Street Sweeper: Detecting and Removing Cars in Street View Images

Wei-Ta Chu, Ying-Chieh Chao, and Yi-Sheng Chang
National Chung Cheng University, Taiwan
wtchu@cs.ccu.edu.tw, raff1219@hotmail.com, kero033@yahoo.com.tw

ABSTRACT

To protect privacy of individuals or companies that may be leaked in street view images, we present a system to automatically detect and remove cars as if they had never been there. Although street view service providers have made efforts on blurring human faces and license plates, we argue that remaining features, such as license numbers and phone numbers printed on car bodies, could cause privacy leak. Given a sequence of street view images, this system first detects cars by the deformable part model, and then determines foreground/background seeds for the GrabCut image segmentation module in order to facilitate automatic car segmentation. After removing cars, an exemplar-based inpainting method is developed with special designs on filling priority determination and road structure propagation. Hierarchical texture propagation and randomized texture propagation are integrated to implement the inpainting process, so that aesthetically pleasing inpainting results as well as privacy protection can be accomplished.

Keywords

Cascade deformable part model; Grabcut; road structure propagation; hierarchical texture propagation; randomized texture propagation.

1. INTRODUCTION

Map services have been widely utilized in many ways, such as trip planning, shop finding, and automatic navigation. Recently, several map service providers, such as Google Maps and Bing Maps, have introduced a new service: street view or streetside exploration. As a part of map services, street view images provide more detailed information than traditional map. Based on street view images, location-based services can be provided with highly interactive interfaces, and users can enjoy traveling on streets around the world by just clicking. However, as we can see, vehicles appearing in street view images not only affect usage of this service, but also leak privacy of owners of these vehicles. Although there have

been studies for blurring license plates [1][2], they are clearly not enough to protect privacy of the objects/persons that were unintentionally captured by the camera car. As shown in Figure 1, on vehicles detailed information other than the license plates would still leak privacy information, such as the company's name or phone numbers printed on car bodies. Sometimes people can still recognize motorbike riders even only the shape of body and the motorbike are shown. Although Google allows users to request further blurring, it is impossible for users to review all street view images to protect their own privacy. Therefore, the goal of this work is to automatically detect and remove the whole cars in street view images, as shown in Figure 2.

This work can be viewed to deal with a problem that is in-between of image inpainting and video inpainting. In image inpainting, only information within an image can be exploited to reconstruct the missing region when some objects are removed. In video inpainting, strong consistency between densely sampled frames can be exploited to interpolate the missing regions. In our problem, "sparsely sampled images" have relatively weaker consistency, and thus existing video inpainting techniques cannot be directly employed.

Figure 1. Detailed information on vehicles would leak privacy.

Figure 2. Two sample pairs of street view images and the corresponding inpainting results.

One may propose to mosaic or blur vehicle regions to protect privacy, just like blurring license plates. To verify this, we conducted a pilot study where both mosaic results and inpainting results were presented to thirteen users, who were asked to give comments about how satisfactory these results are. The subjective evaluation results show that eleven of the thirteen users prefer inpainting results for privacy protection. Although both mosaic and inpainting are able to protect privacy, inpainting results are more aesthetically pleasing and acceptable. In fact, inpainting and mosaic techniques are not conflicting to protect privacy in street view images. For the cases our system doesn't work well, e.g., images with complex road conditions, mosaic effects can be employed as an alternative to protect privacy.

In this paper, we focus on analyzing street view images, automatically detecting cars and removing them, and filling missing regions with inpainting. This system includes the following key components:

- Automatic car detection: Street view images are very complex due to significant variations of lighting conditions, viewing angles, and deformation. All these factors cause severe noises and make automatic car detection quite challenging. We employ the cascade deformable part model [3] to detect cars, and apply the Grabcut algorithm [4] to segment car regions. In contrast to the conventional Grabcut algorithm where foreground seeds and background seeds are assigned manually, we devise an automatic seed finding approach based on results of road detection and car detection.
- Road structure propagation: Images along the same street were captured consecutively and often contain spatial continuity, which can then be used to reconstruct the missing region. In the proposed system, road structure mainly coming from high-gradient spatial continuity, e.g., lane lines and crash barrier, is propagated to neighboring images ahead of the intra-image inpainting process. If road structure can be extracted and propagated appropriately, the missing regions would be well bounded by high-gradient structure, and thus the problem of inpainting is eased.
- Hierarchical and randomized inpainting: The order of inpainting is determined by considering gradient information, and techniques of hierarchical texture propagation [5] and randomized texture propagation [6] are combined in order to obtain appealing inpainting results both for smooth regions and high-gradient regions.

The remainder of this paper is organized as follows. Literature survey is given in Section 2, and necessary preprocesses are described in Section 3. In Section 4, details of automatic car detection and segmentation are provided. The road structure propagation process especially designed for street view

images is given in Section 5. In Section 6, we describe the idea of exemplar-based inpainting, how we determine the filling priority, and how to develop a hybrid method to make results more pleasing. Performance evaluation and limitations of the current work will be presented in Section 7, followed by conclusion and future work in Section 8.

2. RELATED WORKS

2.1 Applications on Street View Images

Street view services such as Google Street View [7] or Bing Streetside have emerged as a novel location-related application providing street-level images of entire cities. Many interesting applications can thus be developed for navigation purposes. Yushimoto et al. [8] developed a 2D/3D navigation system where users issue gesture commands to browse Google Map and Google Street View. From Google Street View, Guy and Truong [9] collected rich intersection information and developed a system called CrossingGuard to provide details of intersection geometry for visual impaired pedestrians. Koef et al. [10] developed a novel interface that seamlessly interchanges browsing between bubbles and multi-perspective panoramas so that a targeted location can be efficiently identified.

Privacy issues in street view images have attracted much attention. Especially in Europe, many countries claimed that Google breaches one or more EU laws. Google responded to this by blurring faces and license plates [1]. Flores and Belongie [11] argued that articles of clothing, body shape and height, may still leak privacy, and proposed a pedestrian removal method when multiple images capturing the same pedestrian and redundant background are available.

In addition to pedestrians, we argue that vehicles or bicycles/motorbikes riders would also leak privacy information, as shown in Figure 1. Based on the street view images capturing the sidewalk and pedestrians with façade of building as background, the work in [11] utilized rich feature points in the background and estimated the geometric relationship between two neighboring images by a planar perspective transformation. In our case, the processed street view images largely contain smooth road regions, with significant scene changes on the roadside (see Figure 7 for example), and thus the relationship between neighboring images cannot be well described by a planar perspective transformation. Moreover, images fed to the system in [11] were manually filtered so that image pairs containing the same pedestrians were known in advance. In our case, given a sequence of street view images where vehicles may appear in arbitrary numbers of images, we aim to automatically detect and remove vehicles on the road, and to reconstruct images with inpainting techniques.

2.2 Inpainting

Inpainting techniques can be roughly categorized into two groups: PDE-based (partial derivative equation) schemes [12] and exemplar-based schemes [13]. The PDE-based scheme propagates texture in a given direction and often introduces blur effects due to the adopted diffusion method. On the other hand, the exemplar-based scheme copies texture from neighboring image patches and is often able to derive more structured content. In our case, we need to smoothly fill the missing regions located on the road, and sharply fill missing regions located on lane lines or crash barriers with structure information. The proposed system extracts road structure from spatially adjacent images, and then fills missing regions suspected to be highly structured with the road structure coming from neighboring images. To reconstruct the left missing regions, we propose a directional inpainting method modified from [5], and team it up with a randomized exemplar-based method, i.e., PatchMatch [6]. The inpainting results thus have sharp structure in line-like regions and have smooth texture in the road region.

2.3 Object Detection and Recognition

The targeted objects to be removed are vehicles, and thus related works on object detection and recognition, especially for vehicles, are briefly surveyed here. Mainly taking car objects as examples, Agarwal et al. [22] proposed a canonical part-based representation and learned a classifier to detect side-view cars in varying conditions consisting of cluttered background and mild occlusion. Hota et al. [23] adopted the Adaboost-based classifier with Haar-like features followed by support vector machine (SVM) based classifiers with histogram of oriented gradient (HOG) to detect side-view and rear-view cars. Inspired by that cars are artificial objects with consistent characteristics in structure, Zheng and Liang [24] proposed image strip features to facilitate efficient detection of multi-view cars. They also proposed a complexity-aware criterion in the boosting scheme, in order to use cheaper features in earlier levels to make more speedup. Kuo and Nevatia [25] constructed lower dimensional embedding of cars in various views, and learned a cascaded tree classifier to achieve robust multi-view car detection.

In addition to the works mentioned above, tremendous studies have been proposed to detect and recognize objects in images. Among them, deformable part models (DPM) [15] achieve state-of-the-art results in the PASCAL object detection challenges[1], and thus we adopt this approach to detect cars. In DPM, an object class is represented by mixtures of multiscale deformable part models, which consist of

[1] http://pascallin.ecs.soton.ac.uk/challenges/VOC/

a coarse root filter and several higher resolution part filters. A spatial model is also constructed to describe how each part is located relatively to the root. Taking the rear view of a car as an example, we can describe a car by parts consisting of tires, taillights, and windscreen. Histograms of oriented gradients (HOG) [16] are used to describe the whole object and its parts, respectively. The root filter and part filters are trained based on latent SVMs. At the detection stage, given a window centered by a point, responses of the root filter and part filters are summed up and fed to the SVM to determine the occurrence of an object.

