Synthesis Project: Digital Image Enhancement Using a Vehicular Image Capturing Platform

Hamed Soroush, Rui Wang, Brian Levine, Mark Corner
{hamed,ruiwang,brian,mcorner}@cs.umass.edu

Abstract

In this work, we define the quality transfer problem and present an approach to tackle it: Given a low-quality image taken from a mobile phone, how could you enhance it in a visually plausible way to include additional details. Our approach is based on creating a temporal, gps-tagged high-quality image data-set by building a vehicular image capturing platform to autonomously capture high-quality images from different areas of the city. For a given low-quality input image, close images from the data-set are selected which would be used for enhancement in a combined process of salient feature extraction and patch matching. We present the results of applying this technique to samples of images taken in downtown Amherst.

1 Introduction

- "Let's magnify it and see if we can get a reflection of her eye."
- "OK. Magnification times 100 for starters."
- "What's that there, looks like a guy in a T-Shirt."
- "Resolution isn't very good" [assistant zooms in again, picture of a guy holding a basketball appears in the reflection of the eye of the woman in the original image]
- "Yes, it is."

The above mentioned dialogue from an episode of CSI [3] is not unique in movies and TV series. Many similar scenes exists in which an apparently low-quality image is "enhanced" through the use of advanced "algorithms" and hi-tech software (see [13] for a collection of these exciting moments). The problem, needless to say, is that these enhancement programs, regardless of the degree of sophistication of the algorithms used in them, do not -and cannot- exist in real-world. No program can digitally enhance a single image as not enough information exist in a single image for that purpose.

Given higher quality images of the same scene, however, it is possible to enhance *some parts* of a low-quality image. The high-quality image(s) could have been captured using an advanced camera and/or at a higher resolution; but possibly in a different lighting condition. We refer to this specific problem as the *quality transfer problem* and in this work, present our approach to tackle it.

The basic steps of our approach could be summarized as follows:

1. Find the closest high-quality image(s) to the input low-quality image by searching in a previously created data-set of high-quality images. We refer to this set as H.
2. For each image h in set H, find which segments of it are best to be used in enhancing the low-quality image. This is to filter out the relevant regions of h which are blocked by obstacles (e.g. people, cars, etc) to avoid unnecessary computations in later steps.
3. Detect the salient features of each appropriate high-quality segment S_h.
4. Define a patch of pixels around the detected salient feature points and then look for corresponding patches in the low-quality image.
5. For each high-score match, scale the colors of the high-quality patch to resemble that of the matched low-quality patch and then copy it over to the low-quality image replacing the matched patch.

We describe each of the above steps in more details in the following sections.

2 Creating A High-Quality Image Data-Set

The collection of high-quality images which are to be used to improve the input low-quality images has a significant impact on the performance of our approach. The more images there are, the higher the chances of successful enhancements.

For our purposes, a data-set of high-quality images from regular scenes of the city is needed. It is best to have the images geo-tagged so that given a low-quality geo-tagged input image, a corresponding high-quality match in the data-set could easily be found.

One approach to create the data-set is to download photos from image sharing websites such as flickr [5]. However, the data-set created in this way might not be as comprehensive due to several reasons: i) Many photos on such websites are taken from popular scenes only (e.g. Eiffel Tower), ii) A significant portion of many of these images are of faces, pets, and objects which does not make them suitable candidates to enhance images of *ordinary scenes* taken in a city, iii) many of them are not geo-tagged.

Consequently, we have attempted to augment such a data-set by new additional images. Particularly, we have installed cameras on three vehicles, one passenger car and two buses, which move on different routes in our city to continuously capture geo-tagged images. This is aligned with the fact that most of the low-quality images we were interested to enhance, were captured in our city. We describe our image capture platform in more detail in the following.

2.1 Image Capture Platform

Creating an image capture platform is more challenging than what it might seem at first. Since the number of required images is huge, it is best to automate the process as much as possible. Furthermore, diagnostic mechanisms have to be present to make sure that system faults are detected and resolved in a timely fashion. In the following, we describe different components of our image capture platform and point out the ways in which we dealt with challenges in construction, monitoring and deployment of it.

