is in high confidence a POP box.

In order to support multiple boxes in one photo we've added a step to cluster similar location candidates into buckets, and find the best match for each bucket.

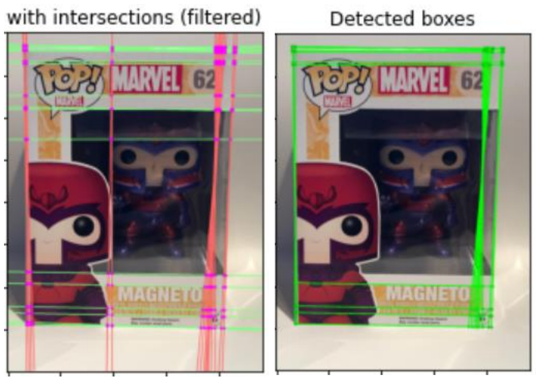


Figure 4 – Detected points and rectangles

To identify whether a rectified image is a POP box we look at the common feature of the POP! Yellow balloon logo on the top left corner. We transform both the candidate image and a POP! balloon patch template (figure 5) to HUE and strip out all colors except for the yellow color range. The reason

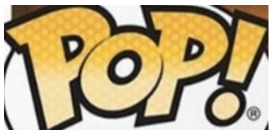


Figure 5 – Pop! Patch template

for the HUE transformation is that with RGB we don't have easy control about color ranges. Then apply a template matching to find a match of the

POP! In the image. The score is calculated using the equation below where I is the segmented box and T is the pop patch template from figure 5. If we get a high score in the top left position we decide it is in fact a POP.

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

```
cv2.TM_CCORR_NORMED max_val: 0.91777503490448
```

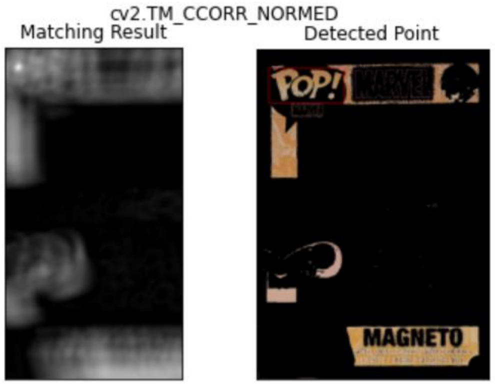
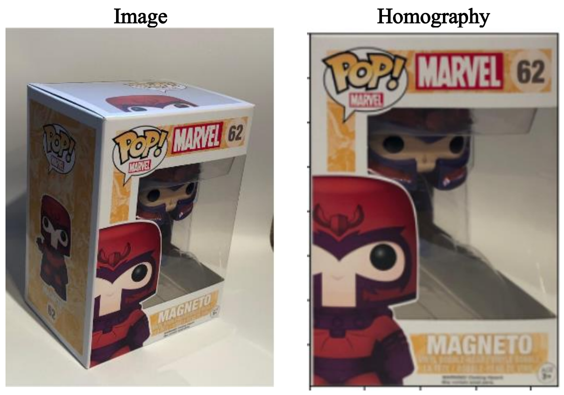


Figure 6 – POP! Match detection

For each detected POP box the pipeline continues with the following steps :

2. **Homography:**

As shown in figure 7 below the homography on the box surface performs really well even in high angles. Even after the transformation, the main features like number and figure image that will later be extracted will not ne negatively impacted by this transform. Rather, it will make it easier to process and recognize these features after the transform.