2.4 Image Segmentation

Image segmentation has been one of the most challenging problems in computer vision and image processing society for decades. Fully automatic image segmentation is a holy grail of image understanding researchers and keeps attracting researcher's efforts until now. Although elegant methods like normalized cuts [26] and graphs [27] were developed, performance of fully automatic segmentation methods is still limited and is much inferior to what humans can do. Therefore, semi-automatic image segmentation methods, i.e., limited user intervention is needed to specify foreground and background, become popular in recent years. In the Graph Cut framework, the user specifies certain pixels that absolutely have to be part of the foreground. Relationships between pixels are described as a graph [14], which is then segmented by the graph cut algorithm to discriminate foreground from background. Modified from the graph cut approach, the GrabCut algorithm [4] is constructed based on an energy function that jointly considers a data term indicating how likely color of pixels matches with the foreground model and the background model, and a smoothness term indicating consistency between neighboring pixels. Users can guide the GrabCut algorithm by simply drawing a rectangle to roughly discriminate foreground from background. Also based on the graph cut framework, the lazy snapping system [17] allows users to specify foreground seeds and background seeds by strokes. In our work, we rely on the GrabCut algorithm to segment car objects from the road region, with a design of automatic seed finding.

3. PREPROCESSING

3.1 Road Detection

Two factors are considered in our road detection module. First, the road region occupies most areas of bottom halves of images, and thus the dominant color in the bottom half is an important clue for

detecting road. Second, the road region in images keeps changing according to terrain, street layout, and lanes. Therefore, we need an adaptive method to detect the road region.

Given consecutive street view images $\{\boldsymbol{f}_1, \cdots, \boldsymbol{f}_n\}$, a sliding window with size of three images is applied to scan the sequence. To detect the road region of the ith image $\boldsymbol{f}_i$, the RGB color histograms of the bottom halves of $\boldsymbol{f}_{i-1}$, $\boldsymbol{f}_i$, and $\boldsymbol{f}_{i+1}$ are extracted and accumulated, denoted by $\boldsymbol{H}_i$. Based on $\boldsymbol{H}_i$, an iterative region growing algorithm is proposed to find the road region, as illustrated in Figure 3. From $\boldsymbol{H}_i$ the color with the highest peak is first determined, denoted by $\boldsymbol{c}_1$ (step 2). The set of pixels P_1 in the bottom half of $\boldsymbol{f}_i$ and with the color $\boldsymbol{c}_1$ is then identified (step 3) and is added to the targeted set R. For each pixel $\boldsymbol{x}$ in R, we find pixels $\boldsymbol{y}$ in $\boldsymbol{x}$'s neighborhood. If pixels $\boldsymbol{y}$ are with similar color to $\boldsymbol{x}$ (denoted by $\boldsymbol{c}_{\boldsymbol{y}} \approx \boldsymbol{c}_{\boldsymbol{x}}$), they are added into the set R (step 5). If the area of the grown region $|R|$ is not larger than a threshold, the growing procedure repeats again by starting from the pixels with the color corresponding to the second peak of $\boldsymbol{H}_i$. In this work, RGB histograms with each color component represented by eight bits are extracted for analysis. The neighborhood $\mathcal{N}(\boldsymbol{x})$ is defined as the 8-connected pixels of the pixel $\boldsymbol{x}$. The threshold δ is set as 20% of the area of the whole image.

Figure 4(b) shows three intermediate results of region growing based on the pixels with the color corresponding to the highest peak. From these results we see that most of the road region can be detected, except for lane lines that are with distinct colors from the road. We find the minimum polygon covering the intermediate results to obtain final road detection results, as shown in Figure 4(c). Based on road detection results, we would adaptively construct an RGB histogram of the road in a spatial locality, which will be used in car segmentation and inpainting later.

Algorithm: Road detection

Input: the color histogram $\boldsymbol{H}_i$
Output: the road region R

1. Set $R = \emptyset$, $j = 1$.
2. Find the color $\boldsymbol{c}_j$ corresponding to the largest peak of $\boldsymbol{H}_i$.
3. Find the set of pixels P_j in the bottom half of $\boldsymbol{f}_i$ and with color $\boldsymbol{c}_j$.
4. Set $R = R \bigcup P_j$.
5. For each pixel $\boldsymbol{x} \in R$, find pixels $\boldsymbol{y}$ its neighborhood $\mathcal{N}(\boldsymbol{x})$, i.e., $\boldsymbol{y} \in \mathcal{N}(\boldsymbol{x})$. if $\boldsymbol{c}_{\boldsymbol{y}} \approx \boldsymbol{c}_{\boldsymbol{x}}$, set $R = R \bigcup \boldsymbol{y}$
6. If $|R| > \delta$, stop.
   Otherwise, for $\boldsymbol{H}_i$, set the value of the bin of corresponding to $\boldsymbol{c}_j$ as zero, $j = j + 1$, and go to step 2.

Figure 3. The road detection algorithm.

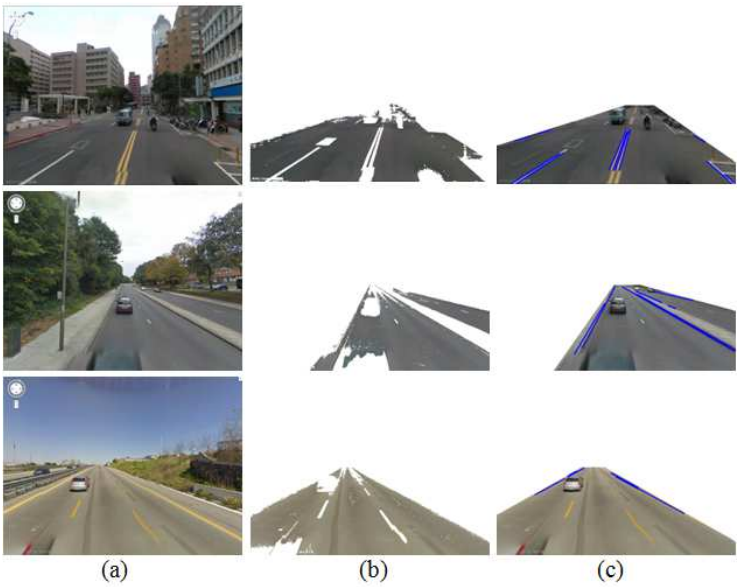

Figure 4. (a) The original images; (b) Intermediate results of region growing based on the pixels with the color corresponding to the highest peak; (c) Final road detection results and line detection results in the road region.

3.2 Line Detection

Lane lines, which ought to be consistent in a small spatial locality, provide important cues to represent road structure. Following the standard line detection procedure, we apply Gaussian blur to the road region, and then transform the road region into the Hough space. Peaks in the Hough space are then selected, which correspond to prominent lines in the original image space. Making blurring in advance facilitates detecting prominent lines. Figure 4(c) illustrates prominent lines detected in the road region.

4. CAR DETECTION AND SEGMENTATION

4.1 Car Detection

We rely on the framework of the deformable part model [15] to detect on-road cars. Because evaluating all configurations of the root filter and part filters for each sliding window is time consuming, the cascade DPM approach was proposed in [3], where the root filter and part filters are put in a cascade, and at the detection stage only the regions passing early evaluation are further evaluated. This early pruning scheme largely improves detection efficiency without sacrificing detection accuracy. In our work, we employ the cascade DPM to detect cars.

Figure 5 illustrates the car detection process. Cascade DPMs trained for specific vehicles, such as sedan, truck, and bus, can be adopted to detect all possible types of cars. Sliding windows in the representation of HOGs are extracted at various scales, and are then examined to tell whether cars appear inside these windows. Note that if a sliding window obviously contains no sedan, for example, it would be rejected by the cascade DPM for sedan at early stages to avoid unnecessary examination by later part filters. In our work, we found the cascade DPM for "car" constructed by the authors in [3] already achieves satisfactory detection performance. Hence, we do not train our vehicle-specific models and simply use the general "car" model provided by [3]. Figure 6(a) and Figure 6(h) show two sample results of car detection. More examples will be provided in the evaluation section.

The cascade DPM gives bounding boxes of detected cars. To prevent the following inpainting process from too many noises, more accurate car regions are needed. Based on the detected bounding boxes, the GrabCut algorithm is then used to obtain fine segmentation.

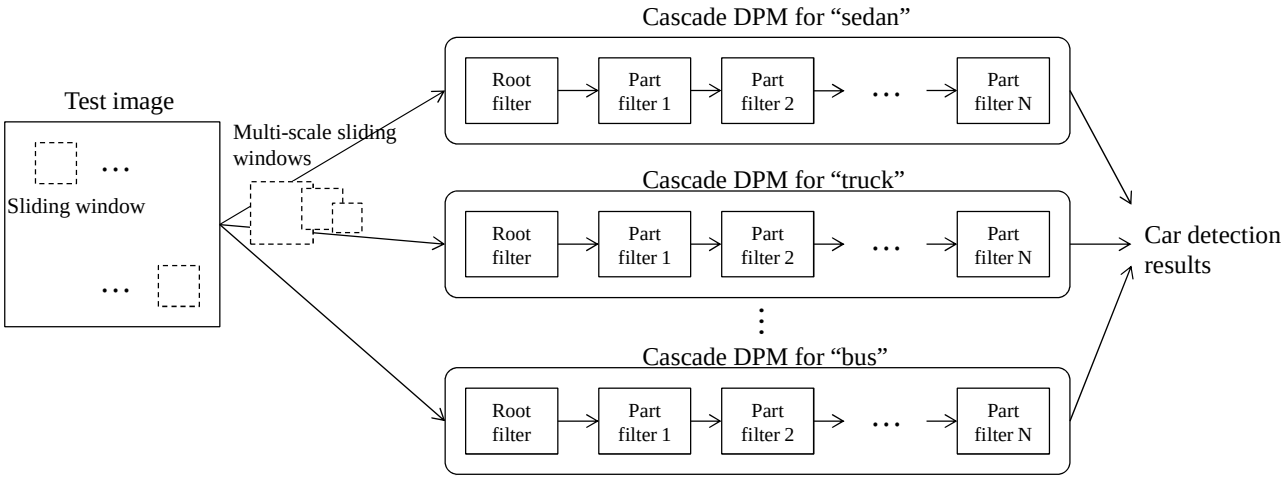

Figure 5. Illustration of the car detection process.