2.1.1 Hardware Components:

We have used three vehicles, a passenger car and two buses moving in different areas of our city to capture photos. Our vehicles are equipped with Hacom OpenBrick 1GHz Intel Celeron M systems (referred to as *bricks*), a GPS receiver, an 802.11abg mini PCI card and, Wireless 3G USB modems. Two Prosilica GC1020 cameras have been used in our platform as well as one Canon G9.

GC1020s are 1024x768 resolution CCD cameras with Gigabit Ethernet Interface (GigE Vision) and incorporate Sony ICX204AL and ICX204AK CCD sensors. What has primarily made this type of camera suitable for us is its relatively small size making it easier to be installed on our vehicles. Furthermore, the ability of these cameras to run at 30 frames per second at full resolution coupled with their providing of an ethernet interface allows for implementation of a quick capture-and-transfer scheme.

While the small size of the Prosilica cameras introduces fewer deployment challenges, the quality of the images taken by them is limited due to their small sensor size. Consequently, we have added the Canon G9 Point-and-Shoot Camera to our platform to take images of higher quality. Addition of this camera to our system, however, introduced new challenges in image capturing and transfer as well as deployment difficulties.

While making sure that the cameras are protected, we had to ensure that our system causes minimum disturbance to both vehicle and vehicle riders. This restricted our options on mounting the cameras and doing the wiring of the equipment. Figure 1 shows one of our deployed cameras on a bus.

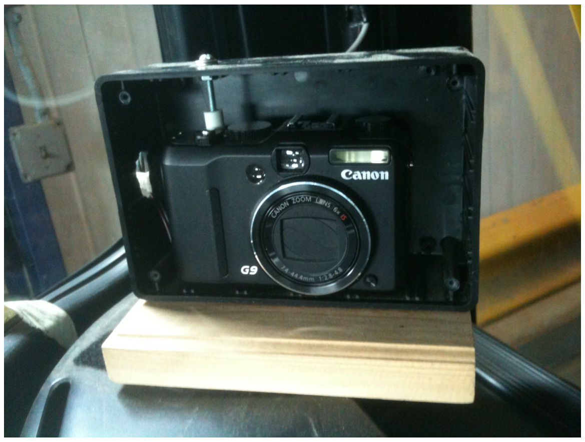


Figure 1: Canon G9 camera mounted inside a bus. The lid of the protecting box is taken out in this picture. The power button of the camera is always pressed down. When the vehicle starts, the bricks turns on instrumenting the camera to start the capturing process.

Software Components: The GC1020 cameras come with a SDK allowing the *bricks* to control the operation of the camera through an ethernet interface. Employing a JNI wrapper of the SDK, we have created a Java program that instruments the camera to capture images in appropriate times of day (excluding nights) and transfer the images over the ethernet port to the *bricks* where they would be geo-tagged using the current reading of the GPS device.

Capturing and transferring images using the G9 camera, however, had to be done in a different way as it does not come with an SDK. While a suit of applications exist to instrument the camera through the USB port (e.g. gphoto2 [6]), we found the delay associated with them is large and does not allow for truly continuous image capture. Consequently, we modified the firmware of the camera to provide support for continuous image capture/transfer from the camera. Particularly, we modified Canon Hacking Development Kit (a.k.a CHDK [2], an open source alternative firmware for many of Canon Powershot cameras) and created an on-camera script to capture images at appropriate times of day. The images would then be transfered in batches to the computer on the vehicle and later be synced with a GPS trace of where the vehicle has been based on the capture time.

While it was possible to transfer captured images to a server over the internet using the 3G modems, we found that impractical due to the large amount of data to be transfered and a bandwidth cap over the 3G link. Therefore, the images were captured into an external hard-drive on the vehicle which was then manually transfered to our lab in order to add the images to our data-set. Due to the large size of the hard drive (300GB), we did not have to perform this manual task more than twice a month.

In addition, we did setup reverse ssh sessions from a computer in our lab to the vehicles for diagnostics and monitoring purposes.

