# Large databases of real and synthetic images for feature evaluation and prediction

by

Biliana K. Kaneva

B.A., Computer Science and Mathematics, Smith College (2000)
M.S., Computer Science, University of Washington (2005)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

February 2012





Author: _____
Department of Electrical Engineering and Computer Science
December 22, 2011

Certified by: _____
William T. Freeman, Professor of Computer Science
Thesis Supervisor

Certified by: _____
Antonio Torralba, Associate Professor of Computer Science
Thesis Supervisor

Accepted by: _____
Leslie A. Kolodziejski, Professor of Electrical Engineering
Chair, Department Committee on Graduate Students

# Large databases of real and synthetic images for feature evaluation and prediction

by Biliana K. Kaneva

Submitted to the Department of Electrical Engineering and Computer Science
on December 22, 2011, in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

## Abstract

Image features are widely used in computer vision applications from stereo matching to panorama stitching to object and scene recognition. They exploit image regularities to capture structure in images both locally, using a patch around an interest point, and globally, over the entire image. Image features need to be distinctive and robust toward variations in scene content, camera viewpoint and illumination conditions. Common tasks are matching local features across images and finding semantically meaningful matches amongst a large set of images. If there is enough structure or regularity in the images, we should be able not only to find good matches but also to predict parts of the objects or the scene that were not directly captured by the camera. One of the difficulties in evaluating the performance of image features in both the prediction and matching tasks is the availability of ground truth data. In this dissertation, we take two different approaches.

First, we propose using a photorealistic virtual world for evaluating local feature descriptors and learning new feature detectors. Acquiring ground truth data and, in particular pixel to pixel correspondences between images, in complex 3D scenes under different viewpoint and illumination conditions in a controlled way is nearly impossible in a real world setting. Instead, we use a high-resolution 3D model of a city to gain complete and repeatable control of the environment. We calibrate our virtual world evaluations by comparing against feature rankings made from photographic data of the same subject matter (the Statue of Liberty). We then use our virtual world to study the effects on descriptor performance of controlled changes in viewpoint and illumination. We further employ machine learning techniques to train a model that would recognize visually rich interest points and optimize the performance of a given descriptor.

In the latter part of the thesis, we take advantage of the large amounts of image data available on the Internet to explore the regularities in outdoor scenes and, more specifically, the matching and prediction tasks in street level images. Generally, people

are very adept at predicting what they might encounter as they navigate through the world. They use all of their prior experience to make such predictions even when placed in unfamiliar environment. We propose a system that can predict what lies just beyond the boundaries of the image using a large photo collection of images of the same class, but *not* from the same location in the real world. We evaluate the performance of the system using different global or quantized densely extracted local features. We demonstrate how to build seamless transitions between the query and prediction images, thus creating a photorealistic virtual space from real world images.

---

Thesis Supervisor: William T. Freeman
Title: Professor of Computer Science

Thesis Supervisor: Antonio Torralba
Title: Associate Professor of Computer Science

# Acknowledgments

This work would not have been possible without the guidance and support of my advisors, professor William T. Freeman and professor Antonio Torralba. I have learned an immense amount from them about research, selecting interesting problems and having the courage and curiosity to tread unexplored areas. Their creativity, generosity and enthusiasm for research and discovery have been a great inspiration to me. I am very grateful to have had the opportunity to work with them.

I would also like to thank my collaborators Josef Sivic and Shai Avidan. They have a great part in this work. Shai has taught me how to write papers and think about problems. Josef's contagious enthusiasm, curiosity and encouragement brought much fun to the research process even during challenging moments. It has been a great pleasure to work with them.

MIT is a unique environment for research with unparalleled resources. I am grateful for the fun and productive atmosphere in the Computer Vision and Graphics groups and am fortunate to have met many incredible people here over the years. They have enriched my life in myriad ways.

Finally, I thank all my friends for their continuous support and faith in me. Most of all, I am eternally grateful to my parents for the many sacrifices that allowed me to have this experience and to my husband, Renaud, for always being there.

---

This work was partially funded by a Xerox Fellowship, Shell Research, Quanta Computer, ONR-MURI Grant N00014-06-1-0734, CAREER Award No. 0747120, ONR MURI N000141010933 and by gifts from Microsoft, Adobe, and Google.

# Contents

# List of Figures

# Chapter 1

# Introduction

THE default representation of visual information in images is usually in the form of a matrix of pixels where each pixel is a quantized value of the brightness of the color at a given location. While this has proven to be a useful representation for humans, it is not necessarily the optimal representation for a computer system. Humans rely on their visual system to process the information in the image in visual recognition tasks. Recently, Parikh and Zitnick compared human performance in visual recognition tasks to that of state-of-the-art computer vision algorithms [72]. Their findings attributed the superiority of human performance in those tasks to the better use and selection of image features. This underscores the need for developing methods for detection and description of points or regions in the image that contain visually rich information. More generally, it calls for understanding how regular the real world is and to what extent we can capture and exploit the structure in images due to these regularities.

There is a significant body of work in the literature dating back to the 1950s [4], focused on the detection of points or regions with high information content and the development of new representations or descriptions of those regions to facilitate different computer vision tasks ranging from wide baseline stereo matching [76, 107] to object recognition [28, 33, 34, 59]. Often evaluation of local image features is done with respect to a specific task; however, Schmid et al. argued that that is limiting and

difficult to generalize [86]. In their work [67, 86], they propose criteria for the evaluation of feature detectors and local feature descriptors including a dataset of images under different transformations. Since then, there has been other work comparing the different detectors and descriptors present in the literature [62, 68, 108, 117, 118]. One of the limitations in both evaluation and design of image features has been in acquiring ground truth labeled data and, in particular, pixel to pixel correspondences between images for complex 3D scenes under different viewpoint and illumination conditions in a controlled way. The difficulty in acquiring such data in a real world setting stems from uncontrolled changes in the scene content and the presence of occlusions, specularities, cast shadows, etc. Hence there is a need for a method for gathering ground truth labeled data that captures these phenomena by providing a complete and repeatable control of the environment.