4.2 Car Segmentation

We employ the GrabCut framework [4] to segment car objects from the road region. However, there are tremendous street view images, and even a small amount of user interaction needed is not feasible. We thus design an approach that initiates foreground/background seeds. With these seeds, color distributions of foreground and background are determined, and the GrabCut algorithm is adopted to segment cars.

Foreground seeds are determined based on the bounding box of car detection. Thirty percent of pixels in the bounding box are sampled, according to a probability distribution defined by a two-dimensional Gaussian centered at the spatial centroid of the bounding box. According to the Gaussian distribution,

pixels nearer to the centroid would be sampled more. On the other hand, three issues are jointly considered to determine background seeds:

- Thirty percent of pixels in the road region are randomly sampled as background seeds. These seeds are important for discriminating cars from the road because most foreground seeds are surrounded by the road.
- Pixels on lines are selected as background seeds. Also surrounded by the road, lane lines have significantly different appearance to the road, and are easily misclassified as foreground if pixels on them are not especially indicated as background.
- In addition to the car bounding boxes, the road region, and lines, ten percent of pixels in the remaining region are randomly sampled as background seeds.

Figure 6 includes two samples showing foreground and background seeds and segmentation results of the GrabCut algorithm. The blue dots in Figure 6(b)(i) illustrate foreground seeds. The red dots in Figure 6(c)(j) and Figure 6(d)(k) illustrate background seeds sampled from the road and lines, respectively. Note that for clear illustration only parts of seeds are illustrated in these figures. Figure 6(e)(l) illustrate all foreground seeds and background seeds. The initial segmentation results in Figure 6(f)(m) contain some false detection, which can be eliminated by morphological operations including erosion and dilation. Figure 6(g)(n) show final results. Sometimes, false detection regions still remain, like the case shown in Figure 6(g). This is not a serious problem because even if the region is erroneously removed from this image, we would fill it with pixels on the road later by inpainting techniques.

We should notice that automatic segmentation is very important for tremendous amount of street view images. With the automatic process, we can trace a route on the map and remove all cars in street view images along this route. Seeing Figure 6(a), when there are multiple cars in an image, users need to define foreground rectangles many times, and segment one car a time by the conventional GrabCut tool. With the proposed process, this system automatically defines foreground seeds and background seeds, and multiple cars can be detected and segmented simultaneously.

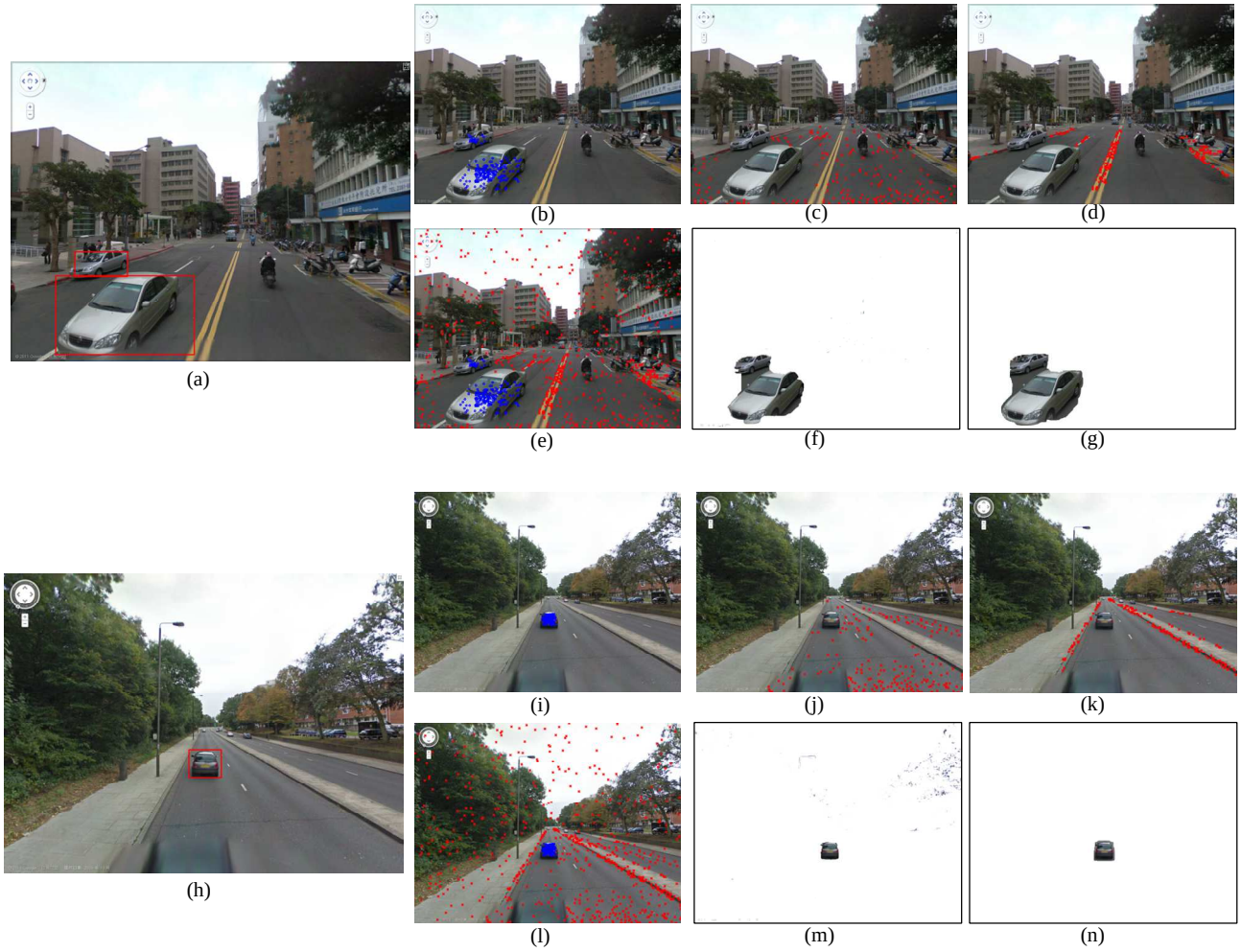

Figure 6. Sample results of car detection and GrabCut-based segmentation. (a)(h) The original images and car bounding boxes. (b)(i) Foreground seeds on cars (blue circles). (c)(j) Background seeds on the road (red crosses). (d)(k) Background seeds on lines. (e)(l) All automatically determined seeds. (f)(m) Segmentation results. (g)(n) Refined segmentation results after morphological operations.

5. ROAD STRUCTURE PROPAGATION

From the classic work [13] to the most recent state of the art [28], missing regions with geometric structure are generally given higher priority to be filled first. When filling the remaining regions, the inpainting process is then guided by the “already” filled regions to achieve promising results. We also follow this thought in this work, but in addition to existing regions within the targeted image, information from spatially neighboring street view images can also be utilized in inpainting highly structured regions. In this section, we consider information across images to fill highly structured regions.

In Section 6, information within an image is considered in determining filling priority, as done by many previous works.

In street view images, the most important (consistent) geometric structure across images is road structure, and thus we aim to fill such regions first. Important road structures mainly consist of land lines and boundaries between the road and the sidewalk, which are often represented as prominent straight lines. The importance of identifying and reconstructing road structure is at least twofold. First, humans are sensitive to reconstruction error on straight lines. Using the designed road structure propagation method can more accurately reconstruct line structure. Second, the reconstructed line structure could be an important clue to guide the calculation of filling priority described in Sec. 6.1.

Motivated by the work [18] that collects statistics of patch offsets between images, we design our road structure propagation process as follows. Given five spatially consecutive images $\{I_{i-2}, I_{i-1}, I_i, I_{i+1}, I_{i+2}\}$, where prominent lines in each image have been detected and cars have been detected and removed, we would like to construct important road structure in the missing region of I_i. Each image is first divided into $4 \times 4 = 16$ blocks. For every two consecutive images, say I_i and I_{i+1}, we collect horizontal offset information of prominent lines block by block. Let $P = \{p_k\}$ denote the set of pixels on prominent lines in the jth block of I_i, and $Q = \{q_k\}$ denote the set of pixels on prominent lines in the jth block of I_{i+1}. For each pixel $p_k \in P$, the pixel $q_{k'} \in Q$ that has the same vertical coordinate as p_k, denoted by $p_k(y) = q_{k'}(y)$, is found. That is,

$$q_{k'} = \arg\min_{q_k \in Q}\{p_k(x) - q_k(x) | p_k(y) = q_k(y)\}. \quad (1)$$

The horizontal offset is calculated as $o_k = p_k(x) - q_{k'}(x)$. If no pixel in Q having the same vertical coordinate as p_k, the offset o_k is undefined. The horizontal offset histogram $H^j_{i,i+1}$ between the jth blocks of I_i and I_{i+1} is then constructed after all pixels $p_k \in P$ are examined.