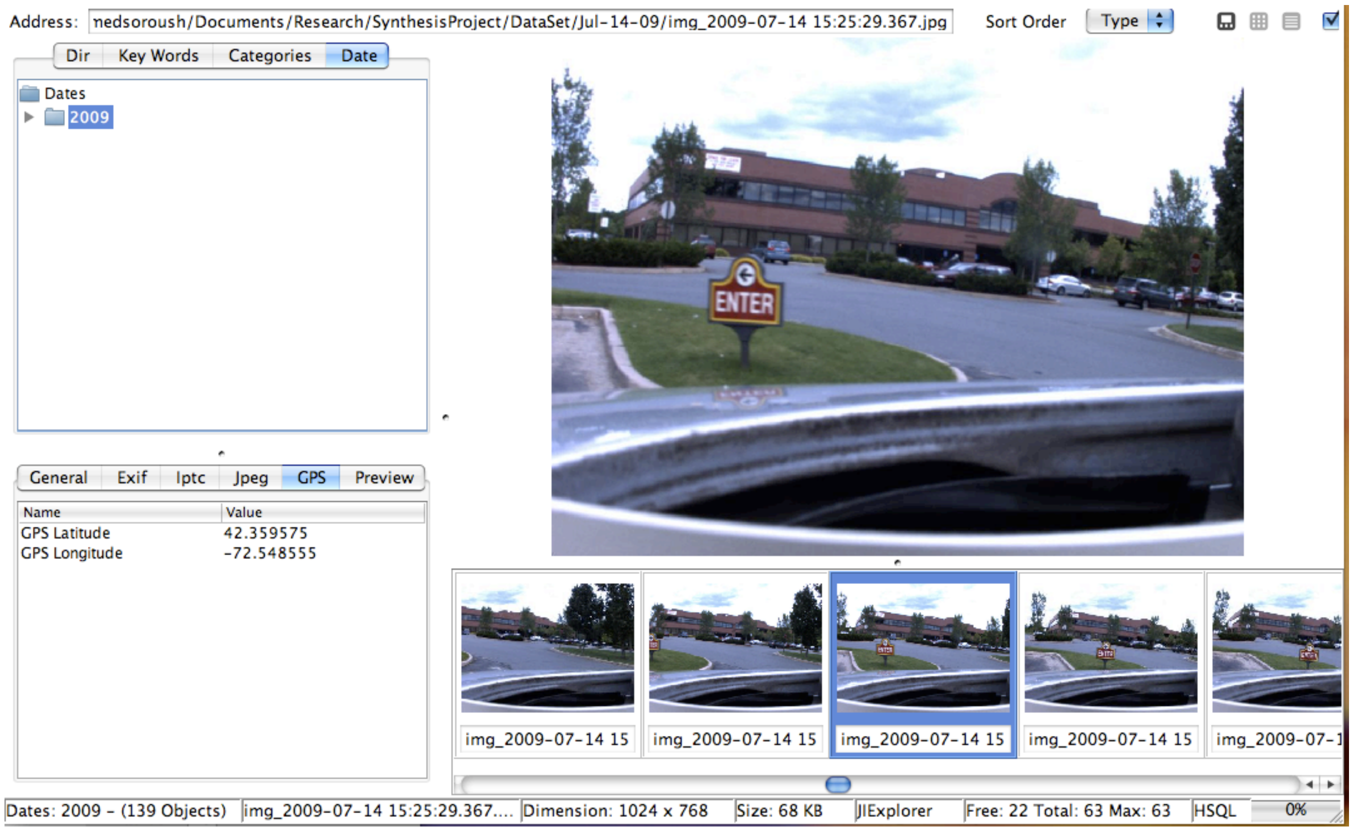
2.2 High-Quality Image Selection

Given an input low-quality image, we need a way to find similar high-quality images in the data-set. Our assumption is that the input image is geo-tagged, that is, the location of where it was captured is stored in the image metadata. Consequently, the corresponding high-quality images could be found efficiently, given that our data-set consists of geo-tagged images too.