Capturing only the local structure in images provides us with a limited amount of information; in particular, it fails to take advantage of the context in which these interest points or regions appear. This is particularly important in tasks such as scene recognition [54, 71], label transfer [11, 58], and image editing or enhancement [25, 44]. Here, we want to capture the global structure of the image and/or the relationship of local regions with respect to each other in order to find semantically meaningful matches in a large database of images. There is a need to understand how many images are required and what features are best suitable for these tasks. Furthermore, we would want to know if there is enough structure in the images to not only match images on a semantic level but also predict parts of the scene that were not directly captured by the camera.

In this thesis, we explore the use of large real world and synthetic datasets for feature evaluation, learning and prediction. We present 1) a highly photorealistic dataset from a virtual city with images taken under different lighting and viewpoint conditions and 2) a comparison of the effects changes in viewpoint and illumination have on the

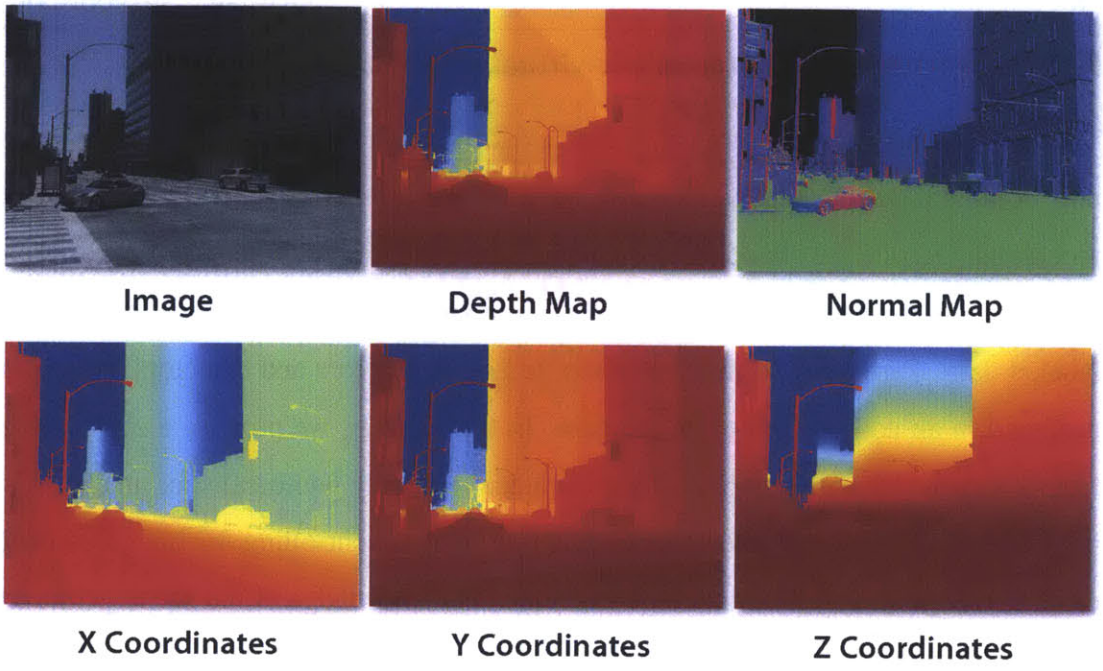



**Figure 1.1:** *Images showing the data we collect for each scene.*

performance of images descriptors. We propose 3) a method for finding good interest points based on our synthetically generated ground truth data and training models for feature detectors using machine learning techniques and a system 4) for predicting what lies beyond the image boundaries using large collections of images from the Internet and 5) for building a photorealistic virtual world by synthesizing new images based on those predictions.

## 1.1 Overview of contributions

In this section, we present an overview of the contributions in this dissertation.

### 1.1.1 Chapter 3: Photorealistic virtual world dataset

The readily available large amount of image data on the Internet has given rise to many new computer vision applications and have allowed researchers to study the world around us. Despite the abundance of images, it is quite difficult to control for different parameters in the data, e.g. lighting and camera viewpoint, and to obtain ground truth labeled data. Partly due to this limitation, image descriptors and detectors have been designed in an ad-hoc fashion in the past. In this chapter, we propose a new dataset from a virtual city with synthetically generated, yet highly photorealistic, images. Using a 3D graphics model allows us to generate images controlling for image content, camera viewpoint and illumination conditions. The knowledge of the scene geometry provides us with ground truth data such as depth maps, normal maps, pixel to pixel correspondences of 3D points between images (see fig. 1.1). We use the dataset to study low-level regularities in images taken under different lighting and viewpoint.

### 1.1.2 Chapter 4: Evaluation of image descriptors

In the past, image descriptors have been evaluated on images typically lacking in changes in perspective [67] or in the ability to control for changes in lighting and camera viewpoint [117]. In this chapter, we use the dataset proposed in chapter 3 to study the effects of changes in illumination and camera viewpoint on the performance of image descriptors. As a first step, we compare the relative performance of the descriptors on synthetic vs. real data. To control for content, we use a dataset of synthetically generated images of the Statue of Liberty described in chapter 3 and the Liberty dataset proposed by [117]. We then use the virtual city dataset to evaluate the performance of the image descriptors with respect to changes in viewpoint and illumination. We also propose augmenting the descriptors in the presence of depth information to further improve their performance.

