Change Detection Tool based on GSV to help DNNs training

Carlos G. Huélamo[1], Pablo F. Alcantarilla[2], Luis M. Bergasa[1] and Elena López[1]

[1] Deparment of Electronics, University of Alcala, Spain,
carlos.gomezh@edu.uah.es luism.bergasa@uah.es elena.lopezg@uah.es
http://www.robesafe.es/personal/bergasa/
[2] SLAMcore Ltd, London, United Kingdom,
pablofdezalc@gmail.com,
http://www.robesafe.es/personal/pablo.alcantarilla/

Abstract. We present a system to carry out the automatic detection of structural changes through a Deconvolutional Neural Network (DNN) in images synthesized from panoramas provided by an online and open source map tool, Google Street View (GSV). Our approach is motivated by the need of more efficient and frequent updates on large-scale maps for autonomous driving applications. To train and evaluate our DNN we build a geolocation database, an order of magnitude larger than other existing datasets, based on pairs of images and their corresponding ground truth that shows changes detection over time. A tool has been implemented to guide manual annotation of changes using panoramas all over the world. The tool chains the panoramas and depth maps creation, the image synthesis and the labelling synthesized images generating their groundtruth. Finally, a DNN has been trained to automatically detect changes validating our methodology by using the obtained dataset, yielding better results that other state-of-the-art approaches.

Keywords: Google street view, Change detection, DNN, Keypoint, Training/Validation instances, Matching

1 Introduction

When viewed at the scale of cities and over periods spanning seasons or years, we realized that the urban visual landscape is a highly dynamic environment, with many navigational signs such as buildings, traffic signs and other structures located at both sides of the road being highly added or removed [1] [2]. From the point of view of an autonomous driving system, keeping an updated map of certain places of interest is essential. The higher the frequency with which maps are updated, the more robust the navigation system will be.

In this work we address the problem of efficient maintenance of a map by detecting structural changes using a powerful tool made available to the public, online and totally free such as Google Street View (GSV). The greatest advantage of using GSV is that structural changes over the years all over the world can be recovered by using the different laps of the Google Car in the same places. This information can be very interesting for historical, engineering, architectural (cadastres) or sociological analysis, among others. According to driving applications, keeping maps up-to-date is not only

useful for robust navigation, but can also provide benefits in monitoring the availability of parking spaces or road closures and deviations due to road works. Detecting structural changes in urban environments from GSV images is a challenging problem, as illustrated in Figure 1.

Fig. 1. Examples on images that present challenging changes generated from Google Street View, taken from our dataset. Left and right columns show a pair of images (older and newer ones), both being as aligned as possible.

Note that all examples in Figure 1 assume changes in luminosity, weather and station, and what makes it even more challenging is the point of view.

Images taken at different times show a great variability that can be induced by structural changes (construction/demolition of buildings/traffic signals, among others), but also due to nuisances generated by changes in point-of-view (the main problem in this work), environmental conditions (luminosity, weather or season) or dynamic changes (pedestrians, vehicles or vegetation). The main task of this work is the detection of structural changes. Therefore, in order to successfully distinguish among noises and real structural changes, the detection method must be able to model both, not-detecting the first ones and detecting the second ones.

So far, structural changes are manually detected on images, being a very time consuming task. For this purpose we present a real powerful tool able to process geolocations all over the world, getting as output pairs of images and its differential groundtruth (structural changes between pairs of aligned images) in order to serve as training/validation dataset for neural network implementations.

Based on the current success of Convolutional Neural Networks (CNNs) in different image processing task: image classification [3], semantic segmentation [4] and site recognition [5], our tool has been validated by using a deconvolutional deep architecture [6] in order to detect structural changes. This DNN, shown in Figure 2, takes as inputs a pair of aligned images of the same geolocation and returns a highlight of the structural changes present between both based on pixel-wise classification.

The main contributions of our paper are the following:

1. We propose a powerful tool that combines an on-line and open source map tool such as GSV and a self-code able to process geolocation databases, obtaining a dataset useful to train a Deconvolutional Neural Network in the task of detecting structural changes.

2. We provide a new dataset for the task of detecting urban changes that presents an order of magnitude larger than other existing datasets. In addition, it contains challenging landscapes, due to variations of light, weather or seasons.

3. We validate our tool and dataset by using an improved deep deconvolutional architecture compared to previous works [7] that significantly improves the results of change detection with hand-crafted descriptors [8][9][10] and a CNN-based approach [11] on the street view change detection task.