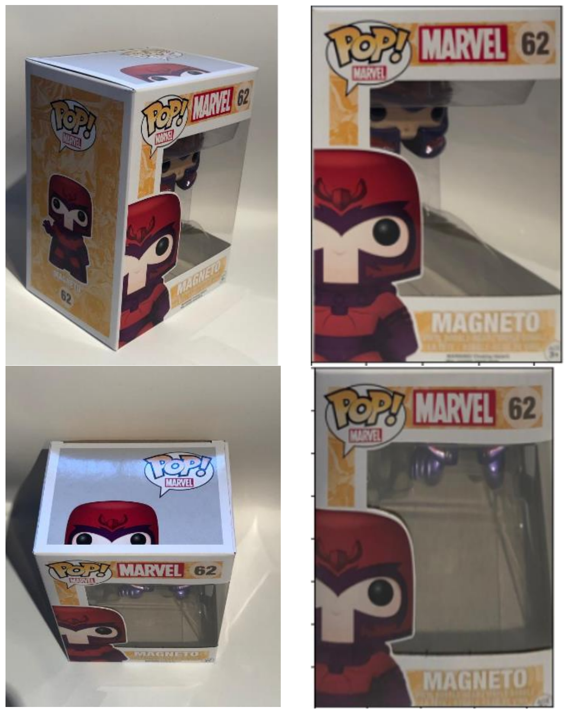


Figure 7 – Homography edge cases

3. **Segmentation**

Once we have a rectified front view scaled image of the box the rough segmentation is pretty easy since we can just segment to pre-defined areas. For this part we hard-coded our desired segments locations:

a. Top right corner balloon (series number)
b. Top left corner pop category
c. Top middle – Series name
d. Left image of the pop
e. Bottom middle – pop name

4. **Number Identification**

For this purpose we first performed pre-processing on the number segment to clean it up and make the numbers more clear. Then we took the clean version and ran the OCR to retrieve the numeric digits.

For the pre-processing we first transformed the image to grayscale, then detected the circle containing the digits using Circle Hough

Transform[6]. Once we have the circle, we look at a central small patch and used a histogram method to find the dominant color. This is assumed to be the digits color. The radius of the central patch was set after determining the optimal radius that would always include the digits color as the dominant color. Using this color we separate the digits using a binary threshold and then trim the image around the digits.

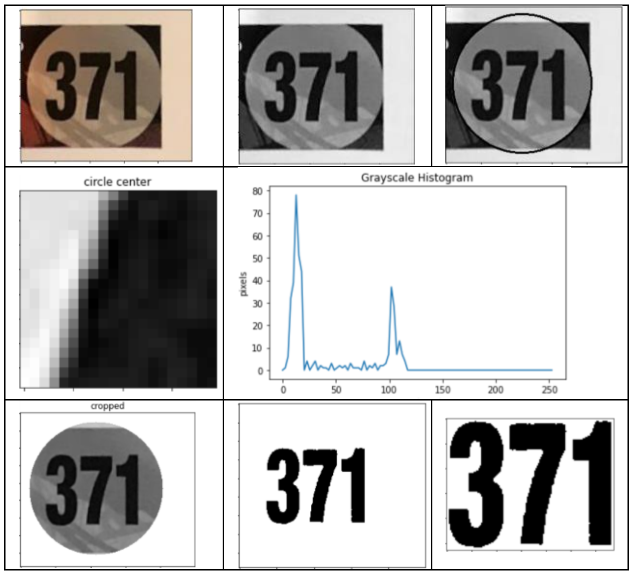


Figure 8 – Number pre-processing

We've also tested this method on box numbers with 2, 3 and 4 digits and got good results for all.

5. **Figure Segmentation**

The goal of this step is to 'crop' out the POP illustrate figure on the bottom left side of the box. In order to achieve this goal we first noticed that the figures are all outlines with a thick white line surrounding the figure's shape.

To leverage this fact we first draw a histogram of the colors in the segment and looked for the brightest color with high volume (this will be the color of the white outline). Then we threshold the image by this color range. Next we applied canny edge detection[7] to extract the lines edges and then dilation to thicken and 'close' down the shapes to create a clear separation between the figure and the background. From this point we were able to distinguish between connected components (the largest one to be the figure itself), and to fill the holes using contouring and polyfill[8]. Last step was to use the calculated mask to extract the figure.