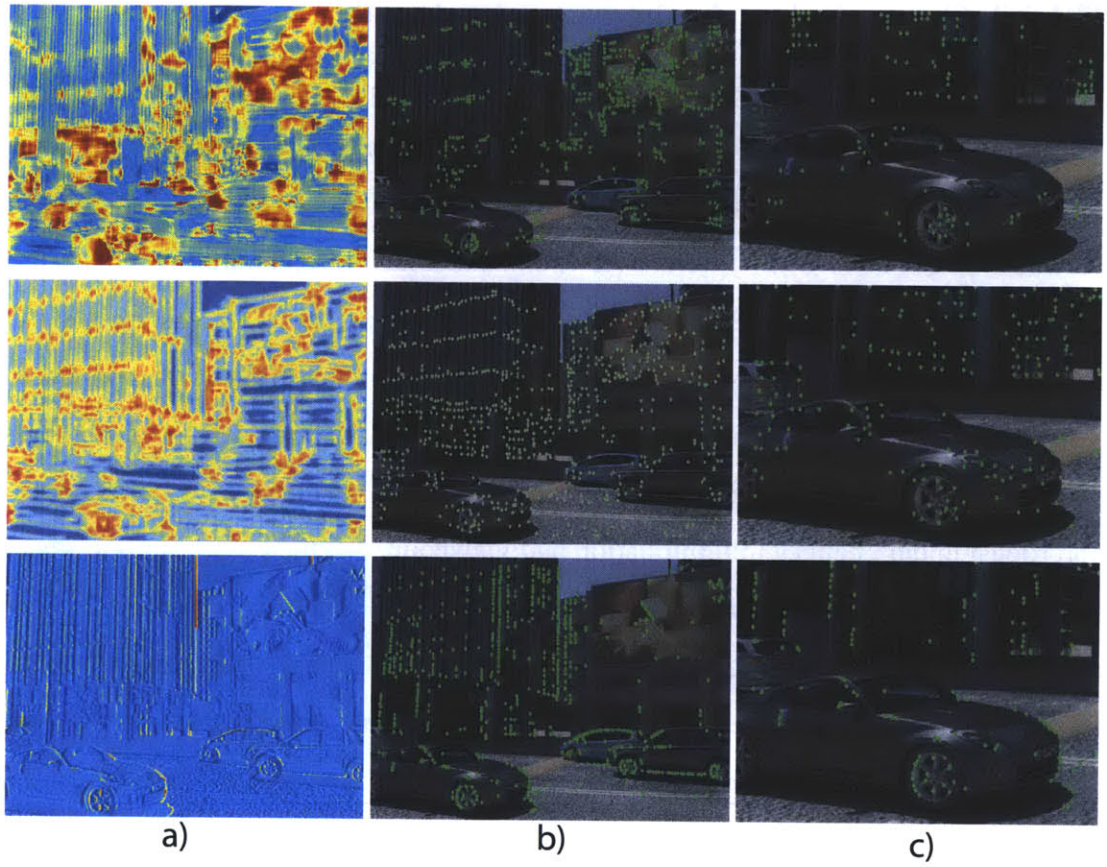



**Figure 1.2: Top row:** *a) The standard score map for good points based on a pair of images of the same scene taken under different illumination. The color map is from blue to red and indicates low to high information content respectively. b) 700 points selected based on the score map in a). Note that smooth regions, e.g. the sky and road have no points. The regions around the cast shadows from the building and the cars also do not contain points since they cannot be matched in images under different illumination. Most of the selected points are of textured or corner like regions. c) A close up of the image in b).* **Middle row:** *a) The classifier score map for the same image based on the model using the SIFT descriptor features. Note that it is fairly similar to the standard score map above. b) 700 points detected by the model. The distribution of the points is more uniform but with the exception of detecting some points on the road surface, it finds points in similar regions as the points in a). c) A close up of the image in b).* **Bottom row:** *a) The classifier score map for the same image based on the model using simple features. b) 700 points detected by the model. It favors points on edges. c) A close up of the image in b).*

### 1.1.3 Chapter 5: Learning interest point detectors

In addition to evaluating already existing computer vision algorithms, we can use the virtual city dataset proposed in chapter 3 to design new ones. In this chapter, we leverage the availability of ground truth data, in particular pixel to pixel correspondences between images taken from different viewpoint and under different illumination to propose new feature detectors. We present a method that finds good interest points with respect to a given feature descriptor using the available ground truth correspondences. This allows us to automatically generate labeled data that can be used to train interest point detectors. Using the SIFT descriptor, we find points that would be distinctive and robust under changes in illumination and viewpoint. We then use a support vector machine to learn models for feature detectors under those conditions. Figure 1.2 shows examples of the selected ground truth interest points and the points found by one of the learned models. We provide a quantitative evaluation of the learned models and compare them to two of the popular feature detectors.

### 1.1.4 Chapter 6: Looking beyond image boundaries

In the previous chapters, we focused on low level features, here we study the high-level regularities in images for predicting parts of the scene not captured by the camera. We present a system for predicting what lies beyond the boundaries of an image given a particular camera motion. We use a set of features as a compact and coarse representation of the scene in each image. We study the performance of different image features for both the scene matching and prediction tasks. A key component to our system is that we use images of the same category but not of the same place in the real world. We use city images to evaluate the performance of the system which allows us to study how regular street images are and how well these regularities are captured by the different image features. The images in our database were gathered from photosharing websites

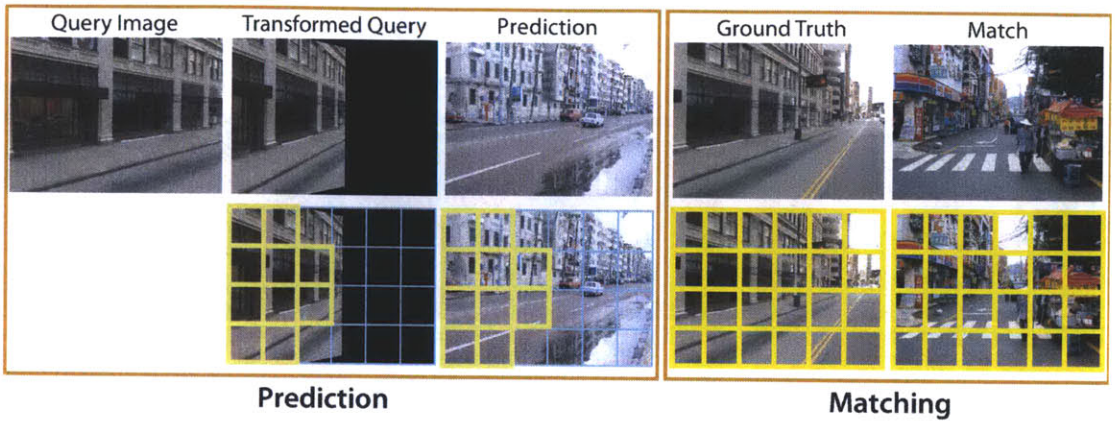



**Figure 1.3: Prediction and matching.** *Top row: Query Image. Transformed query image for rotate right motion. The prediction obtained by finding the nearest neighbor using the transformed query (the prediction task). Ground truth (the actual view seen after the camera rotated from the query image). Top match to the ground truth image (the matching task). Bottom row: The transformed query image and the top prediction image retrieved with the bins used for retrieval overlaid. The ground truth and the top match images and the bins used for the retrieval overlaid.*

such as Flickr while the ground truth dataset used for evaluation was synthesized from Google's Street View data from Pittsburgh. We used a database of 100,000 city street images for our experiments.