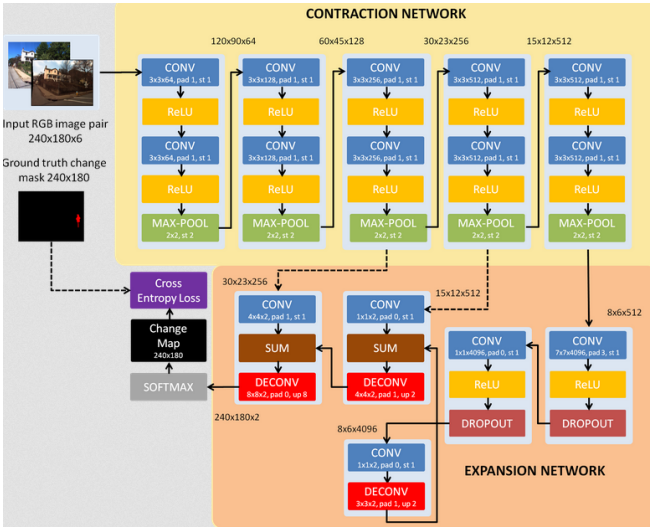


Fig. 2. DNN architecture adopted throughout this work.

2 Change detection tool architecture

Image-based change detection is a major problem in robotics and computer vision. This detection can increase the efficiency of maintaining a 3D map by updating only the areas of change [12] or allowing a system to learn about the nature of objects in the environment by segmenting while they are changing [13].

In order to train the DNN, our tool must generate training/validation instances, consisting in two aligned RGB images of the same place in different time and a groundtruth image with the structural differences between the two images. Flowchart 3 shows the architecture followed in our tool. As inputs, it takes two different in time links for the same geolocation. As output, it returns training packages constituted by pair of aligned images (from where the DNN will highlight the main structural changes perceived) and its differential groundtruth in order to validate the results.

2.1 Creating the URL database

The first step is to obtain two images for the same geolocation at different times. The conditions that should have the geolocation (both situations included) must take into account these requirements: Low or no presence of pedestrians, vehicles and vegetation, little difference in brightness and moderate structural changes in order to carry out the subsequent extraction of keypoints and alignment. These structural changes should not reach the stage where is impossible to realize if both images correspond to the same information. Figure 4 shows the GSV interface with the option of the time machine

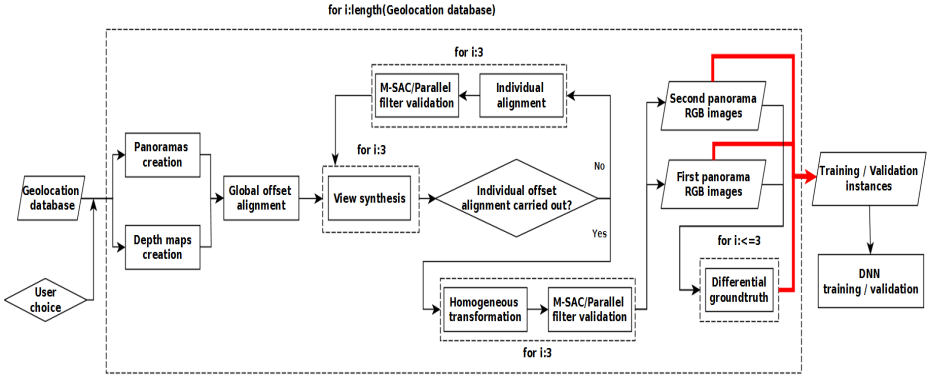


Fig. 3. Flowchart adopted in our tool.

framed in a red square. The user can obtain two different in time links for the same geolocation (eg: March 2015 and February 2017). After that, these links are stored in a geolocation database linked to our application. For that reason, if an user modify the database, the tool automatically will detect the change and operate in an updated way.

Notice that the more effective the search of geolocations (better coincidence in position and camera orientation), the more coincident will be the information represented in each pair of images, so the easier and more useful will be these images in the training process.

Fig. 4. GSV interface. Framed in a red box, represents the time machine that allows the user to navigate through different for the same geolocation.

2.2 Creating the panoramic image

Both panoramic image generation and its depth map are based on the three-dimensional reconstruction of cities from GSV [14]. GSV does not provide the user with a panoramic image, but thanks to these previous geolocation links we must perform a panorama reconstruction by using an API provided by GSV:

https://geo0.ggpht.com/cbk?cb_client=maps_sv.tactile&authuser=0&hl=en&panoid=%s&output=tile&x=%d&y=%d&zoom=3

The arguments to be introduced to the API are the panorama identifier (or panoid) and the horizontal (x) and vertical (y) position of the specific tile we are analyzing. By default, GSV divides the panorama into 21 tiles, grouped into 7 columns and 3 rows. As illustrated in Algorithm 1, introducing both panoid and a row and column identifier (ix, iy), rows and columns are traversed and each tile analyzed for a given geolocation. Notice that when representing the information of each tile into the panorama, the function Ipan(ix, iy, :) = im means that this portion of the panorama adopts the whole tile information (512 x 512 pixels) including the RGB channels. Finally, panorama dimensions are 1536 x 3584 pixels. Figure 5 shows an example of estimated panorama reconstruction, where the panoramic projection of the spherical image caught by GSV car be perfectly seen.