According to the horizontal offset histogram, we fill missing regions in I_i in a block-by-block manner. Assume that the largest peak of $H^j_{i,i+1}$ corresponds to the horizontal offset $\hat{o}$, a missing pixel p in the jth block of I_i is filled with a known pixel q in the jth block of I_{i+1} if q is on prominent lines, $p(y) = q(y)$, and $p(x) = q(x) + \hat{o}$. For the missing pixel p, if there is no known pixel q in I_{i+1} such that $p(x) = q(x) + \hat{o}$, or a known pixel q is not on prominent lines, appropriate pixels satisfying the criteria mentioned above are searched from I_{i-1}, I_{i+2}, and I_{i-2}, according to largest peaks in $H^j_{i,i-1}$, $H^j_{i,i+2}$, and $H^j_{i,i-2}$, respectively. All blocks in I_i that consist of missing regions are filled with road structure by the aforementioned process.

Figure 7(a) shows five consecutively captured images, and Figure 7(b) shows corresponding results of car removal and line detection. Note that a lane line is often detected as two parallel thin lines. To fill

missing regions with complete lane lines or important line-like structure, we apply a dilation operation for each pixel on detected lines. Figure 7(c) shows the expanded lines in each image, where pixels on them are viewed as known pixels, and are used to fill missing regions with the consideration of offset histograms. Figure 7(d) shows the propagation results of I_i. By considering road structure in spatially adjacent images, the line structure between road and sidewalk is constructed even though the missing region is extremely large.

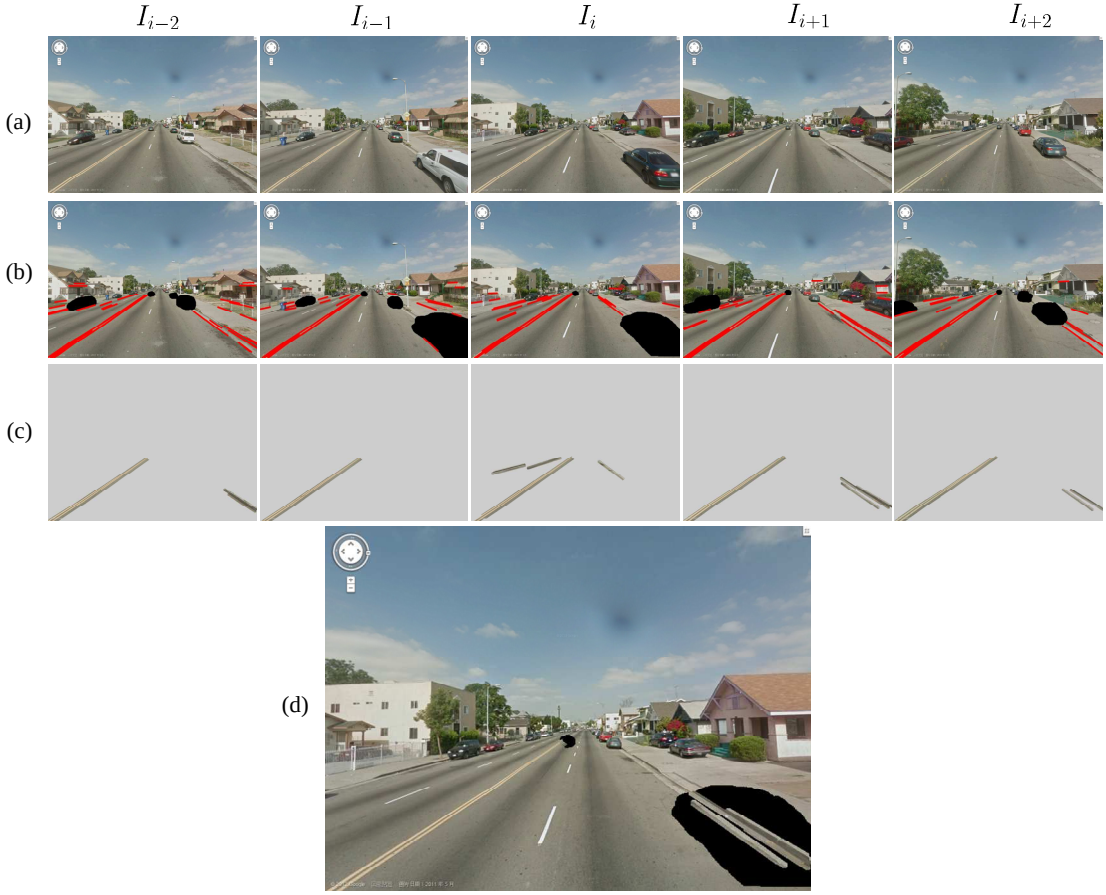

Figure 7. (a) Original images. (b) Images after removing cars and detecting lines (shown in red). (c) Lines with dilation. (d) The propagation result of the image I_i.

6. INPAINTING

After filling the road structure by considering the inter-image relationship, we adopt inpainting techniques considering intra-image patches to fill the remaining regions. An inpainting process consists of two important stages: filling order determination and texture propagation. In filling order determination, we modify the priority definition in [5] by considering road characteristics and texture relationship between missing and existing areas. In the texture propagation process, two state-of-the-art

exemplar-based methods, i.e., [5] and [6], are integrated. If gradient magnitude of a missing patch is large, a hierarchical texture propagation method modified from [5] is adopted. Otherwise, the randomized propagation method defined in [6] is adopted. In both propagation processes, known exemplars are searched from a range guided by the direction of the prominent line that is spatially closest to the missing patch.

6.1 Determining Filling Priority

An image I may have several missing regions, denoted by $\Omega = \{\omega_1, ..., \omega_n\}$. The source region is denoted by ϕ, and $\phi = I - \Omega$. A square image patch centered by the pixel $\boldsymbol{p}$ is denoted by ψ_p. In [5], the filling order (priority) of the patch ψ_p on the boundary of a missing region ω_i is defined by the product of two terms: $P(\boldsymbol{p}) = C(\boldsymbol{p})D(\boldsymbol{p})$. The first term indicates the confidence value, and is designed to favor patches having more known pixels:

$$C(\boldsymbol{p}) = \frac{\sum_{q \in \psi_p \bigcap (I-\Omega)} C(\boldsymbol{q})}{|\psi_p|}, \tag{2}$$

where $|\psi_p|$ is the area of ψ_p. The confidence value is initialized as $C(\boldsymbol{p}) = 1, \forall \boldsymbol{p} \in (I - \Omega)$ and $C(\boldsymbol{p}) = 0, \forall \boldsymbol{p} \in \Omega$.

The second term considers data characteristics around the patch ψ_p and is designed to favor image patches having high color gradients. A method modified from the approach proposed in [19] is described in the following. To eliminate the influence of noise, the image patch ψ_p is first smoothed by a Gaussion kernel. For a pixel in ψ_p and in the source region ϕ, its horizontal gradient magnitudes ρ_x^R, ρ_x^G, and ρ_x^B in R, G, and B color channels are calculated, respectively, and the overall horizontal gradient magnitude is calculated as $\rho_x = \rho_x^R + \rho_x^G + \rho_x^B$. Similarly, the overall vertical gradient magnitude is calculated as $\rho_y = \rho_y^R + \rho_y^G + \rho_y^B$. Each pixel is described by a vector $[\rho_x \quad \rho_y]^T$, and the gradient magnitudes of the pixels in the n by n image patch ψ_p are then arranged as a 2 by n^2 matrix $\boldsymbol{J}$. Eigen-decomposition is then applied to the matrix $\boldsymbol{J}$, and the first and the second largest eigenvalues λ_1 and λ_2 are found. The data term D is accordingly defined as:

$$D(\boldsymbol{p}) = \alpha + (1 - \alpha) exp\Big(-\frac{\eta}{(\lambda_1 - \lambda_2)^2 + \epsilon}\Big), \tag{3}$$

where η is a positive value and $\alpha \in [0, 1]$. When the image patch has larger gradient, i.e., $\lambda_1 \gg \lambda_2$, the data term is emphasized, and the filling priority raises. The term ϵ is a very small positive number to avoid a zero denominator. In [5], α is set as 0.001. The size of an image patch is 16×16.

We modify the data term $D(\boldsymbol{p})$ mentioned above to meet the conditions of street view images. In street view images the road region is visually smooth, i.e., low spatial frequency area. In contrast to

conventional inpainting techniques that usually work well for high-frequency regions, small amounts of inpainting errors in the road region would be perceptually significant. Therefore, we especially modify the data term by considering two factors: the color gradient around the boundary pixels, and the spatial relationship between existing and missing area around the boundary pixels.

The first factor is defined as

$$d_1(\boldsymbol{p}) = exp^{-(\lambda_1 - \lambda_2)}. \tag{4}$$

When the existing area of ψ_p is smooth, i.e., $\lambda_1 \approx \lambda_2$, this term is larger, and the filling priority raises.