The scene matching task uses traditional image retrieval based on exhaustive nearest neighbor search. For the prediction task, we introduce the notion of *transformed image retrieval*. We first apply geometric transform to the query image using 2D approximations for the 3D camera motions (move left or right, rotate left or right, move forward or back). Then we use only the valid portion of the image descriptor to query the database for matches. Figure 1.3 shows the matching and prediction tasks.

### ■ 1.1.5 Chapter 7: Infinite images: The image graph

Finally, we exploit the high level regularities, we found in the previous chapter, in a computer graphics application. We use the transformed image retrieval method pro-


**Figure 1.4:** *Scene matching with camera view transformations. First row: The input image with the desired camera view overlaid in green. Second row: The synthesized view from the new camera. The goal is to find images which can fill-in the unseen portion of the image (shown in black) while matching the visible portion. The third row shows the top matching image found in the dataset of street scenes for each motion. The fourth row illustrates the induced camera motion between the two pictures. The final row shows the composite after Poisson blending.*

posed in chapter 6 to build a large photorealistic virtual space from a large collection of images of the same semantic category but not of the same place in the real world. The retrieved image from the prediction task can be used to extend the query image based on the given camera motion. We represent the virtual world as an image graph where the nodes in the graph are images and the edges represent the transitions between images based on a given camera motion. To create seamless transitions between the query and retrieved images as shown in figure 1.4, we first align the images, then find the optimal seam for stitching and finally blend them using Poisson blending in the gradient domain. We propose several different applications that can be built on top of the image graph - from creating infinite images to traveling between images using "image taxi".

# Chapter 2

# Background

THE need for better representation of visual data than raw image pixels can provide has encouraged a large body of research. For many applications the high dimensionality of pixel data is prohibitively expensive and difficult to generalize. In object recognition, we would like some way to compare image data within an image or across images to find examples of similar objects whether they are of the same instance or of the same category. In stereo matching, we would like to match points between stereo pairs in order to infer relative depth between objects in the scene or real depth if we know the camera calibration parameters. The pixel intensities describing an object can be affected by changes in lighting, camera viewpoint, object position and orientation; thus directly comparing pixel intensities is not desirable since it will often fail to correctly match similar objects and points. To reduce the dimensionality of the visual data and to improve robustness to changes in lighting and viewpoint, many different interest points or region detectors have been proposed. In section 2.1, we review several of the more popular feature detectors. Once interest points or regions have been localized, it is necessary to provide a compact but distinctive descriptor to allow for accurate matching of features across images. There have been many descriptors proposed though most focus on orientation histograms; the most widely used one is SIFT. We give brief descriptions of different feature descriptors in section 2.2. The last important part in enabling many vision applications is finding correspondences among features

and there have been many algorithms proposed for finding good matches as discussed in section 2.3.

## 2.1 Feature detectors

The task of a feature detector is to find distinctive points in an image that can be matched under different transformations or different instances within an object category. Those are usually found in textured regions or along object boundaries. Some of the most popular interest points are corners since they provide repeatable and informative features. Points or patches in solid color regions or along edges are difficult to distinguish because there can be many other points/patches that look exactly the same locally. Various feature detectors have been proposed in the literature. Here we offer an overview of the most widely used ones.

### 2.1.1 Corner detectors

Various corner detectors have been proposed in the past and many of them are built based on the *second moment matrix* or the *auto-correlation matrix*.

The matrix is defined as:

$$A(x) = g(\sigma_I) * \begin{bmatrix} I_x^2(x) & I_x I_y(x) \\ I_x I_y(x) & I_y^2(x) \end{bmatrix} \tag{2.1}$$

where the Gaussian function

$$g(\sigma_I) = \frac{1}{2\pi\sigma_I^2} e^{-\frac{x^2+y^2}{2\sigma_I^2}} \tag{2.2}$$

is used as a weighting function and $I_x$ and $I_y$ are the local image derivatives in the $x$ and $y$ direction, respectively.

#### Harris Detector and Extensions

One of the most popular corner detectors is the Harris detector and it uses the *auto-correlation matrix* defined in equation 2.1 to detect local image structures. Harris pointed out that the matrix $A$ will have two large eigenvalues at locations where there is a corner [43]. Instead of computing the eigenvalues of the matrix which can be computationally expensive, Harris proposed a single measure of the two eigenvalues or the cornerness of the location as:

$$C = det(A) - \alpha trace(A)^2. \tag{2.3}$$

The Harris corner detector has been shown to be very repeatable and informative in detecting interest points [86]. The Harris detector has been extended by [67] to be both scale invariant, Harris-Laplace, and affine invariant, Harris-Affine.

The Harris detector has been used in variety of vision applications that need to localize distinctive points.

#### Shi and Tomasi's detector

In their work on good features to track, Shi and Tomasi proposed to use the eigenvalues of the auto-correlation matrix $A$ [91]. They show that good features to track are those that have significant eigenvalues and unlike Harris, they compute the eigenvalues explicitly. Thus the cornerness measure is defined as:

$$C = min\{\lambda_1, \lambda_2\}, \tag{2.4}$$

where $\lambda_1, \lambda_2$ are the eigenvalues of $A$.

#### Forstner's detector

In the work of Förstner and Gülch [37], they use the auto-correlation matrix $A$ to locate interest points by classifying the pixels into regions, contours, junctions and circular features. The cornerness measure is defined as:

$$C = \frac{det(A)}{trace(A)}. \tag{2.5}$$

Another corner detector proposed more recently is the SUSAN detector, though it often detects features along edges, rather than corners.