Algorithm 1 Computing the panorama in function of chosen resolution

Input: Panoid
Output: Estimated panorama reconstruction

1: for ix = 0 $\longrightarrow$ xtiles - 1
2: for iy = 0 $\longrightarrow$ ytiles - 1
3: $st\_url = sprintf('https://geo0.ggpht.com/cbk?cb\_client = maps\_sv.tactile\&authuser = 0\&hl = en\&panoid = \%s\&output = tile\&x = \%d\&y = \%d\&zoom = 3', panoid, ix, iy)$;
4: im = imread(st_url);
5: Tile composition:

$$ypos = iy \cdot tile\_height + 1 \longrightarrow (iy+1) \cdot tile\_height;$$
$$xpos = ix \cdot tile\_width + 1 - \longrightarrow (ix+1) \cdot tile\_width;$$

6: Ipan = (xpos,ypos,:) = im;

2.3 Computing the depth map

Now that we have the panoramic image, we need to retrieve its corresponding depth map. Google Maps REST API lets us to download a depth map JSON representation from the following link:

https://maps.google.com/cbk?output=json&cb_client=maps_sv&v=4&dm=1&pm=1&ph=1&hl=en&panoid=%s(3)

If we decompress the downloaded file it can be observed an alphanumeric file that contains the distance from the camera to the nearest surface for each pixel of the panoramic image. After decoding from Base64 data, transforming into UINT 8-bits array and then

parsing the raw data in order to obtain the header information (that contains the number of planes, width and height of the decompressed depth map). This header information is pretty useful since parsing again its values we realize that each pixel of this 512 x 256 grid corresponds to one of these planes (adjacent pixels with the same depth are contained in the same plane), each plane featured by its normal vector and minimum distance to the camera, respectively plane.n and plane.d in Algorithm 2. As mentioned above, the panorama is reconstructed by using spherical image caught by GSV. By European agreement, considering spherical coordinates, ϕ angle or azimuthal angle goes from 0 to 360 degrees (0 to 2π radians). On the other hand, θ angle or colatitude angle goes from 0 to 180 degrees (0 to π radians). Notice that an offset of $\frac{\pi}{2}$ is added to azimuthal angle in order to obtain a more realistic representation. The variable ur in Algorithm 2 represents the vector that joins a given pixel with the center of the camera, that is, the vector to transform our spherical coordinates into cartesian coordinates. To determine the depth of each pixel, it must be calculated the intersection of a ray that starts at the center camera [14] (whose vector is ur) and its corresponding plane. It must be taken into account that if planeIndex is equal to 0, the depth associated to that pixel will be maxDepth (input parameter). Otherwise, if planeIndex is higher than 0, the associated depth will be the absolute distance from the center of the camera to that pixel, represented by pixdistance, as illustrated in Algorithm 2. Iterating for all planes (all pixels per plane), we can compute our depth map as a bidimensional array of 512 x 256 elements each one 32 bits length. Figure 5 shows an example of depth map generation based on panoramic projection of the spheric image caught by GSV.

Algorithm 2 Computing the depth map

Input: Header, planes
Output: Estimated depth map

1: depth = zeros $(header \cdot height, header \cdot width)$
2: for h = 0 $\longrightarrow$ header.height - 1
3: for w = 0 $\longrightarrow$ header.width - 1
4: planeId $\longleftarrow$ $index[h \cdot header.width + w + 1]$;
5: $\phi \longleftarrow 2\pi \cdot \frac{header.width - w - 1}{header.width - 1} + \frac{\pi}{2}$;
6: $\theta \longleftarrow \pi \cdot \frac{header.height - h - 1}{header.height - 1}$;
7: $ur \longleftarrow [sin(\theta) \cdot cos(\phi), sin(\theta) \cdot sin(\phi), cos(\phi)]$
8: If planeId > 0 then

$$plane \longleftarrow planes[planeId];$$
$$t = \frac{plane \cdot d}{ur \cdot plane.n};$$
$$pix\_camera\_vector \longleftarrow v \cdot t;$$
$$pix\_camera\_distance \longleftarrow \sqrt{pix\_camera\_vector \cdot pix\_camera\_vector'};$$
$$depthMap[y \cdot w + (w - x - 1)] \longleftarrow pix\_camera\_distance;$$

9: else

$$depthMap[y \cdot w + (w - x - 1)] \longleftarrow maxDepth;$$

2.4 View synthesis

After elaborating the panoramic image and associated depth map a binary differential groundtruth could be generated in order to train the DNN [15]. However, previously images must be aligned. To do that we synthesize images from the panorama, using the

panoramic image and its depth map (Figure 5), because is easier to implement structural changes in smaller images and geolocation efficiency is a critical point.

This synthesis pretends to deal with the problem of visual shift and visual recognition of the place at large-scale where the scene suffers important changes, such as illumination, wear through time or explicit structural changes.