The second factor comes from implicit texture of the road. Although the road region is generally smooth, there is often implicit directional texture due to long-term extrusion by cars. We jointly consider the spatial relationship between missing and existing areas, and the directional texture of the road. First, the spatial centroid $c_{\mathcal{M}}$ of the missing area in ψ_p, and the spatial centroid $c_{\mathcal{E}}$ of the existing area in ψ_p, are calculated. The vector from $c_{\mathcal{E}}$ to $c_{\mathcal{M}}$ is then determined and normalized, and is denoted as $\boldsymbol{v}_p$. Assume that the direction of the prominent line (based on the results of Sec. 3.2) closest to the patch ψ_p is $\boldsymbol{v}_l$ (normalized to a unit vector), the consistency between $\boldsymbol{v}_p$ and $\boldsymbol{v}_l$ is measured by

$$d_2(\boldsymbol{p}) = exp^{-|\boldsymbol{v}_p \cdot \boldsymbol{v}_l|}. \tag{5}$$

Figure 8 shows an illustration of determining filling priority. Let us consider three boundary patches ψ_{p_1}, ψ_{p_2}, and ψ_{p_3}. The black dash lines passing through these patches illustrate the vector $\boldsymbol{v}_l$, and the red dash lines represent the spatial relationship between missing areas and existing areas. From this figure we can see that the spatial relationship in ψ_{p_3} most coheres with the closest prominent line, and thus $d_2(\boldsymbol{p}_3)$ is the largest. From this perspective, ψ_{p_3} is preferred to be filled first.

Finally, the modified data term $\hat{D}(\boldsymbol{p})$ is defined as $\hat{D}(\boldsymbol{p}) = d_1(\boldsymbol{p})d_2(\boldsymbol{p})$, and the overall filling priority for the patch ψ_p is determined by $\hat{P}(\boldsymbol{p}) = C(\boldsymbol{p})\hat{D}(\boldsymbol{p})$. Patches on boundaries of missing regions are filled in descending order of the priority. Filling priorities will be recomputed after all the patches on boundaries of missing regions have been filled. That is, a missing region is progressively filled from the border to the center.

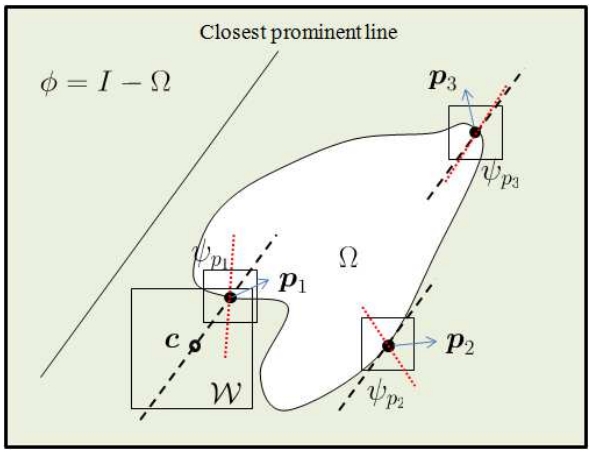

Figure 8. Illustration of determining the filling priority (described in Sec. 6.1), and the search window along the direction of the prominent line (described in Sec. 6.2).

6.2 Hierarchical Texture Propagation

In [5], the best image patch to be used to fill the boundary patch ψ_p is determined by

$$\psi_{\hat{q}} = \arg\min_{\psi_q \in \mathcal{W}} \Big(d(\psi_p, \psi_q) + (\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2})^2 \times f(p,q) \Big), \tag{6}$$

where $d(\psi_p, \psi_q)$ denotes the sum of squared differences between a candidate patch $\psi_q \in \mathcal{W}$ and the already filled or known pixels of ψ_p. In [5], the search window $\mathcal{W}$ is entirely the source region, i.e., $\mathcal{W} = \phi$. The second term gives higher costs for image patches with larger color gradient (i.e., larger difference between λ_1 and λ_2). On smooth areas, this term tends to 0. The function $f(p,q)$ is designed as

$$f(p,q) = \Big(\epsilon + \frac{|\boldsymbol{v}_p \cdot \boldsymbol{v}_{p \to q}|}{\|\boldsymbol{v}_{p \to q}\|} \Big)^{-1}, \tag{7}$$

where $\boldsymbol{v}_{p \to q}$ is the vector from the center of ψ_p to the center of ψ_q. The value ϵ is set as 0.001. If the vector $\boldsymbol{v}_{p \to q}$ is not collinear to $\boldsymbol{v}_p$, this term gives a penalty attributed to $|\boldsymbol{v}_p \cdot \boldsymbol{v}_{p \to q}|$. When two vectors are collinear, the function $f(p,q)$ tends to one.

For the same reason to design eqn. (5), in order to favor candidate image patches that are relatively located at the direction similar to the closest prominent line, we limit the search window $\mathcal{W}$ in eqn. (6) as follows. The closest prominent line usually comes from lane lines or crash barrier, and coheres with implicit trace on the road. For a boundary patch ψ_p centered at (x_p, y_p), if the gradient of its closest prominent line is (d_x, d_y), we first find a point $\boldsymbol{c}$ as $\boldsymbol{c} = (x_p, y_p) + (2d_x, 2d_y)$. The search window $\mathcal{W}$ for finding the best patch is defined as

$$\begin{aligned} \mathcal{W} = \{ \boldsymbol{q} | & \boldsymbol{c}(x) - w_1/2 \le \boldsymbol{q}(x) \le \boldsymbol{c}(x) + w_1/2, \\ & \boldsymbol{c}(y) - w_1/2 \le \boldsymbol{q}(y) \le \boldsymbol{c}(y) + w_1/2 \} \end{aligned} \tag{8}$$

where $\boldsymbol{q}(x)$ and $\boldsymbol{q}(y)$ denote the x and y coordinates of the point $\boldsymbol{q}$, respectively. By this definition, the $w_1 \times w_1$ square region centered at $\boldsymbol{c}$ is the search window. The image patch centered at each $\boldsymbol{q}$ in $\mathcal{W}$ is a candidate patch, and is compared with ψ_p according to eqn. (6). Figure 8 illustrates the search window for $\boldsymbol{p}_1$, along the direction of the closest prominent line. In this work, w_1 is set as 60 pixels.

To further improve robustness, we enhance the search process with the idea of hierarchical decomposition in motion estimation. Two levels of searching are conducted. In the first-level (coarse-level) search, the image is downscaled twice in both horizontal and vertical dimensions. For a boundary patch, the corresponding search window is determined by eqn. (8), and the search process defined in eqn. (6) is used to find the best patch. Assume that the best patch found in the first-level search is centered at $\boldsymbol{q}^{(1)}$. In the second-level (fine-level) search, we take the point $\boldsymbol{q}^{(1)}$ as the role of $\boldsymbol{c}$ in eqn. (8) and define a new search window in the image at the original scale. The second-level search is then conducted based on the original image and the search window centered by $\boldsymbol{q}^{(1)}$ to further refine the choice of best patch.

Finally, the best ten patches from the second-level search are all considered to fill the missing pixels. The value of a missing pixel $\hat{\boldsymbol{p}}$ in ψ_p is determined by

$$\hat{\boldsymbol{p}} = \frac{\sum_{i=1}^{10} s_i \boldsymbol{r}_i}{\sum_{i=1}^{10} s_i}, \tag{9}$$

where s_i is the similarity between the patch ψ_p and the ith best patch ψ_i [12]. $\boldsymbol{r}_i$ in ψ_i is the pixel located at the same relative position as $\hat{\boldsymbol{p}}$ in ψ_p.

Figure 9 shows six intermediate results sampled from an inpainting process. Focusing on the missing region on the left (the solid-line yellow ellipse), from Figure 9(a) to Figure 9(f) the filling order is mainly from top-right to bottom-left or vice versa. For the missing regions on the right (the dash-line blue ellipse), which are actually parts of lane lines, the filling order is mainly from top-left to bottom-right or vice versa. This example clearly shows the effect brought by the design of eqn. (5).

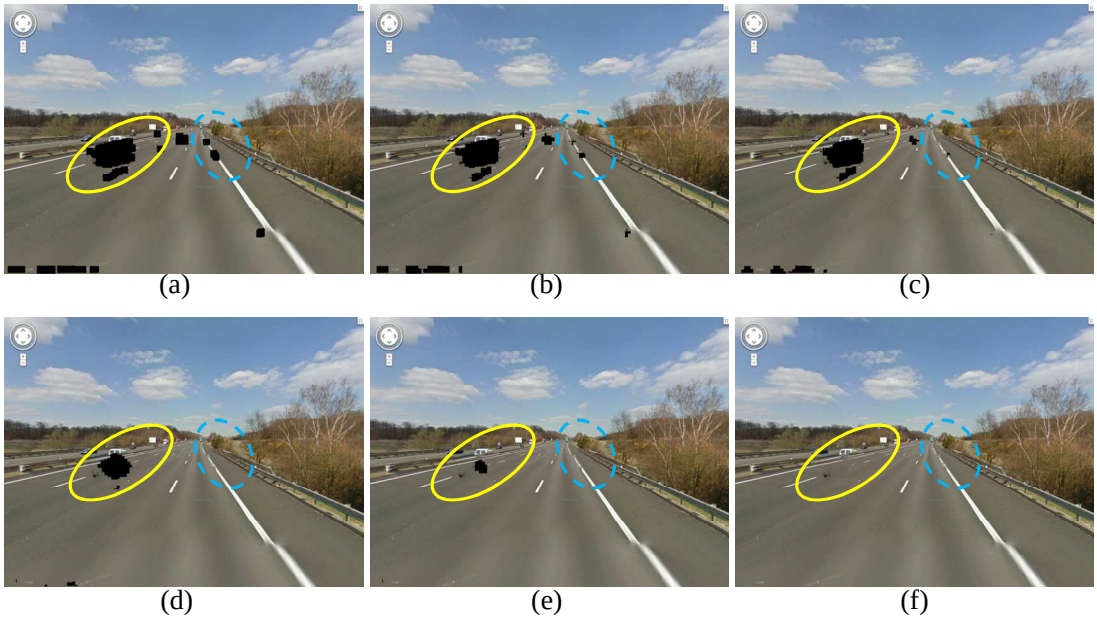

Figure 9. Six intermediate inpainting results sampled from an inpainting process.