Selection of high-quality images is done in a semi-automatic way. We have created an interface to our image-dataset to assist a user in selecting relevant image to an specific input image. Given the location of an input image and a *proximit range R*, the interface, shows thumbnails of those high-quality images of the data-set which have been taken in distance R of the input image. This allows the user to select the images which he thinks can be used in the enhancement process easily and provides for filtering out those high-quality images which are not useful due to the existence of obstacles. A snapshot of the Image selection interface, which is based on an open source image viewer called JIExplorer [9] is shown in Figure 2.2.

2.3 Detecting Salient Features

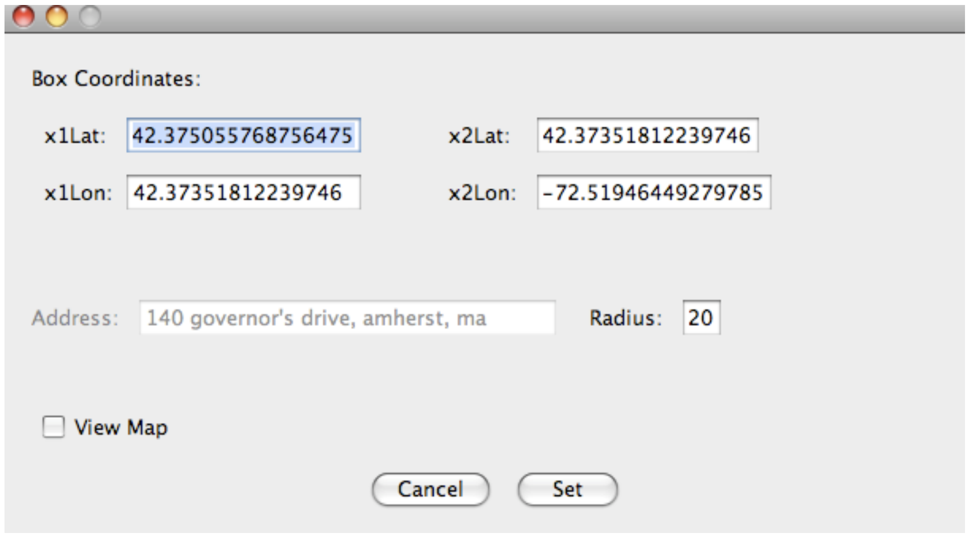
Once a candidate high-quality image is found to be used to enhance a given low-quality image, the next problem to solve is to detect the most visually interesting areas of the image to enhance. This is because not all of the parts of the high-quality image are



Name	Value
GPS Latitude	42.359575
GPS Longitude	-72.548555

Dates: 2009 - (139 Objects) | img_2009-07-14 15:25:29.367.... | Dimension: 1024 x 768 | Size: 68 KB | JIExplorer | Free: 22 Total: 63 Max: 63 | HSQL | 0%

(a)



Box Coordinates:

x1Lat: 42.375055768756475   x2Lat: 42.37351812239746

x1Lon: 42.37351812239746   x2Lon: -72.51946449279785

Address: 140 governor's drive, amherst, ma   Radius: 20

☐ View Map

(b)

Figure 2: Image selection interface allows the user to select images from the collected dataset based on the location or the time they were captured.

suitable to be used in enhancement process due to differences in camera angles and possible presence of objects which are absent in the input image.

To tackle this problem, we have attempted to detect the salient features in the high-quality image and then only consider patches around those feature points in the enhancement process. We have tried three different types of feature points for this purpose: *i*) SIFT features, *ii*) image corners (Rosten's machine-learning based corner detection algorithm [15, 16]), and *iii*) Loy and Zelinsky [12] salient feature points.

Our most interesting results where achieved using Loy-Zelinsky salient feature point detection which utilizes local radial symmetry to identify regions of interest within a scene. In addition to its reasonable performance, their technique requires small amount of computations and therefore runs fast.