Planar structure of the depth map provides a really coarse 3D structure of the scene that will be represented in future synthetic views. Nevertheless, its precision is good enough for our purposes. The main objective of synthesizing both panoramas is to obtain pairs of images with approximately the same field of view so as to appreciate possible changes.

The flowchart adopted by our tool, shown in Figure 3, defines a feedback process in relation with the view synthesis process. First, an initial synthesis of the first panorama is carried out. Second, a global offset alignment is calculated and applied to the second panorama with the purpose of this second panorama with the same point-of-view of the first one. An individual extra offset is calculated, if required, based on matching techniques from each pair of images (from first and second panorama) in order to align them horizontally as best as possible. The synthesis process is explained in detail in paper [15]. For our purposes, a 5 degrees pitch angle and a field of view of 35 degrees are considered for our synthetic images from the center of the panorama. Notice that really these synthetic images do not exist but there are created by using GSV information. This is one of the highest contributions of this paper because our tool is able to generate different point-of-view images in an easy way. With the purpose of aligning

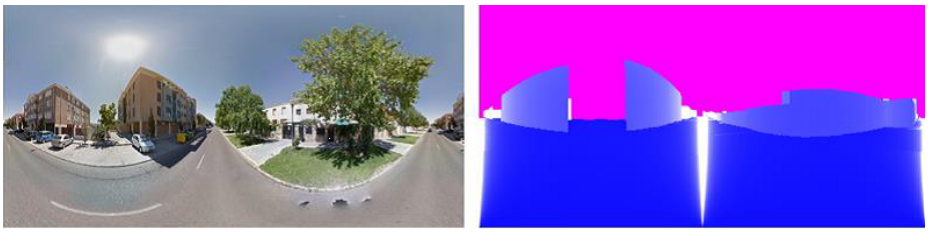


Fig. 5. Inputs for view synthesis. Left side, generated panoramic image. Right side, associated depth map.

both panoramas with a global offset, the tool is able to return the first panorama initial column from where the view synthesis is going to start, that is, zero degrees in our yaw vector, that represents the array that stores the view synthesis initial angles in spherical projection. In order to obtain a precise global offset alignment the tool stores the polyline manually drawn by the user, which must correspond to the same information that showed in the first initial column. An example can be observed in Figure 6. Global alignment is based in the following equation:

$$Global\_offset = \frac{mean(User\_line(:,1)) - mean(in\_col)}{ImWidth} \cdot 360$$

As mentioned, ImWidth is equal to the panorama width, in this case 3584 pixels. This

equation represents the equivalence in degrees between the horizontal offset presented in both panoramas and the projection of the panorama over a horizontal plane. If the global offset was negative (red column more to the right than blue column), it must be performed a positive anti-offset (adding the corresponding degrees to the heading angle of the camera). Otherwise, a negative anti-offset is carried out. Pitch angle and field of view are the same in both cases.

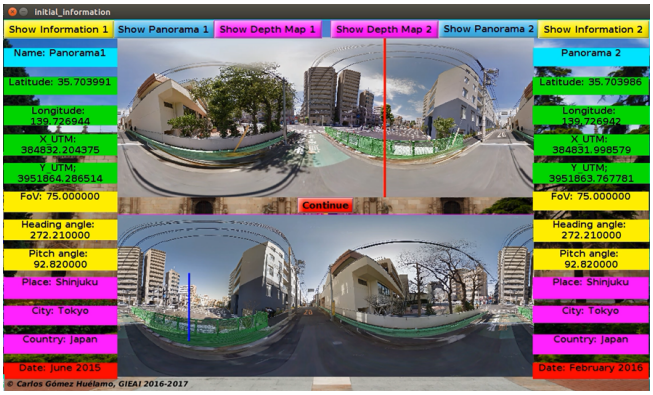


Fig. 6. Global alignment interface. It can be seen in red the first panorama initial column, calculated by the tool, and the second panorama initial column, drawn by the user.

2.5 Individual pair images alignment

Despite the view synthesis process is generated from globally aligned panoramas, it is not weird finding these camera points-of-view presenting strong displacements in terms of rotation and/or rotation. To ensure a fine alignment, it is convenient to individually align each pair of images by using matching techniques.

The previous global alignment is calculated determining the difference between initial columns. On the other hand, this additional individual pair images offset is calculated determining the arithmetic average among horizontal differences of theoretical good pairs of inliers (Algorithm 3) validated after the M-SAC and parallel filter (Algorithm 4).

With this double offset correction (global and individual offset) these pair of images are well aligned. However, in order to generate in an easier way the differential groundtruth, second panorama synthetic views are generated again, as illustrated in Flowchart 3, by using a homogeneous transformation to correct errors in terms of rotation or vertical translation. Notice that these homogeneous transformed images will not be introduced into the network but it will be useful in order to generate the differential groundtruth.