6.3 Randomized Texture Propagation

The aforementioned texture propagation method achieves great performance on reconstructing lane lines and boundaries between the road and sidewalk/crash barrier. However, size of the search window defined in eqn. (8) may be content dependent. Another flaw is that inpainting errors would be significantly propagated. For example, if a missing patch ψ_a that should be purely smooth is reconstructed with a smooth patch containing small amounts of high-gradient pixels, the missing patch ψ_b next to ψ_a would inherit the error when ψ_b is reconstructed. This type of error gives little visual inconsistency in high-frequency regions, but gives rise to significant visual annoyance in smooth regions like road. Therefore, to generate pleasing inpainting results, we reconstruct missing patches with high gradient by the hierarchical texture propagation, and reconstruct missing patches with low gradient by the randomized texture propagation described as follows.

To fill a missing patch ψ_p centered at (x, y), the PatchMatch method [6] randomly finds a patch ψ_q centered by (x', y') from the whole known region or the manually defined region. To evaluate the degree how ψ_q is suitable to replace ψ_p, ψ_q's neighboring patch ψ_{ql} centered by $(x' - 1, y')$, and ψ_{qt} centered by $(x', y' - 1)$, are also checked. The idea is that, if ψ_q is a good choice to reconstruct ψ_p, then the distance between ψ_{ql} and ψ_p, and the distance between ψ_{qt} and ψ_p, should both be small. Based on ψ_q, ψ_{ql}, and ψ_{qt}, the best patch to reconstruct ψ_p is determined by $\arg\min\{d(x', y'), d(x' - 1, y'), d(x', y' - 1)\}$, where $d(x', y')$

denotes the patch distance between ψ_p and ψ_q. Assume that the best choice is ψ_{ql}, a choice better than ψ_{ql} is to be found in the next iteration. Start from $(x' - 1, y')$, a few patches within some circular range are randomly sampled, and the same process is applied to find a better choice. In most cases, this iterative procedure quickly converges after four or five iterations [6].

In our work, information of the closest prominent line is used as the prior information for randomization. Suppose that a missing patch ψ_p is located at (x_p, y_p), and the gradient of its closest prominent line is (Δ_x, Δ_y). Rather than randomly sampling from the whole known region, only patches centered by pixels on the extended line that pass through (x_p, y_p) and has gradient (Δ_x, Δ_y) are considered to be the candidate patch ψ_q. Details of the PatchMatch method can be found in [6].

7. EVALUATION

7.1 Car Segmentation

We manually captured Google Street View images along the same street from eight different places to form eight different datasets. Each dataset includes thirty spatially consecutive images, and different datasets have significantly different road situations. Detailed information is shown in Table 1. The blue-shaded regions in Figure 10 are car detection results. Basically, most parts of cars are included if the cascade DPM successfully detects cars. However, due to significant variations of camera views and car occlusion, the cascade DPM often fails to detect all cars in images. The cars that are very close to or very far from the camera are usually not detected, as illustrated by the dash circles in Figure 10. Fortunately, these cars cause less privacy leak, and thus for the purpose of privacy protection, these cases are acceptable.

Table 1. Detailed information of the evaluation dataset.

ID	Type	Loc.	Properties
1	Highway	Asia	Traffic marks, crash barrier, fewer cars
2	Highway	Asia	Traffic marks, crash barrier, fewer cars
3	Highway	USA	Traffic marks, crash barrier, more cars
4	Residential area	USA	Trees, buildings, parking cars, tree shadow
5	Residential area	USA	Tree, buildings, parking cars, narrow lanes
6	Tunnel	Asia	Dusky light
7	Inside city	Asia	Complex traffic marks, buildings, intersection of roads
8	Inside city	Asia	Complex traffic marks, buildings, trees

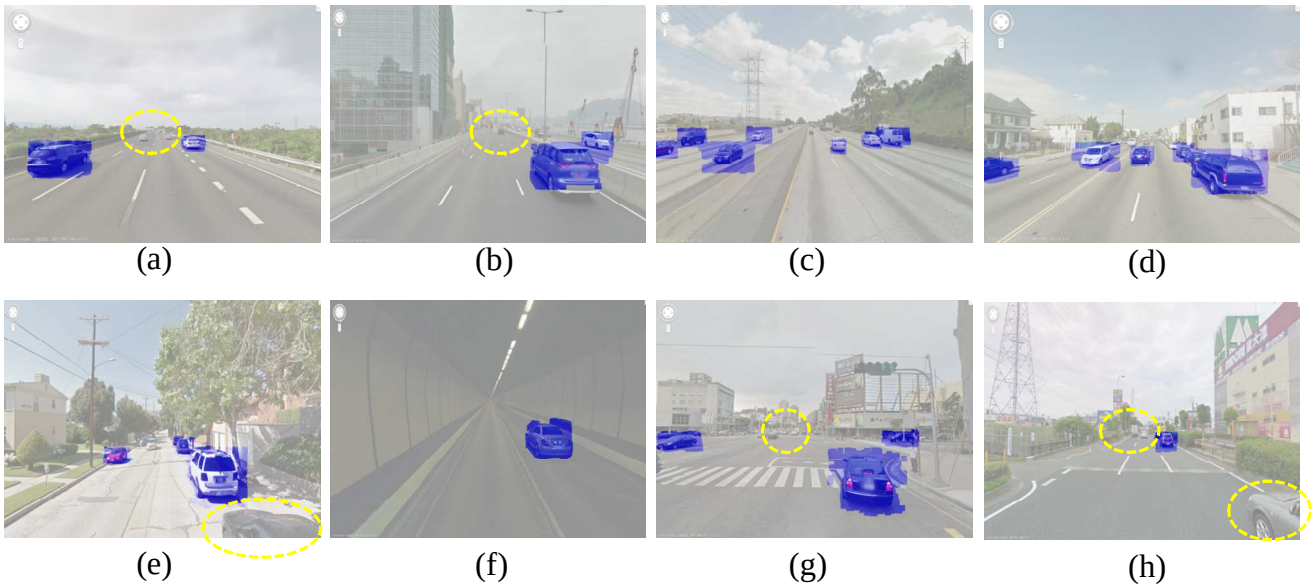

Figure 10. Sample car detection results. Sample results for Dataset 1 to Dataset 8 are shown from left to right, and top to down.

We quantitatively evaluate car detection and car segmentation in the following. Car regions in each image are first manually defined. A car detection result is claimed to be correct if more than 80% of the detected region overlap with the ground truth. For a dataset, the precision rate of car detection is thus calculated as the ratio of the number of correctly detected regions to the number of all detected regions. The recall rate is the ratio of the number of correctly detected regions to the number of regions that are truly cars. By jointly considering precision and recall, the F-measure is calculated as $\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$. Table 2 shows performance of car detection for each dataset. The trend of content-dependent performance is not surprising. Performance for images in the tunnel (the 6th dataset) is very poor due to dusky light. Overall, the cascade DPM has higher precision than recall. In street view images, car sizes and views change significantly, and it is often that only parts of cars are shown. This gives rise to great challenges in car detection. The average F-measure is 0.38, which is similar to the state-of-the-art car detection performance in the PASCAL VOC challenge.

To evaluate car segmentation, we inspect pixels on cars, and manually define the ground truth. The precision rate is the ratio of the number of correctly detected pixels to the number of all detected pixels. The recall rate is the ratio of the number of correctly detected pixels to the number of pixels that are truly on cars. Table 3 shows the car segmentation performance for each dataset. Limited by the results of car

detection, performance of car segmentation is just fair. From these results, we see that highly accurate automatic car detection in street view images is still not achievable by the state of the art. To fairly reflect performance of inpainting, the inpainting process starts from manually removed car regions in the following evaluation.

Table 2. Performance of car detection in terms of precision, recall, and F-measure.

	1	2	3	4	5	6	7	8	Avg.
Precision	0.73	0.65	0.88	0.84	0.70	0.07	0.82	0.91	0.70
Recall	0.36	0.15	0.32	0.36	0.28	0.07	0.45	0.12	0.26
F-measure	0.48	0.25	0.47	0.51	0.40	0.07	0.58	0.30	0.38

Table 3. Performance of car segmentation in terms of precision, recall, and F-measure.