#### SUSAN Detector

The SUSAN (Smallest Univalue Segment Assimilating Nucleus) detector is a corner detector employing a morphological approach to corner and edge detection and noise suppression [94]. The pixels in a circular neighborhood around a central point are partitioned with respect to their similarity in intensity to the center point (nucleus). Thus, the relative size of the area with a similar brightness provides information about the structure around a given point in the image. To increase robustness, pixels closer to the nucleus are given higher weights. The features detected by the SUSAN detector are mostly along edges, rather than corners, and can be sensitive to noise.

Corners are only one type of repeatable and distinctive interest points. Blob detectors represent another important category of feature detectors whose goal is to find salient points/regions in the image that are either darker or brighter than the surrounding area.

## 2.1.2 Blob detectors

### Hessian Detector and Extensions

The Hessian detector is considered a blob detector and uses the auto-correlation or (Hessian) matrix of the image $I(x)$. The determinant and trace of the Hessian matrix can be used to find interest points at their local maxima. The points detected using the determinant are the ones that belong to a blob of similar size to the size of the Gaussian kernel $\sigma_I$. Using the determinant of the Hessian as an interest measure was proposed by [8]. The interest points found using the trace of the Hessian matrix may be unstable, in particular if they are lie on a contour or an edge. One approach to increasing their stability is to find the points where both the trace and the determinant of the Hessian have a local extremum.

Similarly to the Harris detector, the Hessian detector has been extended by [67] to be both scale invariant, Hessian-Laplace, and affine invariant, Hessian-Affine. The Hessian detector is not invariant to perspective transformation. To address this limitation, Maximally Stable Extremal Regions (MSER) were proposed.

### DoG: Difference of Gaussians

The Difference of Gaussian (DoG) detector approximates the Laplacian (or the trace of the Hessian matrix) thus extracting scale-invariant blob-like features [59, 60]. The DoG detector is very efficient as it avoids computing the second-order image derivatives in the $x$ and $y$ directions, instead it approximates the Laplacian by computing the difference between smoothed versions of the image at difference scales. To further improve the stability of the features, the eigenvalues of the full Hessian matrix are computed to discriminate amongst points along edges where the Laplacian produces a strong response.

#### Superpixels

Superpixels were first proposed as local features by [69] and [79]. These are the image patches resulting from over-segmenting the image. It has been argued that superpixels are a more meaningful representation than the original image pixels. Unlike other local image features, the superpixels do not overlap and cover the entire image. In general, superpixels are not invariant to scale since they are extracted at a similar scale. Superpixels are not as suitable for object recognition or stereo correspondence because of the uniform segments; these lack distinctiveness and have poor repeatability around boundary regions. They have primarily been used in region grouping or semantic region segmentation.

Many vision applications require a large number of features to be detected, usually hundreds or thousands. Thus, it is necessary that these detectors are highly performant. This need increases further as the image database size grows into the hundred thousands or millions. Several algorithms have been proposed for detecting feature points/regions based on approximately computing either corners based on the SUSAN detector or blobs based on the Hessian matrix.

#### SURF: Speeded Up Robust Features

The SURF (Speeded Up Robust Features) detector is a scale-invariant detector that uses an approximation of the the Hessian matrix [7]. The determinant of the Hessian matrix is used for selecting both the scale and the location of the interest points. The second-order Gaussian image derivatives are approximated using box filters and computed very efficiently using integral images. Integral images (or summed area tables), originally proposed as an efficient texture mapping technique [23], were introduced in computer vision by Viola and Jones [113] in the context of real-time face detection. Here, they produce a significant speedup over an already efficient detector like the DoG while

suffering from little to no impact on performance due to the approximation.

### 2.1.3 Region detectors

#### MSER: Maximally Stable Extremal Regions

The Maximally Stable Extremal Regions (MSER) were first used in the context of wide baseline stereo matching [65]. The extremal regions are computed based on thresholding the intensity of the image at different levels. Each region is a connected component over the image at a given threshold. The regions are invariant to affine transformations of the intensities, stable, and can detect both fine and coarse image structure. The MSER are selected at image areas that are local minima of the relative change of the area with respect to the change in the threshold; these are the parts of the image that are stable over a large range of thresholds.

The MSER detector is only based on the image intensity and does not take into account color information. An extension the algorithm to include color information was proposed in [36].

## 2.2 Feature descriptors

An essential task in many vision applications is to be able to match interest points accurately. Wide baseline matching can work with only a sparse set of correspondences, while applications such as object or scene recognition require a dense sampling of interest points. The first require high accuracy matching while it is able to tolerate higher dimensional feature vectors. The latter can tolerate lower accuracy matches but requires high speed matching due to the large number of feature vectors; thus, more compact, yet distinctive feature descriptors are needed. In tasks, such as scene recognition or object classification, accurate matches may not be possible due to within-class variations; however, it is still desirable to be able to find a compact representation of

the object or scene that would allow us to make semantic matches accurately. Ideally, feature descriptors will be compact, but descriptive enough to produce accurate interest point matches. In this section, we provide a brief descriptions of various feature descriptors.

### 2.2.1 Local features

Local features are used to provide a description of a small (local) patch around a pixel in the image and, in particular, around interest points found by a feature detector as the ones presented in the section 2.1. They have been used in a wide variety of computer vision application from stereo matching [76, 101, 107] to panorama stitching [16]. If computed densely, they have also been used in object [24, 33, 34, 59] and scene [54, 71, 121] recognition. Below we present an overview of some of the popular local feature descriptors.

#### SIFT

The SIFT (Scale Invariant Feature Transform) descriptor [59] is the most widely used feature descriptor. It has its roots in biological vision, in particular the model of the complex neurons in the primary visual cortex. Its goal is to be invariant to changes in viewpoint and illumination. The descriptor is computed by creating gradient orientation histograms over a 4x4 sample grid around the keypoint. Each histogram has 8 orientation bins which results in a 4x4x8 = 128 element feature vector. The gradient magnitude at each sample point is weighted by the $\sigma$ scale factor based on the Gaussian window. Orientation invariance is achieved by rotating the coordinates of the descriptor and the gradient orientations relative to the keypoint orientation. Invariance to affine changes in illumination is produced by normalizing the feature vector to unit length.