2.4 Matching Patches

After detecting salient feature points of the image and defining patches around them, next step is to match each of them to their corresponding patch in the low-quality image. We tried both SSD-based and Normalized Cross Correlation (NCC)-based techniques and as we will show in Section 3 the latter seemed to produce better results as the matchings are independent of illumination and dependent on texture only. The implementation of our matching techniques is done using FFT based correlation. Only matches with a matching score higher than a fixed threshold are considered as valid matches which would later be color-scaled and copied over to the matched location in the input low-quality image.

2.5 Color Scaling

Once the matchings are found, we apply a uniform color scaling to each high-quality patch which is to replace another low-quality one. For each color channel, we calculate the average color of the source and target patches and scale the color of the source by the ratio of the two.

3 Results

We present the result of running our approach in the context of a "zoom and enhance" application that we developed. The user is provided with an interface allowing it to zoom and select part of a low-quality image. The enhanced version of the selected part, which has been created using our technique, would then be presented to the user. Figure 3(a) shows a low-quality image captured using a mobile phone. As a point of reference, we have presented a close high-quality image from the data-set in Figure 3(b).

Suppose that the user selects the part of the image shown in Figure 4(a). As described earlier, our system detects salient features of the corresponding segment in the matched high-quality image and attempts to match patches defined around them to locations in the low-quality image. Salient features of the corresponding high-quality segment is shown in Figure 4(b).

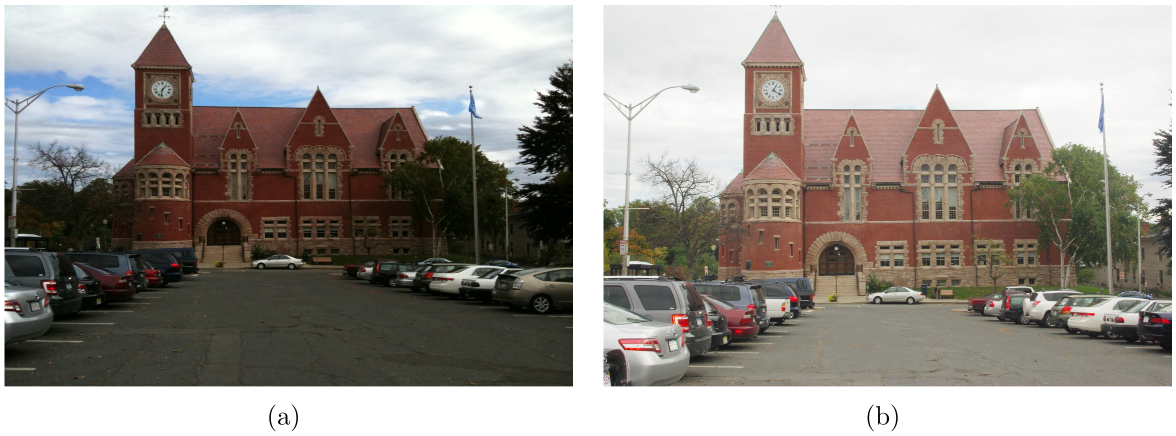
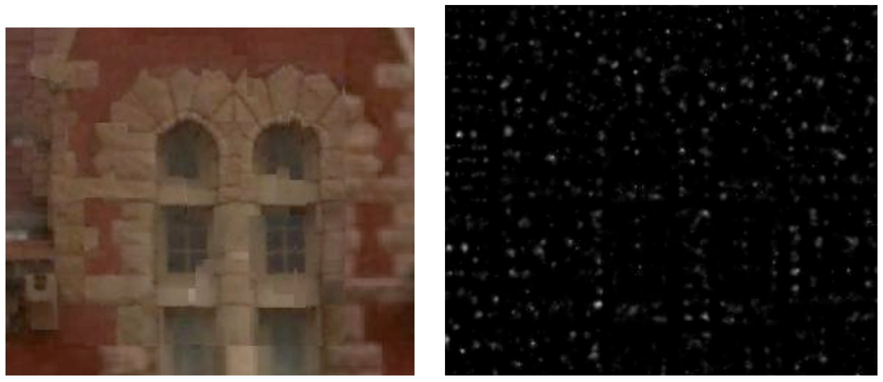


Figure 3: An input low-quality image (a) taken from a mobile phone and its corresponding high-quality image from our data-set (b). The images are not shown in their actual sizes.