	1	2	3	4	5	6	7	8	Avg.
Precision	0.63	0.46	0.60	0.64	0.52	0.10	0.57	0.53	0.51
Recall	0.63	0.33	0.48	0.59	0.41	0.13	0.57	0.08	0.40
F-measure	0.63	0.38	0.53	0.61	0.46	0.11	0.57	0.14	0.43

7.2 Results of Inpainting

To demonstrate performance of inpainting, we compare our proposed method with [5] and [6], which are the base methods we modify from, with [20], which is a typical exemplar-based inpainting method, and with [21], which is embedded in the OpenCV library and is especially computationally cheap. Figure 11 shows sample results of each dataset. The regions where we can clearly notice the difference between methods are marked as yellow dash circles. Generally, the method in [21] usually gives rise to large-scale blurred effects. This annoyance becomes more apparent when structured objects, such as central reservation and crash barrier, are reconstructed. The exemplar-based inpainting method in [20] obtains satisfactory performance for Datasets 2, 3, 6, and 7. However, this method may give rise to unnatural traffic marks because it does not specially consider line structure of road. Comparing with two base methods [5] and [6], our method clearly has superiority on reconstructing road structure, seeing especially from the results of Datasets 1, 4, and 8. Comparing sample results of [5] with [6] in the first dataset, we see that [5] works better in reconstructing lane lines, and [6] works better in reconstructing smooth regions. With the designed filling priority that considers road structure, and the combination of

hierarchical texture propagation [5] and randomized texture propagation [6], our method obtains much better inpainting performance in the applications of street view images.

The sample result for the fifth dataset reveals one of the limitations of our method, i.e., the reconstructed edge between the road and the sidewalk is unnatural (the left-bottom region). Because the proposed method reconstructs a missing region with existing patches from spatially neighboring images, the inpainting results would be unnatural if views in neighboring images change significantly.

7.3 User Study

We conduct a user study to verify usability of the proposed system. Twenty-seven subjects, all heavy computer users, were asked to evaluate inpainting results obtained by three systems: our method, the hierarchical texture propagation method [5], and the randomized texture propagation method [6]. Each subject was asked to evaluate in totally twenty runs, and thus totally 540 sets of inpainting results were compared. At each run, a street view image was randomly selected from the eight datasets, and three corresponding inpainting results were randomly juxtaposed so that subjects can easily compare and rank the three inpainting results into the best, the worst, and in-between.

Figure 12 shows the numbers of times an inpainting method is evaluated as the best, in-between, and the worst. From this figure, it is clear that our method and PatchMatch [6] generally obtain better results. Comparing ours with [6], 73.7% of the 540 evaluated images were ranked as the best or in-between for our method, and 64.6% were ranked so for PatchMatch [6].

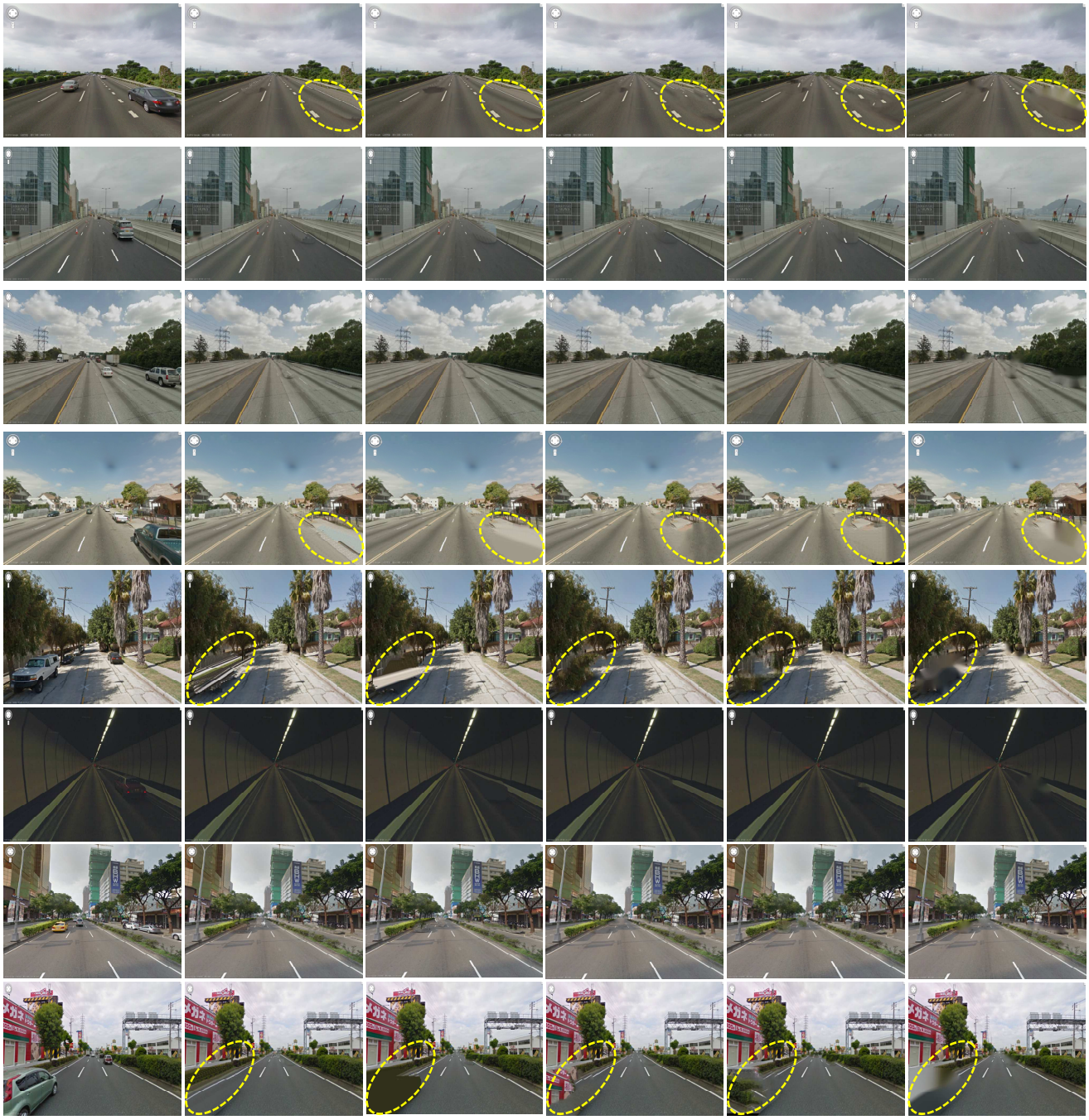

Figure 11. Sample inpainting results of the eight datasets, from top to down. (a) Original images; (b) Results of the proposed method; (c) Results of Le Meur [5]; (d) Results of PatchMatch [6]; (e)Results of Komodakis [20]; (f)Results of Telea [21].

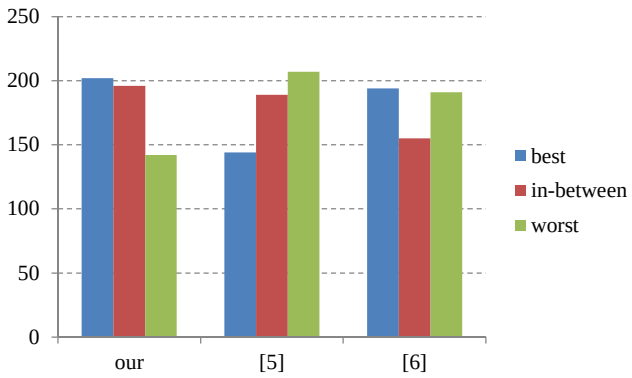

Figure 12. Subjective comparison between three inpainting methods.

Another question was asked when a subject finished all twenty runs: do you think these inpainting results provide more privacy protection than the original street view images? Not surprisingly, most subjects agree. The only one subject who did not agree thought car bodies would not leak much privacy, but riders on motorbikes would. Not removing motorbike riders really is the limitation of our current work.

Our method obtains more superiority over PatchMatch for Datasets 4 and 5. There are many parking cars at the roadside, and inpainting results of [5] and [6] are often unnatural. Thanks to propagating road structure from neighboring images, our method relatively gives better results. On the other hand, our method obtains less superiority over PatchMatch for Datasets 7 and 8. Discontinuous road structure diminishes the advantage of road structure propagation.

7.4 Complexity Issues

This system consists of the following major components: detecting cars by the cascade DPM, road detection and line detection, car segmentation by the GrabCut algorithm, road structure propagation, and hierarchical/randomized texture propagation (inpainting). Table 4 shows the average time needed by each component to process a street view image in eight datasets. Note that our program has not been optimized, so that we should pay more attention to the relative time complexity rather than the absolute execution time. From Table 4 we can clearly see that the computation bottleneck is inpainting, i.e., averagely sixty percentage of execution time is attributed to the inpainting process. Searching for appropriate patches (including patch searching and patch evaluation) for each missing patch takes much time. One possible direction to accelerate the inpainting process is to utilize parallel algorithms and hardware like GPU (graphics processing unit), so that many targeted patches can be evaluated in parallel.

Another direction is to relax the criterion to select an appropriate patch, making sort of sacrifice on the quality of inpainting results.

Table 4. Average time (in seconds) needed by each component to process a street view image in eight datasets.

ID	Car detection	Road detection	Car segmentation	Structure propagation	Inpainting	%Inpainting
1	1.11	3.99	2.06	6.25	30.93	69.76%
2	0.92	3.74	2.05	12.77	17.19	46.87%
3	1.04	3.63	2.08	8.76	22.06	58.73%
4	1.05	3.47	2.08	6.95	32.28	70.43%
5	0.91	15.04	2.23	4.14	40.20	64.30%
6	0.67	3.74	2.10	6.52	11.54	46.98%
7	0.98	4.35	2.16	5.31	23.31	64.55%
8	0.75	3.73	2.25	7.65	23.82	62.34%

7.5 Limitation

Because there are various road conditions and vehicle types in street view images of various viewing angles, the current system is not able to handle all possible cases. One of the most challenging cases is a downtown scene, where many cars move on the road or park at the roadside, and many buildings with various facades are located along the street. In Figure 13, we show four sample images captured at the same place but from different viewpoints[2], and these samples' processing results, as the auxiliary information to describe the limitations of this system.