Due to its good performance, SIFT has been used in many vision applications. It has also inspired further research and several feature descriptors have been proposed as extensions to the original descriptor.

#### PCA-SIFT

PCA-SIFT [49] is an extension to the original SIFT descriptor. A 41x41 image patch is extracted at the keypoint location at the given scale and rotated to align its dominant orientation with a canonical direction. This results in a 2x39x39 = 3042 element input vector that is normalized to unit length to handle variations in illumination. Then an eigenspace is built from a diverse collection of images and a large number of patches. Each patch results in an input vector of size 3042 and then PCA is applied to the covariance matrix of these vectors. The matrix containing the top $n$ eigenvectors is used as a projection matrix. Thus, the final descriptor for a given keypoint location is $n$ dimensional vector created by projecting the input gradient vector using the eigenspace. PCA-SIFT results in a more compact representation and according to the authors of [49], the feature vector is more distinctive and more robust than the original SIFT descriptor.

The orientation histograms for the SIFT descriptor are computed on a regular grid. To further improve distinctiveness, a histogram computed on a log-polar location grid was proposed as described below.

#### GLOH

The gradient location-histogram (GLOH) is an extension of the SIFT descriptor [67]. The SIFT descriptor is computed on a log-polar location grid with three bins in the radial direction and eight bins in the angular direction. The size of the resulting histogram vector is reduced using PCA with a covariance matrix estimated from approximately 50,000 image patches from various images. The goal of GLOH is to increase the dis-

tinctiveness of the descriptor and it has been shown to work better in some context than SIFT and others.

Even with the help of PCA, the dimensionality of the descriptors is still relatively high in order to preserve their distinctiveness. Better normalization methods are needed to improve the accuracy of matches between feature vectors.

#### SIFT-Rank

Rank-ordered SIFT (SIFT-Rank) descriptors exploit ordinal description as a way of normalizing invariant image features [100]. Rank-ordering produces descriptors invariant to small deformations of the underlying image data and it improves finding correspondences in high-dimensional feature vectors. The elements in the descriptor vector are sorted based on their values and their rank is considered in the final ordinal description instead of their raw values. SIFT-Rank has been shown to perform better than the standard implementations of SIFT and GLOH descriptors.

The variants of SIFT and GLOH descriptors are expensive to compute on a dense grid due to their high dimensionality. One way to improve the efficiency is to sacrifice some of the distinctiveness of the feature vector.

#### DAISY

The DAISY descriptors is inspired by SIFT and GLOH, but it is less expressive and a lot more efficient to compute [101]. It is intended to be computed over a dense grid for wide-baseline matching. There are 8 orientation maps computed from the image and then convolved several times with different Gaussian kernels. The descriptor at each pixel is then composed as a vector of the values from convolved orientation maps centered on concentric circle with center at the pixel location. The full DAISY descriptor is a concatenation of the vectors at the pixels over a circular grid, similar to GLOH. Computing the descriptor over a circular grid makes it invariant to small

rotation transformations and allows for efficient computation of the descriptor in any orientation by simply re-sampling the grid. Winder et al. propose an algorithm for learning the best DAISY descriptor configuration [117].

The descriptors discussed so far are entirely based on 2D information from the image data. The presence of 3D information could provide extra information and improve the expressiveness of the feature descriptor.

#### VIP: Viewpoint invariant patch

The Viewpoint invariant patches (VIPs) are local image features that extend the SIFT features by incorporating 3D information from previously obtained depth maps [120]. The use of 3D information produces viewpoint invariant image features, not sensitive to distortion under projective transformations. The key idea is estimating the normal to the local tangent plane at each point on the surface and generating a texture patch by orthogonal projection on the plane. The VIP descriptor augments the SIFT descriptor by adding information such as 3D position, patch size, surface normal at the point location, and the 3D vector describing the dominant orientation of the texture. Matching VIPs allows to find points that represent the same surface.

Generally, due to their high dimensionality descriptors are computed only at interest points. However, for certain tasks it might be suitable to compute the descriptors on a dense grid which requires a very efficient computation of the descriptor.

#### HOG: Histograms of oriented gradients

The Histogram of oriented gradients (HOG) descriptor was first proposed for the task of human detection [24], but it has since been shown to work well for scene classification as well [121]. Orientation histograms have also been used in the past for gesture recognition [38]. The HOG features are computed over a dense image grid where for each cell in the grid, a local 1-D histogram of gradient directions is accumulated. The

HOG descriptor is a histogram over a larger spatial region that is a combination of the normalized histograms of grid cells in the region. Dalal and Triggs propose two types of HOG descriptors: R-HOG, computed over a rectangular block similar to SIFT, and C-HOG, computed over a circular block similar to GLOH. The HOG features were intended to be used in a dense grid and unlike SIFT, they are only computed at a single scale without alignment along a dominant orientation.

The descriptors described so far are based on some form of orientation histograms since they can provide invariance to affine transformations. Descriptors based on histograms of color pixel values and relative pixel locations have also been proposed in effort to capture the appearance of objects.

#### Shape Context

The Shape Context descriptor was proposed by [10] with the goal to be used in category level recognition. First a set of $n$ points are sampled from the contours of the shape. The shape descriptor at a point $p_i$ on the shape is the histogram of the relative coordinates of the other $n-1$ points. The histogram bins are defined in log-polar space giving more importance to points closer to $p_i$ than to those further away. In addition to the histogram, during matching a color or gray level appearance similarity of the patches around the points can be considered. Establishing correspondence between the two shapes is achieved by solving the bipartite graph matching problem [48, 80].

#### Spin Image

The use of spin images as a local feature descriptor was introduced in [55]. A spin image is a two-dimensional histogram of affine-normalized patch. It encodes pixel intensity values and their locations, i.e. their distance from the center of the patch. The descriptor is invariant to affine geometric and photometric transformations.

#### SSIM: Self-similarity Descriptor

The local self-similarity (SSIM) descriptor is compact and not sensitive to small affine transformations [90]. It is used to capture the self-similarities based on color, edges, complex textures and repetitive patterns. The self-similarity descriptor is computed at every pixel by correlating a small patch around the pixel with the patches in a larger region surrounding the pixel. The resulting "correlation surface" is binned in a log-polar histogram. Unlike other descriptors, the SSIM descriptor captures regions with similar shapes even though they may vary significantly in appearance, e.g. textured vs uniformly colored region.