To carry out both alignments (individual and homogeneous transformation), some keypoints must be found in both images of each pair of images, matching those that correspond to the same information. Three keypoints detection techniques are applied, such as Harri's corner detector [16], SURF technique (Speed-Up Robust Feature) and SIFT technique (Scale-Invariant Feature Transform) [9]. Depending on three different techniques instead of only one provides us a greater flexibility facing condition

changes, such as rotation/translation of the point of view, different illumination, vegetation or season. After obtaining these required keypoints, a M-SAC (M-Estimator

Algorithm 3 Checking individual offset

Input: Validated pair of inliers, Image width
Output: Individual offset alignment

1: Sum of individual offsets:

$$\text{for i} = 0 \longrightarrow n = lenght(inliers\_pairs)$$
$$\text{num(i)} = \frac{second\_panorama\_inlier(i) - first\_panorama\_inlier(i)}{Image\_width} \cdot 360;$$
$$final\_num = final\_num + num(i);$$

2: $Individual\_offset\_alignment \longleftarrow offset\_angle = \frac{final\_num}{n}$;

SAmple and Consensus) filter is applied to remove the outliers. The main advantage of M-SAC filter versus the classic RANSAC (RANdom SAmple Consensus) technique [17] is that RANSAC technique can be sensible to the chosen threshold that determines which pair of keypoints is valid facing to the whole set of keypoints or not. For that reason, M-SAC variant is used based on evaluating the set of pairs quality calculating its probability. However, after checking some panoramas, we observed that after applying this M-SAC filter, just with two pair of erroneous keypoints outliers, the evaluated individual offset was really different from the real one and even after the second stage of this feedback (homogeneous transformation).

For that reason, we create a parallel filter based on reinforcing outliers discard, based on the Algorithm 4. It can be proved that if the second panorama section is located following the first one (both presenting 640 x 480 pixels, RGB images and groundtruth dimensions) can be considered a vector relating the second panorama section keypoint (end) and its counterpart in the first panorama section (origin). With elementary trigonometry the module and angle of this vector can be calculated.

Considering the module and angle of the initial vector characterized as our reference, a pair of keypoints will be validated if its module and angle is adjusted under established tolerance in relation with angle and module conditions. In order to validate the theoretical purged inliers after M-SAC filter, all keypoints vectors must be validated. Otherwise, even just one of evaluated inliers is defective, the combination among detection techniques (Harris, SURF and SIFT) and different homogeneous transformation applying the M-SAC filter (similarity, affine or projective) will do the inlier is automatically discarded. It can be summarized on a 3 (detection techniques) x 3 (homogeneous transformations) quality matrix. If the combination was discarded by the parallel filter, the position will be zero in the matrix. In any case, this matrix element will store the number of resulting pairs of inliers after applying M-SAC filter.

When the matrix is totally filled, it is ordered from highest to lowest number of inliers, considered in this work as the main quality parameter. The greater the number of matched pairs, the greater the accuracy of the estimated transformation. If exist four or more elements non-zero, there will be chosen the three highest values. A great checked matching example is observed in figure 7. Notice that the number of keypoints in the homogeneous transformation stage tends to be larger due to the extraction is carried out now with both images horizontally aligned as far as possible. Figure 8 shows an example of progressive alignment. First and second row show original first and second

synthesized panorama views. Third row shows second synthesized panorama view after horizontal alignment and fourth row shows second synthesized panorama views after applying homogeneous transformation when required. Be aware that middle column (second pair of images) has such amount of change in point-of-view that neither the global alignment offset nor individual alignment offset by using keypoints can correct the initial yaw angle, so this pair of images would be discarded.

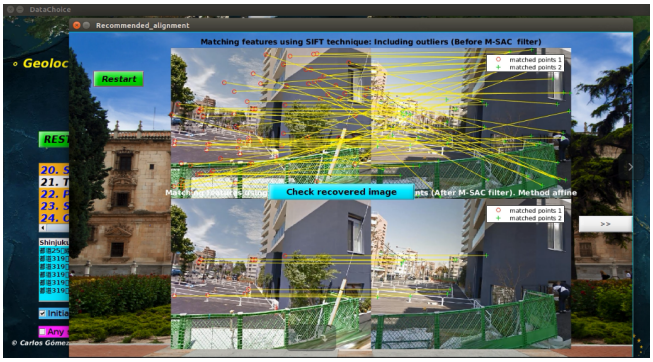


Fig. 7. Recommended alignment interface. It can be observed that results are completely coherent and successful.