(a) Zoomed-in area of the low-quality Image.

(b) Salient features of the corresponding high-quality segment to low-quality image (a)

Figure 4: Selected zoomed-in area of the low-quality image in (a) and the detected salient features of the corresponding segment in the high-quality image in (b).

As far as matching high-quality patches to corresponding locations in the low-quality segment is concerned, two techniques is possible, SSD-based or NCC-based error minimizations. We have found that the NCC-based technique works better for the images of our concern as it is independent of illumination and only dependent on texture. Figure 5 shows the matching of a sample high-quality patch around a salient feature (Figure 5 (a)) to its proper location on the input low-quality image (Figure 5 (b)). Figures 5 (c) shows the sum of squared differences (SSD) of the patch and low-quality photo as an image, with the SSD sign reversed and its value normalized to range [0 1]. The normalized cross correlation image with values ranging from 0 to 1 is depicted in Figure 5 d. As it can be observed, NCC image is more successful at finding the location of the patch. Our system can be tuned to use any of the two techniques depending on the type of the input image and the corresponding high-quality images

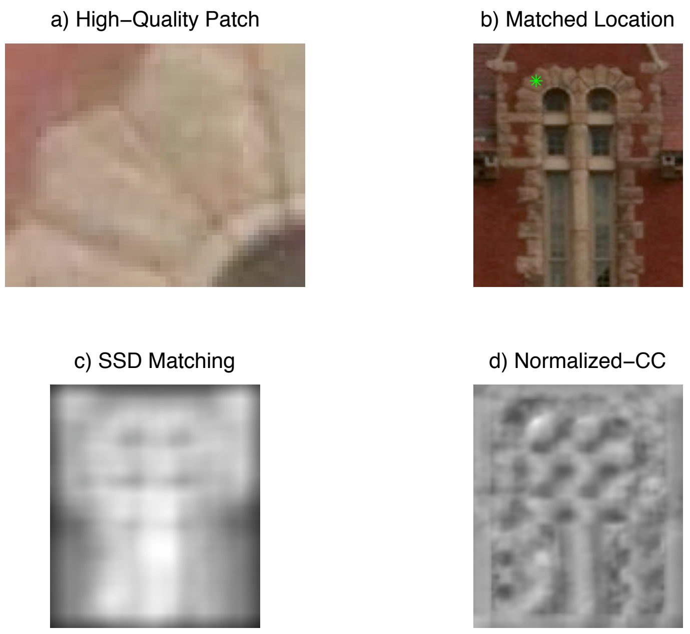


Figure 5: A high-quality patch, depicted in (a) is matched to its proper location in a low-quality segment using a NCC-based technique (b). Figures (c) and (d) depict the effectiveness of each of the SSD-based and NCC-based techniques in matching the patch. Brighter values in the NCC image show higher correlation and darker values in the SSD image show smaller SSD values. NCC-based technique seems more successful as its approach does depend on illumination. Our system can be tuned to use any of the two techniques depending on the type of the input image and the corresponding high-quality images in the data-set.

in the data-set.

The enhanced image is depicted in Figure 6.

Another area of the low-quality image is depicted in Figure 7(a), its salient features in Figure 7(b) and the enhanced image in 8.

Running our image enhancement scheme on an entire low-quality image takes a long time, currently making it impractical in real-time. The bottleneck seems to be the matching technique and we are currently looking into optimizing this process by using tree structures [21, 8, 10, 11, 20].