#### BRIEF: Binary Robust Independent Elementary Features

More recently, Calonder et al. introduced a very efficient feature point descriptor [18]. They encode the information about the image patch around the feature point as a binary string computed from a small number of intensity difference tests. One limitation of the descriptor is its lack of invariance to changes in orientation. To address this limitation, Rublee et al. introduced ORB, an extension to the BRIEF descriptor that also has rotational invariance [83].

### ■ 2.2.2 Learned descriptors

Originally, feature descriptors were created in an adhoc manner by hand-tuning the various parameters. More recently, learning algorithms have been proposed for automatic parameter and feature selection [14, 19, 20, 74, 118]. In [118], the authors propose a number of building blocks of a feature descriptor and use them to learn the best feature descriptor based on a combination of the proposed building blocks. Prior algorithms use ad-hoc methods in establishing the values for the different algorithm parameters.The descriptor algorithm uses pre-smoothing, seven different types of trans-

formation blocks based on SIFT, GLOH, DoG, and Spin Images, four types of spatial pooling, and a normalization to remove the dependency on image contrast. Among all the different combinations, the best performing one was built by combining steerable filters and a spatial polar arrangement of Gaussian pooling regions with 25 centers.

Local feature descriptors can be augmented with mid-level features to further improve matching accuracy. One such example are the Gestalt inspired featured described in the next section.

### 2.2.3 Global features

Global image features are used to capture the coarse spatial structure of the image. They are estimated by histograms of local features but they give a holistic representation of the scene by capturing the relationships between the different parts. Inspired by studies of the human visual perception showing the ability of humans to quickly gauge the meaning of a complex novel scene, they have been used in scene classification [71] and as a first step to object recognition [54]. Below we describe some of the global features from the literature.

#### GIST

The GIST descriptor is a global feature descriptor introduced by [71] and it has been shown to work well for scene classification. It measures the oriented edge energy at different scale levels aggregated into coarse spatial bins. The image is first converted to grayscale and then subdivided into *nxn* bins. Then oriented edge filters at several different scales are applied to each bin. The GIST descriptor has the property that it matches well to the coarse structure of the scene but is largely invariant to color and contrast due to its local band-pass intensity normalization. While this is beneficial for scene categorization, it might negatively affect the visual similarity of the returned

matches. To address this limitation, [44, 92] proposed to use a simple color descriptor in addition to GIST.

#### Visual words

The notion of visual words comes from the bag of words models first proposed in natural language processing as a representation of documents. The bag of words is an unordered set of words from the dictionary of words. In computer vision, a similar idea has been used as a representation of image content. First local features are computed densely or at interest points. Then visual words, representative of similar image patches, are extracted using for example K-means clustering over all the feature vectors. The centers of the learned clusters represent the visual words in the dictionary. The bag of words model has been shown to perform well in whole-image categorization tasks[42, 114].

#### Spatial Pyramid

While the bag of words approaches have shown promising results, Lazebnik et al. argued that the spatial relationship between visual words carries important information that needs to be considered in scene and object recognition tasks [54]. They showed that their spatial pyramid model outperforms the simpler bag of words ones.

#### Gestalt inspired features

The Gestalt features are inspired by the gestalt principles of continuity, symmetry, closure and repetition [12]. They can be used to augment the existing local feature descriptors, e.g. HOG. These features are more mid-level features, rather than local. The min-max continuity descriptor is used to find long contours in the image by filtering the image using oriented filters and then processing the rectified output by alternating dilation and erosion locally. The longer contours are constructed by concatenating

shorter ones. The circle and form descriptor captures the notion of a closed form and uses an algorithm similar to the Hough transform except the edge elements vote for the circle centers only if they are tangential to the circle. The symmetry feature is computed by comparing patches to mirrored images of patches across a particular line of symmetry and summing all the match strengths that contribute to the same line of symmetry. The repetition descriptors captures the relative similarity of different spatial subregions of the image by comparing image patches after filtering the image into a set of 4 orientation images.

## 2.3 Matching algorithms

Various vision tasks, such as stereo correspondence or object recognition, require accurate matching of interest points or regions. Finding repeatable feature points and developing compact and distinctive feature descriptors are necessary for providing solutions to these tasks. Most feature matching algorithms are based on some form of nearest neighbor search. There have been a number of efficient algorithms for finding good matches proposed in the literature, e.g multidimensional search trees (kd-trees) [9], different hashing methods [15, 52, 78, 88, 103, 116], approximate nearest neighbors [70].

# Chapter 3

# Photorealistic virtual world dataset

## 3.1 Introduction

THERE has been an explosion in image data on the Internet in recent years. Despite the abundance of image data, it is difficult to acquire ground truth data under controlled conditions. The images found on photo-sharing websites such as Yahoo's Flickr or News Corp's Photobucket, are taken from various viewpoints and under many different lightning conditions. Images from a common location can be calibrated to a common 3D space as implemented in the PhotoTourism system by [96]. However, even for popular landmarks with tens of thousands of images, there are challenges in reconstructing the 3D geometry of the scene. Finding pixel to pixel correspondences between images is particularly difficult in the presence of occlusions, illumination changes, and specular surfaces. We would like to study what effects different phenomena, such as changes in lighting or viewpoint, have on the design and performance of image features. For this, we need ground truth data that will allow us to explore the effects of these phenomena in isolation.

In the past, several different datasets have been proposed for the evaluation of image descriptors. Mikolajczyk et al. presented a dataset of several images under various transformations [67, 86] addressing the need for ground truth data. Due to difficulty of attaining correspondences, the dataset was limited to planar scenes or images taken

**Figure 3.1:** *Sample images from the Virtual City dataset.*

from a fixed camera position. These do not capture the full complexity of viewpoint changes - changes in perspective beyond those of planar scenes or the presence of occlusions. The dataset includes an example of change in illumination simulated by changing the camera settings, essentially changes in brightness and contrast. However, these do not capture changes in light source position that result in shadows and non-uniform changes in intensity.