Algorithm 4 Validation by using parallel filter

Input: Estimated keypoints pairs
Output: Alignment validation

1: Resulting keypoints pairs after M-SAC filter are taken.
2: Reference point calculation:

$$k(1)=(\text{second.keypoints.Location.x} + 640)-(\text{first.keypoints.Location.x});$$
$$l(1)=(\text{second.keypoints.Location.y} + 640)-(\text{first.keypoints.Location.y});$$

3: The first pair represents the reference absolute distance and reference angle:

$$module(1) = \sqrt{k(1)^2 + l(1)^2};$$
$$angle(1) = \frac{360}{2 \cdot \pi} \cdot \arcsin(\frac{l}{module(1)});$$

4: Alignment validation depending on manual set tolerances:

$$\text{for i=2} \longrightarrow \text{index(n)}$$
$$module(i) = \sqrt{k(i)^2 + l(i)^2};$$
$$\text{If (module(i) inside module tolerances then} \longrightarrow Calculate\_angle(i);$$
$$\text{If } (abs\_angle \text{ inside angle tolerances) then} \longrightarrow cont++;$$

2.6 Differential groundtruth

Labeling represents the last stage of our application. By using Liblabel Matlab tool created by Autonomous Vision Group of the University of Tübingen [18], we are able to label in an easy way structural changes that span from road repairements to facades, traffic signals or construction/demolition of new buildings. This lasts interface lets the user adding/removing a label or even generating the groundtruth at the same moment in order to check if a right semantic segmentation has been carried out.

Notice that in this work we will label in a mono-class segmentation way: There is structural change (taking into account its several classes) or not, not distinguishing in several classes which would correspond to a multi-class segmentation. Figure 9 shows main structural changes labeled throughout this work, being the three most representative classes: Repairements/maintenance (36 %), large structural changes (21 %) and combination among different classes (21 %). To a lesser extent, facades (10 %), non-presence of changes (9 %) and singular changes in traffic signals (3 %) are found. It can be observed that the percentege of repairements is bigger than others because there were more structural changes related to demolition/construction of buildings rather than traffic signals.

Fig. 8. Progressive alignment results performed by our application.

An important contribution in this work is the incorporation of a link between the tool and output database that contains the polygon coordinates of those pair of images from where the groundtruth has already been generated. Thus, if we try to label a pair of images that were previously labeled in order to improve the quality of the groundtruth or even labeling to a complex multi-class segmentation, last information remains unless it is removed, speeding up the process.

3 Validation results

So as to validate our proposed tool and the dataset obtained with it, we trained and validated a DNN proposed by one of the authors and we compare its performance with other works of the state-of-the-art. First, we divided the obtained instances dataset into a training set of 844 instances and a validation set of 233 instances. The split among sequences was chosen at random, with whole image descriptor matching used to confirm that test and training sets did not contain similar looking sequences.

In order to elaborate this training/validation instances dataset, 559 geolocation links where collected all over the world, then processed 354 of them by using the tool (as illustrated in Table 1) with the intention of generating a total of $354 \cdot 2 \cdot 3 = 2{,}124$ synthetic images, because each link represents the same place in different time. From each

place three synthetic views are obtained, as shown throughout this work. Notice that is much more convenient talking about $354 \cdot 3 = 1{,}062$ images pairs rather than 2,124 synthetic images because an image pair is the main element of each training/validation instance. Those image pairs that presented strong misalignment in terms of rotation and

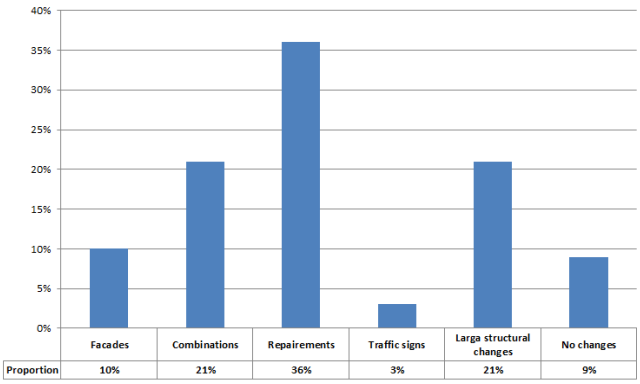


Fig. 9. Main structural changes detected. Most of them are repairements/maintenance tasks and secondly medium size structural changes.

translation, even after global and individual offset, were discarded in order not to introduce unnecessary noise to DNN. So, from a total of 1,062 theoretical training/validation instances we got 909 effective instances (as average, 4 minutes distributed in analyzing and labeling each pair) in order to train and test the DNN, obtaining an 85.59 % of efficiency. In the instances obtaining process our dataset presents an order of magnitude larger that previous databases of the state-of-the-art, as shown in Table 1.