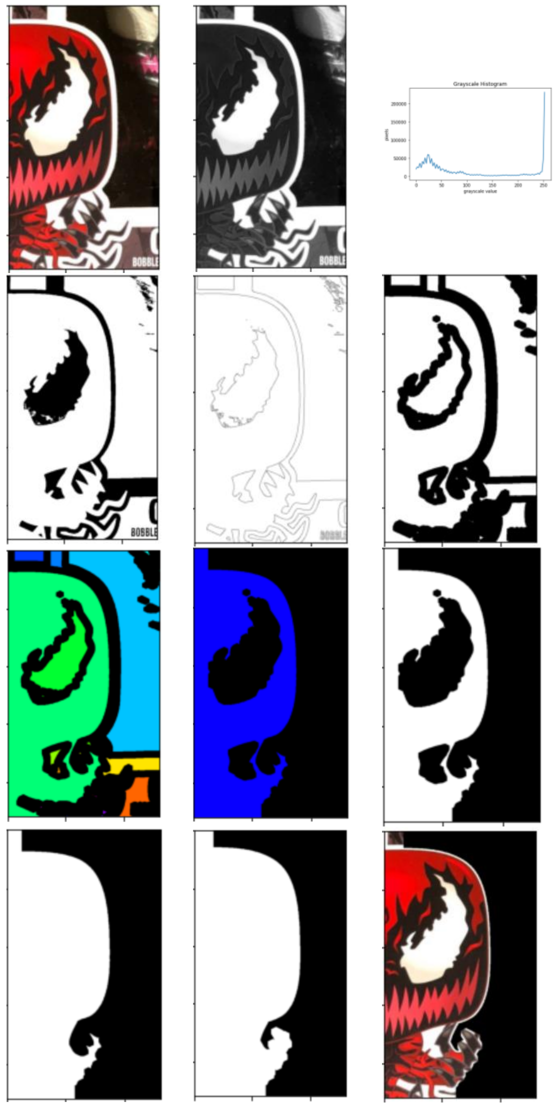


Figure 9 – Figure extraction

[6] J. Illingworth and J. Kittler, "The Adaptive Hough Transform," PAMI-9 , Issue: 5, 1987, pp 690-698

[7] Canny, J., *A Computational Approach To Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

[8] Open CV find contours implementation based on Suzuki, S., and Be, K. (1985). Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing 30, 32–46. doi:10.1016/0734-189X(85)90016-7.

6. **Find Candidates**

This process leverages our database. It takes the identified number and retrieves all the potential candidates, which are the data points with the same series number. From our investigation of the dataset we saw that there is a good distribution among numbers, meaning on average we will have ~20 candidates with a max of 70 candidates to examine as potential matches, as (shown below in figure 10). Using this step as the first step in candidate retrieval limits the potential candidates from the full dataset (10K) to about 20 matches.

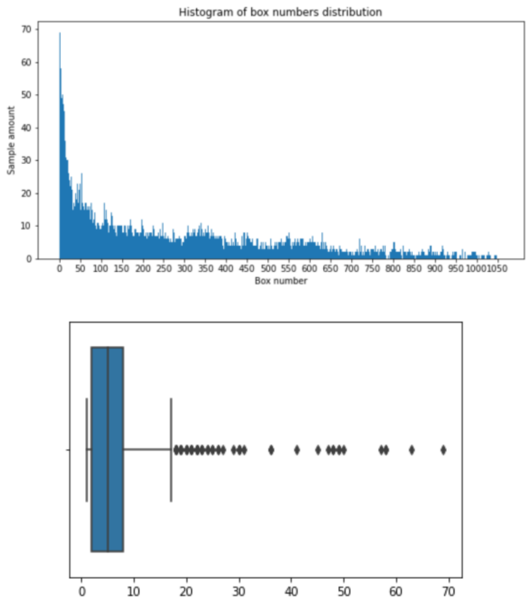


Figure 10 – Box number distribution

7. **Find Best Match**

Now that we have the possible candidates, in order to determine which one best represents the POP – we will use the pop figure extracted in step 5 and match it to all potential candidates. For this we use SIFT feature matching detailed in the methods. All the potential matches are matched with the POP. The best match is defined as the candidate with the highest number of matches to the POP after the ratio test is performed to drop less accurate matches.