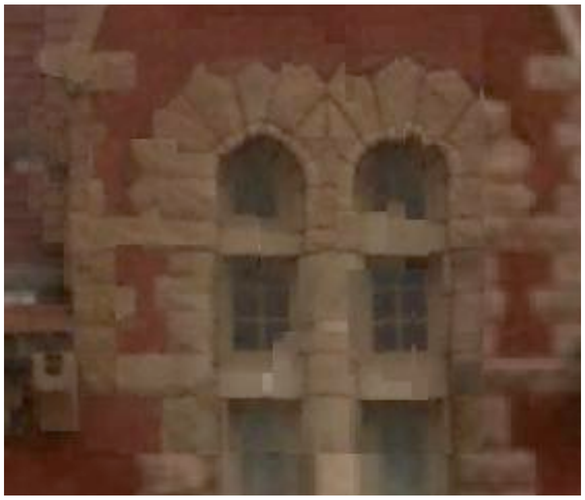
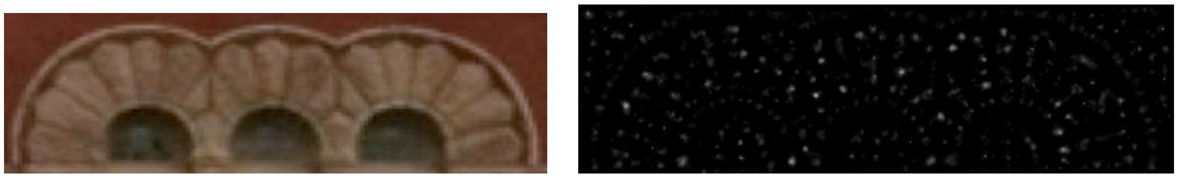


Figure 6: Enhanced version of Figure 4(a)



(a) Another zoomed area of the low-quality Image. (b) Salient features of the low-quality segment in (a).

Figure 7: A zoomed-in region of the low-quality image 3(a) and the salient features of the corresponding high-quality image in our data-set.

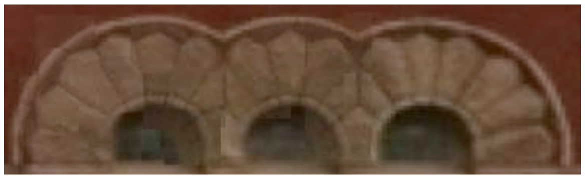


Figure 8: Enhanced version of Figure 4(a)

4 Related Work

In this section we categorize and briefly go over some of the related work in literature.

4.1 Image Collection Techniques

Creating large image data-sets has been an important corner stone for many computer vision and graphics research projects. Many approaches such as [7, 18] download photos from the Internet using image search engines and/or popular photo sharing websites such as flickr [5]. While the data-set collected in this way could also be used in our system, we believe that by itself, it cannot be the as comprehensive as we need due to several reasons: i) Many photos on such websites are taken from popular scenes only (e.g. Eiffel Tower), ii) A significant portion of many of these images are of faces, pets, and objects which does not make them suitable candidates to enhance images of *ordinary scenes* taken in a city, iii) many of them are not geo-tagged. The images that we have been collecting using the buses fills in this gap to a great extent providing us with a more useful data-set to enhance low-quality images taken in our city. Furthermore, constant image capturing on the buses, also captures the *temporal change* in the looks of the city providing for filtering out images in both time and space dimensions and consequently, finer enhancements. For example, if we were to enhance an input image taken in winter, we can only consider the high-quality pictures of the same scene which are taken in winter.

A promising image collection approach has been proposed in PhotoCity [19]. In this approach, photo collecting process has been modeled as the intersection of two types of games: ARG (alternate reality games) and GWAP (games with a purpose). In this game, teams of people attempt to capture pictures of various buildings in a geographic area (e.g. campus) which would be used in creating dense 3D models of that area. We believe that similar crowd sourcing approaches to image collection could provide a more comprehensive coverage of various areas of the city and therefore, be a great addition to our current data-set.

4.2 Texture synthesis and Image Completion

Texture synthesis and image completion, form a body of work which is, to some extent, related to our approach for image enhancement. While being a type of image synthesis, the major difference between our approach and many of these techniques is that our goal is to enhance a low-quality image using high-quality images of mostly the same object. The synthesized image, in our case, points at a real object in the real world.