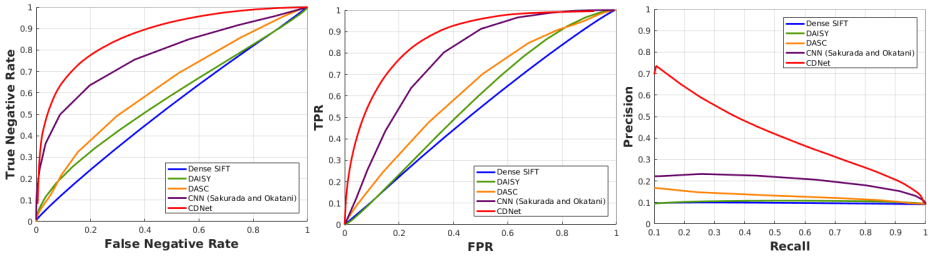
Table 1. Comparison of existing Street-View Change Detection Datasets

Dataset	**#Sequences**	**#Pairs**	**#Type**
Taneja et al. [12]	4	50	Perspective
Sakurada et al. [11]	23	92	Perspective
Sakurada and Okatani [19]	-	200	Panoramic
VL-CMU-CD [7]	152	1,362	Perspective
Current Dataset (Ours)	559 (354 used)	909	Perspective

In addition, with the purpose of helping prevent overfitting during training we used data augmentation by adding pairs of images containing both no changes of interest and artificial changes (by adding synthetic changes to existing images) [20]. This place recognition method has shown [21] impressive performance to recognize images from the same place under different weather and lighting conditions, taking into account that there are no large structural changes between both images, providing a pretty useful source of data augmentation for structural change detection systems.

These additional images were added in an approximate ratio of 25 % augmented changes and 75 % real changes, presenting the training dataset a total amount of 909 + 300 = 1,209 instances.

We compared our DNN called CDNet against multiple state-of-the-art techniques: Hand-crafted descriptors such as DAISY [10], DASC [8] (recently introduced as dense descriptor for multi-spectral and multi-modal correspondences) and dense SIFT [9]. Furthermore, our CDNet is compared to a pre-trained CNN for image recognition combined with superpixel regularization and sky segmentation [19].



(a) TNR vs FNR (b) TPR vs FPR (c) Recall vs Precision

Fig. 10. Validation test results after applying the architecture of the DNN to our dataset

Quantitative comparison: Figure 10 shows a comparison of the CDNet based on our instances obtaining process versus previous baseline methods. We show *True Negative Rate (TNR) or specificity* versus *False Negative Rate (FNR)*, *True Positive Rate (TPR) or Recall* versus *False Positive Rate (FPR)* and *Precision (Pr)* versus *Recall (Re)* graphs. These are defined as follows: TPR or Recall: TP/(TP+FN), FPR: TP/(TP+FN), Pr: TP/(TP+FN) and f1_Score: $2 \cdot \text{Pr} \cdot \text{Re}/(\text{Pr} + \text{Re})$.

Table 2. Quantitative comparison versus baseline methods at FPR = 0.10

Method	**Pr**	**Re**	**F1-Score**
Dense SIFT [9]	0.1133	0.1078	0.1105
DAISY [10]	0.1112	0.1063	0.1087
DASC [8]	0.1944	0.1799	0.1868
CNN (Sakurada and Otakani [19]	0.3671	0.3000	0.3302
CDNet (Ours)	**0.6078**	**0.5889**	**0.5982**

Table 2 compares our method over different change detection metrics for a FPR of 0.10. This same value was used in the qualitative comparison, as illustrated in Figure 11. Notice that our improved DNN architecture outperforms other methods on all metrics by a significant margin, being these best results highlighted in bold.

Qualitative comparison: Figure 11 shows qualitative results after applying our improved DNN to validate instances generated by the tool, with a FPR of 0.10. It is illustrated the predicted change maps of some methods for randomly selected validation image pairs. The performance of the DNN over these validation instances generated by the tool is quite better than other methods, both in detecting when there is structural change (from S1 to S5 row in Figure 11) and no detecting when there is no structural change (S6 and S7 in Figure 11), what shows both great performance of the network and great performance of the tool in the task of training and validating the DNN. Notice that changes due to vegetation, pedestrians or vehicles are not labeled.

4 Conclusions and future works

We have created a tool able to process links from Internet with GSV and generate aligned pair of images from the same geolocation but different time situations as well as a binary groundtruth able to detect structural differences between the pair of images. In addition, we create a new dataset in order to train and validate a DNN able to detect street-view structural changes. Our method combines panoramic images and depth maps generation, view synthesis, image alignment and manual labeling. Taking into account that the efficiency of our tool is 85.6 %, in relation to the searching and processing time of the dataset (2 months) results can be considered successful.