- Limitation of road detection. The main idea of the algorithm depicted in Figure 3 is to determine the most dominant colors in the bottom halves of images. For the roads different from the straight roads, e.g., "intersection of roads", "streets in downtown", "roads with wider middle lane", and "non-straight roads", this algorithm can still successfully work if the color of road is really dominant in the bottom halves of images. The "roads with thicker shadows" or "roads jammed with cars" would be the most harmful case. It is often that the shade of trees or buildings around the road

[2] "Front," "Back," "Left," and "Right" mean street view images captured by the cameras facing forward, backward, leftward, and rightward, respectively.

largely covers the road region. The shade region would cause the dominant colors, and make our road detection algorithm fail. Figure 13(b) shows this challenging case where the bottom halves of the "Left" image and the "Right" image are more or less occupied by parking cars or buildings' shade. Results of road detection are thus not promising in these cases.

- Limitation of car detection. The cars that are very close to or very far from the camera are difficult to be detected. In addition, detecting cars on a jammed street is difficult, where preceding cars may be occluded by following cars. We found the general cascade DPM for "car" provided by [3] is able to detect cars with appropriate sizes in both frontal view and side view. Theoretically, we can construct a car model for each type of vehicle, like bus or concrete mixer lorry. However, we did not do this because there are relatively fewer such vehicles on the road, and using an existing general car model makes us more focus on the main tasks, i.e., car segmentation and inpainting. In the "Front" and "Back" images of Figure 13(c), cars parking at the roadside are often miss-detected because of occlusion. The motorcycles in the "Right" image are not detected because currently the motorcycle cascade DPM is not specially constructed.
- Limitation of inpainting. Currently the inpainting process is mainly designed to handle street view images in frontal views. For side-view images where façade of building occupies most space or cars are very close to the camera, cars may still be detected and removed, but road structure cannot be easily defined and extracted, and consequently the inpainting process without road structure propagation often obtains worse results. To only show the influence of road structure propagation on the inpainting results, for the "Left" and "Right" images in Figure 13, we manually remove the car/motorcycle regions and accomplish the inpainting process without road structure propagation to obtain the results shown in Figure 13(d). Overall, satisfactory inpainting results can still be obtained, except that structure of the boundary between road and sidewalk cannot be strongly reconstructed.

Comparing with inpainting results, the mosaic results shown in Figure 13(e) totally ignore scene structure, and we can clearly see the mosaic artifacts. To simply protect privacy, mosaicking car regions would be enough. However, completely removing cars and recovering with the inpainting technique simultaneously protect privacy and maintain visual completeness.

Figure 13. Sample images captured by four cameras at the same place and their corresponding processing results. (a) Original images; (b) road detection results; (c) car detection results; (d) inpainting results; (e) mosaic results.

8. CONCLUSION AND FUTURE WORKS

We have presented an automatic approach that detects and removes cars in street view images. Based on the cascade deformable part model, line detection, and road detection, the proposed system automatically determines foreground seeds and background seeds, which are later fed to the GrabCut algorithm to locate car regions. After removing cars, the missing regions are reconstructed by the proposed inpainting technique that especially considers characteristics of road to determine the filling order and to conduct preliminary road structure propagation. A hierarchical texture propagation method and a randomized

texture propagation method are integrated to smoothly reconstruct the road region and sharply reconstruct line structure in the meanwhile. The evaluation results show that our method especially works better in street view images.

The current work is limited by the following issues, which give the hints for developing future works:

- The DPM-based car detection module does not work very well for challenging street view images, and motorbikes, bikes, and other vehicles like buses are not considered in this work. More advanced vehicle detection methods are needed in the future.
- Google Street View images are actually 3D imageries. In this work, only images in a single view along a street are considered. More studies are needed to extend the current work to a 3D imagery obtained from Google Street View.
- Performance of car detection and inpainting for complex scenes with large-area shadows and traffic jams is not good now. As for inpainting, propagating information from neighboring images may not work in these cases.
- For a large-scale street view image collection, time consumption is a big issue. Parallel processing by the GPU programming model would be investigated in the future.

Acknowledgement

The work was partially supported by the National Science Council of Taiwan under the grants NSC101-2221-E-194-055-MY2.

9. REFERENCES

[1] A. Frome, G. Cheung, A. Adbulkader, M. Zennaro, B. Wu, A. Bissacco, H. Adam, H. Neven, and L. Vincent. 2009. Large-scale Privacy Protection in Google Street View. In *Proc. of ICCV*, pp. 2373-2380.

[2] J. Weir and W. Yan. 2010. Resolution Variant Visual Cryptography for Street View of Google Maps. In *Proc. of ISCAS*, pp. 1695-1698.

[3] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade Object Detection with Deformable Part Models. 2010. In *Proc. of CVPR*, pp. 2241-2248.

[4] C. Rother, V. Kolmogorov, and A. Blake. 2004. Grabcut: Interactive Foreground Extraction using Iterated Graph Cuts. *ACM Trans. on Graphics*, vol. 23, no. 3, pp. 309-314.

[5] O. Le Meur, J. Gautier, and C. Guillemot. 2011. Examplar-based Inpainting based on Local Geometry. In *Proc. of ICIP*, pp. 3401-3404.

[6] C. Barnes, E. Shechtman, A. Finkelestein, and D.B. Goldman. 2009. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Trans. on Graphics*, vol. 28, no. 3.

[7] L. Vincent. 2007. Taking Online Maps down to Street Level. *Computer*, vol. 40, no. 12, pp. 118-120.

[8] Y. Yoshimoto, T.H. Dang, A. Kimura, F. Shibata, and H. Tamura. 2011. Interaction Design of 2D/3D Map Navigation on Wall and Tabletop Displays. In *Proc. of ACM International Conference on Interactive Tabletops and Surfaces*, pp. 254-255.

[9] R. Guy and K. Truong. 2012. CrossingGuard: Exploring Information Content in Navigation Aids for Visually Impaired Pedestrians. In *Proc. of CHI*, pp. 405-414.

[10] J. Kopf, B. Chen, R. Szeliski, and M. Cohen. 2010. Street Slide: Browsing Street Level Imagery. *ACM Trans. on Graphics*, vol. 29, no. 4.

[11] A. Flores and S. Belongie. 2010. Removing Pedestrians from Google Street View Images. In *Proc. of CVPR*, pp. 53-58.

[12] D. Tschumperle and R. Deriche. 2005. Vector-Valued Image Regularization with PDEs: A Common Framework for Different Applications. *IEEE Trans. on PAMI*, vol. 27, no. 4, pp. 206-517.

[13] A. Criminisi, P. Perez, and K. Toyama. 2004. Region Filling and Object Removal by Examplar-based Image Inpainting. *IEEE Trans. on Image Processing*, vol. 13, no. 9, pp. 1200-1212.

[14] Y.Y. Boykov and M.-P. Jolly. 2001. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. In *Proc. of ICCV*, pp. 105-112.

[15] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. 2010. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Trans. on PAMI*, vol. 32, no. 9, pp. 1627-1645.

[16] N. Dalal and B. Triggs. 2005. Histograms of Oriented Gradients for Human Detection. In *Proc. of CVPR*, pp. 886-893.

[17] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. 2004. Lazy Snapping. In *Proc. of ACM SIGGRAPH*, pp. 303-308.

[18] K. He and J. Sun. 2012. Statistics of Patch Offsets for Image Completion. In *Proc. of ECCV*, pp. 16-29.

[19] S. Di Zenzo. 1986. A Note on the Gradient of a Multi-Image. *Computer Vision, Graphics, and Image Processing*, vol. 33, no. 1, pp. 116-125.

[20] N. Komodakis and G. Tziritas. 2007. Image Completion Using Efficient Belief Propagation via Priority Scheduling and Dynamic Pruning. *IEEE Trans. on Image Processing*, vol. 16, no. 11, pp. 2649-2661.

[21] A. Telea. 2004. An Image Inpainting Techniques based on the Fast Matching Method. *Journal of Graphics Tools*, vol. 9, no. 1, pp. 25-36.

[22] S. Agarwal, A. Awan, and D. Roth. 2004. Learning to Detect Objects in Images via a Sparse, Part-Based Representation. *IEEE Trans. on PAMI*, vol. 26, no. 11, pp. 1475-1490.

[23] R.N. Hota, K. Jonna, and P.R. Krishna. 2010. On-Road Vehicle Detection by Cascaded Classifiers. In *Proc. of the Third Annual ACM Bangalore Conference*.

[24] W. Zheng and L. Liang. 2009. Fast Car Detection Using Image Strip Features. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2703-2710.

[25] C.-H. Kuo and R. Nevatia. 2009. Robust Multi-View Car Detection using Unsupervised Sub-Categorization. In *Proc. of Workshop on Applications on Computer Vision*.

[26] J. Shi and J. Malik. 2000. Normalized Cuts and Image Segmentation. *IEEE Trans. on PAMI*, vol. 22, no. 8, pp. 888-905.

[27] P.F. Felzenszwalb and D.P. Huttenlocher. 2004. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, vol. 59, no. 2, 167-181.

[28] J.-B. Huang, S.B. King, N. Ahuja, and J. Kopf. 2014. Image Completion using Planar Structure Guidance. In *Proc. of ACM SIGGRAPH*, 2014.