Patch-based techniques -similar to the one in our work- have been used for texture synthesis before. The texture synthesis method proposed in [4] samples patches from a texture example and pastes them in the synthesized image. This approach outperformed previously proposed model-based methods to solve texture synthesis problems. In [7] an algorithm is proposed that patches up holes in images by finding similar image regions in a database. The database is constructed from the images downloaded from the web. The presented technique uses gist scene descriptor [17, 14] to find images of similar scenes to a given input image (with a hole). The approach attempts to fill in the holes in a both seamless and semantically valid manner. However, the synthesized scene may not actually refer to a real existing one.

4.3 Nearest Neighbor Search Methods

Various techniques have been proposed in literature to speed up the search process (we use a similar process when matching patches). These schemes generally involve tree structures such as kd-trees [21, 8, 10], vp-trees [11], and TSVQ [20]. All of these approaches support both exact and approximate search. In [1] a randomized correspondence algorithm has been proposed which aims at quickly finding approximate nearest neighbor matches between image patches. The main idea behind this scheme is *i*) random sampling is effective in finding good patch matches, and *ii*) natural coherence of imagery allows propagation of such matches to the surrounding areas quickly. We have tried their scheme in our patch matching phase, but found that the number of wrong matches -at least for the images of our data-set- was not low enough for enhancement purposes. However, we believe that similar optimizations such as the ones proposed above could be used in our scheme to speed up the enhancement process. We plan to implement such optimizations to our system in future.

5 Conclusion and Future Work

In this work, we presented a new approach to the *quality transfer problem*: Given a low-quality image how could you enhance it to include additional details. Our approach which is based on creating a temporal, high-quality image data-set, enhances the input low-quality image by finding the most relevant and visually interesting points from the images in the data-set and copying the best matches over to create a new synthesized image with more details. We are planning to augment our current data-set with manually-taken pictures of mobile phones in a similar way to [19]. We are also working on improving the speed of patch matching by using tree structures [21, 8, 10, 11, 20]. Another optimization is to store the extracted salient features of each high-quality image in a database to allow faster matchings.

References

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.

[2] http://chdk.wikia.com/.

[3] http://www.youtube.com/watch?v=3uoM5kfZIQ0.

[4] Alexei Efros and Thomas Leung. Texture synthesis by non-parametric sampling. In *In International Conference on Computer Vision*, pages 1033–1038, 1999.

[5] www.flickr.com.

[6] http://www.gphoto.org.

[7] James Hays and Alexei A Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3), 2007.

[8] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 327–340, New York, NY, USA, 2001. ACM.

[9] http://jiexplorer.sourceforge.net/.

[10] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. Solid texture synthesis from 2d exemplars. *ACM Trans. Graph.*, 26, July 2007.

[11] Neeraj Kumar, Li Zhang, and Shree Nayar. What is a good nearest neighbors algorithm for finding similar patches in images? In *Proceedings of the 10th European Conference on Computer Vision: Part II*, pages 364–378, Berlin, Heidelberg, 2008. Springer-Verlag.

[12] Gareth Loy and Er Zelinsky. A fast radial symmetry transform for detecting points of interest. In *In: 7 th Euproean Conference on Computer Vision*, page 358. Springer, 2002.

[13] http://www.youtube.com/watch?v=Vxq9yj2pVWk.

[14] Aude Oliva and Antonio Torralba. Building the gist of a scene: the role of global image features in recognition. In *Progress in Brain Research*, page 2006, 2006.

[15] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511, October 2005.

[16] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[17] Antonio Torralba, Kevin P. Murphy, William T. Freeman, and Mark Rubin. Context-based vision system for place and object recognition. In *In International Conference on Computer Vision*, pages 273–280, 2003.

[18] Antonio Torralba, Antonio Torralba, Rob Fergus, and William T." freeman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 2008.

[19] Kathleen Tuite, Noah Snavely, Dun-Yu Hsiao, Adam M. Smith, and Zoran Popović. Reconstructing the world in 3d: bringing games with a purpose outdoors. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 41–44, New York, NY, USA, 2010. ACM.

[20] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 479–488, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[21] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29:463–476, March 2007.