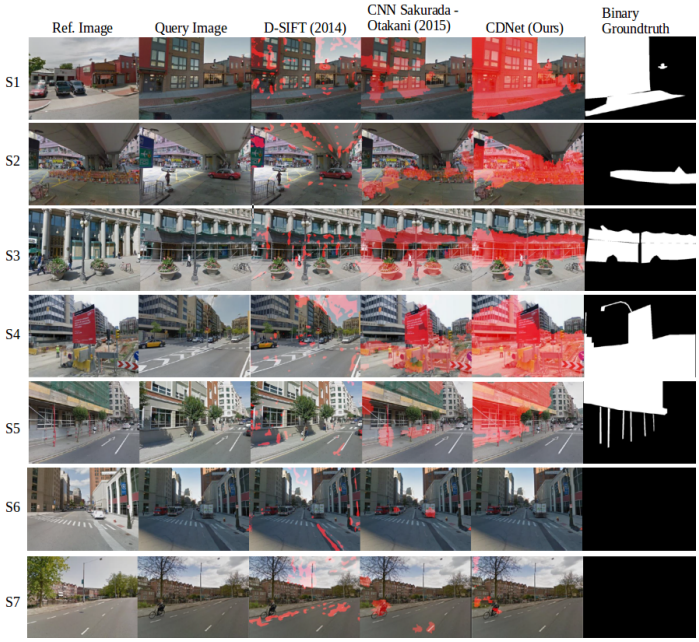


Fig. 11. Structural changes detection by using validation instances and different approaches. Changes detected are labelled in red.

Future improvements are the generation of our own depth map from the difference of pose between two panoramas of the same temporal situation and same geolocation, an improved camera pose correction in terms of translation in the synthesis process, the development of a better alignment process in order to increase the DNN labeling efficiency and the geolocation database to at least ten thousand of links.

5 Acknowledgment

This work has been partially funded by the Spanish MINECO/FEDER through the SmartElderlyCar project (TRA2015-70501-C2-1-R), the DGT through the SERMON project (SPIP2017-02305), and from the RoboCity2030-III-CM project (Robótica aplicada a la mejora de la calidad de vida de los ciudadanos, fase III; S2013/MIT-2748), funded by Programas de actividades I+D (CAM) and cofunded by EU Structural Funds.

References

1. R. Martin-Brualla, D. Gallup, and S. M. Seitz, "Time-lapse mining from internet photos," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 62, 2015.
2. K. Matzen and N. Snavely, "Scene chronology," in *European conference on computer vision*, pp. 615–630, Springer, 2014.
3. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.
4. H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1520–1528, 2015.
5. N. Sünderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, and M. Milford, "Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free," *Proceedings of Robotics: Science and Systems XII*, 2015.
6. G. Ros, S. Stent, P. F. Alcantarilla, and T. Watanabe, "Training constrained deconvolutional networks for road scene semantic segmentation," *arXiv preprint arXiv:1604.01545*, 2016.
7. P. F. Alcantarilla, S. Stent, G. Ros, R. Arroyo, and R. Gherardi, "Street-view change detection with deconvolutional networks.," in *Robotics: Science and Systems*, 2016.
8. S. Kim, D. Min, B. Ham, S. Ryu, M. N. Do, and K. Sohn, "Dasc: Dense adaptive self-correlation descriptor for multi-modal and multi-spectral correspondence," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2103–2112, 2015.
9. D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.
10. E. Tola, V. Lepetit, and P. Fua, "Daisy: An efficient dense descriptor applied to wide-baseline stereo," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 5, pp. 815–830, 2010.
11. K. Sakurada, T. Okatani, and K. Deguchi, "Detecting changes in 3d structure of a scene from multi-view images captured by a vehicle-mounted camera," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 137–144, IEEE, 2013.
12. A. Taneja, L. Ballan, and M. Pollefeys, "Image based detection of geometric changes in urban environments," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2336–2343, IEEE, 2011.
13. R. Finman, T. Whelan, M. Kaess, and J. J. Leonard, "Toward lifelong object segmentation from change detection in dense rgb-d maps," in *Mobile Robots (ECMR), 2013 European Conference on*, pp. 178–185, IEEE, 2013.
14. M. Cavallo, "3d city reconstruction from google street view," *Comput. Graph. J*, 2015.
15. A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla, "24/7 place recognition by view synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1808–1817, 2015.
16. C. Harris and M. Stephens, "A combined corner and edge detector.," in *Alvey vision conference*, vol. 15, pp. 10–5244, Citeseer, 1988.
17. H. Wang, D. Mirota, and G. D. Hager, "A generalized kernel consensus-based robust estimator," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 1, pp. 178–184, 2010.
18. A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3d traffic scene understanding from movable platforms," *Pattern Analysis and Machine Intelligence (PAMI)*, 2014.
19. K. Sakurada and T. Okatani, "Change detection from a street image pair using cnn features and superpixel segmentation.," in *BMVC*, pp. 61–1, 2015.

20. S. Stent, R. Gherardi, B. Stenger, and R. Cipolla, "Detecting change for multi-view, long-term surface inspection.," in *BMVC*, pp. 127–1, Citeseer, 2015.
21. R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, and E. Romera, "Fusion and binarization of cnn features for robust topological localization across seasons," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 4656–4663, IEEE, 2016.