In figure 11 we can see an example of the extracted figure being matched to a stock image from our dataset. We can see it matches both to the box figure and also for the figure itself.

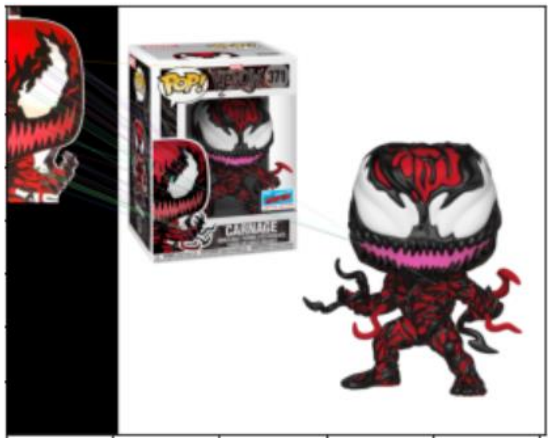
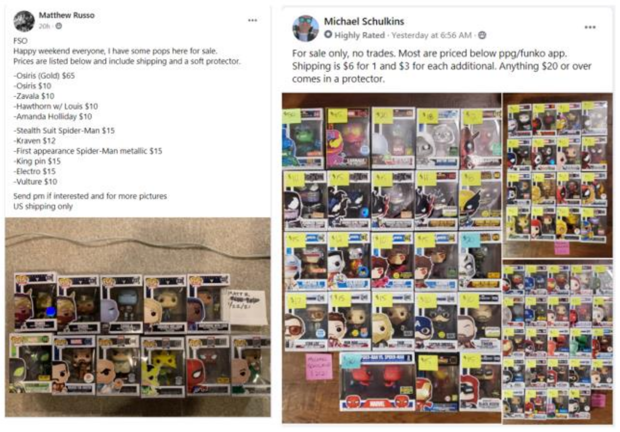


Figure 11 – Feature matching

IV. Conclusion

In conclusion, we showed in this report how we can use several advanced computer vision techniques to segment and identify a POP using a single input image. All but one (OCR) of these techniques did not require machine learning techniques. It only requires the input of a single image where the front face of the box is reasonably showing. This makes it a powerful tool for a user who is trying to easily photograph and upload details of his POP to an online marketplace.



Matthew Russo
20h

FSO
Happy weekend everyone, I have some pops here for sale.
Prices are listed below and include shipping and a soft protector.

-Osiris (Gold) $65
-Osiris $10
-Zavala $10
-Hawthorn w/ Louis $10
-Amanda Holliday $10

-Stealth Suit Spider-Man $15
-Kraven $12
-First appearance Spider-Man metallic $15
-King pin $15
-Electro $15
-Vulture $10

Send pm if interested and for more pictures
US shipping only

Michael Schulkins
Highly Rated · Yesterday at 6:56 AM

For sale only, no trades. Most are priced below ppg/funko app. Shipping is $6 for 1 and $3 for each additional. Anything $20 or over comes in a protector.

Figure 12 – Facebook listing example

This can also be a useful tool for a buyer who is interested in finding the market price and full product details of a listing she is interested in buying. For example, in figure 12 above it is hard to evaluate the details in these crowded listings. By running the image in our tool, we could ideally recognize each box and return more product details than what is even featured by the seller.

V. Future Work

One improvement in the future for the efficiency and accuracy of the tool would be to add more layers to the candidate retrieval process before running the matching on the potential candidates. OCR on the extracted figure name and franchise name could be run and the results compared to the potential candidates already retrieved from the number matching.

Another modification that can be made is running the box segmentation with a machine learning algorithm. In commercial use the runtime will be a valuable factor and machine learning techniques may be able to return faster results.

We would also want to extend the tool to be able to recognize POP variations as identified by their sticker type. Sometimes a sticker is the only differentiation between 2 different products so we will need to recognize the sticker specifically.

Lastly, we would want to extend the tool to be able to differentiate between real and fake POPs. This will require more accurate modeling of the box and more descriptive techniques.