To address such problems, Winder et al. recently proposed using a dataset of patches from several famous landmarks [117, 118]. They used camera calibration and multiview stereo data of 1000 images for each landmark to find corresponding inter-

**Figure 3.2:** *Sample images from the Statue of Liberty dataset .*

est points between the images using estimated dense surface models . While these datasets contain image patches taken from different points of view and under different illumination, it is difficult to evaluate the effect each of these has on the descriptor performance, since the variations in viewpoint, illumination and camera type are uncontrolled. Moreels et al. proposed a dataset of 100 real 3D objects viewed from 144 calibrated viewpoints under three different lighting conditions [68]. However, those do not contain complex scenes and interactions between objects such as occlusions, cast shadows, specularities, etc. We want to be able to capture a wide range of scenes under different transformations. To gain complete and repeatable control over specific aspects of the environment, we propose using a photorealistic virtual world.

With the great progress in the field of computer graphics in the last two decades, it is possible to generate high quality realistic scenes. Recent work has shown that the use of synthetic image/video data can be used to evaluate the performance of tracking and surveillance algorithms [99], to train classifiers for pedestrian detection [64] and to learn locations for grasping novel objects [85]. The first two works use images rendered from the game engine of Half Life 2 [64, 99]. While this is a modern game


**Figure 3.3:** *Samples images from the virtual city. Images from a static camera of a scene under different illumination (5 different times of the day).*

engine, capable of generating high quality images, the quality is limited by the processing power required to render the scenes in real time. The later work uses simple, non-photorealistic renderings of individual objects rather than complex scenes [85]. In contrast, we propose the use of a highly photorealistic virtual world for the evaluation and design of image features. We generated two data sets of images taken under different illumination and from different viewpoints from high resolution 3D graphics models of a virtual city and of the Statue of Liberty. The images were rendered with 3ds Max's Mental Ray renderer using advanced materials, including glossy and reflective surfaces, high resolution textures, and the state-of-the-art Daylight System for illumination of the scene.

**Figure 3.4:** *Samples images from the virtual city.* ***Top row:*** *Scene from a panning camera at 22.5 degree rotation stops.* ***Bottom row:*** *Images taken from a different camera viewpoint and location of the center image in the* ***Top row****.*

## ■ 3.2 Photorealistic Virtual World Dataset

Figures 3.1 and 3.2 show sample images rendered from the Virtual City and the Statue of Liberty 3D models, respectively. We used the Statue of Liberty dataset to calibrate our virtual evaluations against feature rankings made using photographic data as described in chapter 4.

### ■ 3.2.1 Photorealistic City Model

For our virtual city dataset, we used a high resolution city model from Turbosquid [106] containing over 25 million polygons. The model has 12 city blocks with 82 unique buildings with highly detailed geometry and advanced textures from residential and commercial ones to churches, schools, theaters and museums. It also includes parks, sport fields, parking lots, and objects found in a city environment, from lamppost

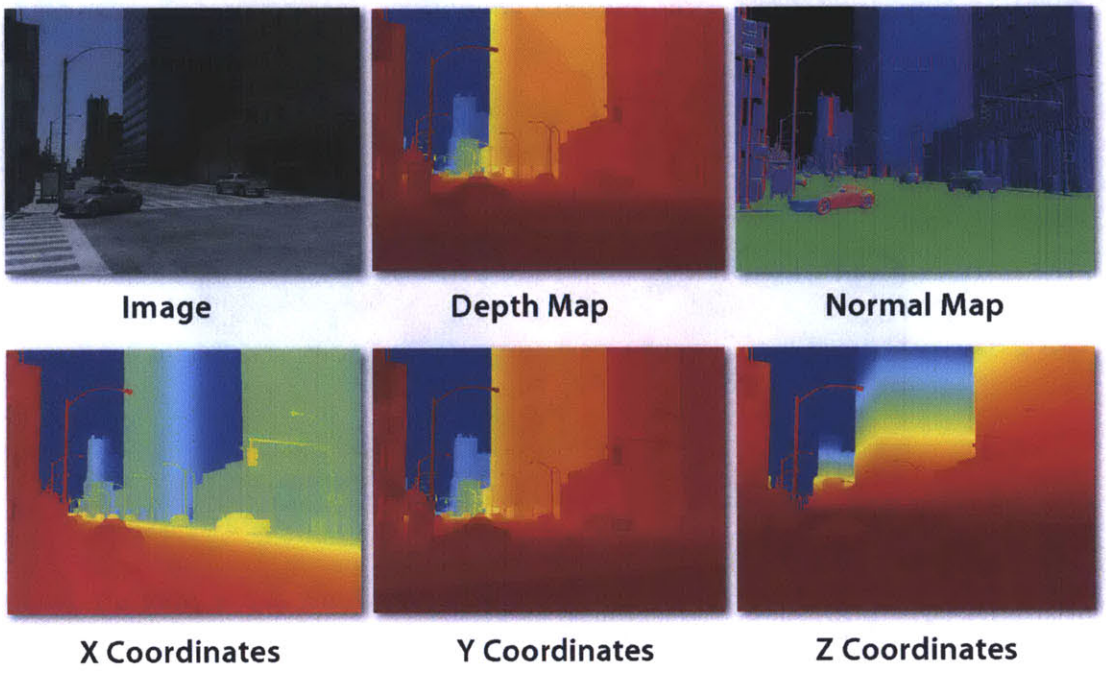



**Figure 3.5:** *Images showing the data we collect for each scene.*

and trashcans to benches and bus stops (although no people). We also added 25 different high resolution vehicles to the model that contain advanced glossy and reflective surfaces. To increase the number of vehicles, we varied their colors. The dataset was rendered using 3ds Max's Mental Ray renderer to produce high quality photorealistic city images.

To light the environment, we used 3ds Max's Daylight system that positions the sun light source automatically after specifying the location, the date and time [5]. We rendered five images for each scene taken at 9am, 11am, 1pm, 3pm and 5pm on a sunny June summer day in Boston, MA (Fig. 3.3). We used a 35 mm camera lens. To automatically render the different scenes, we created a fly-through camera animation simulating a person walking along the city streets and varied the illumination at each camera position. At each camera location, we took three different shots panning the

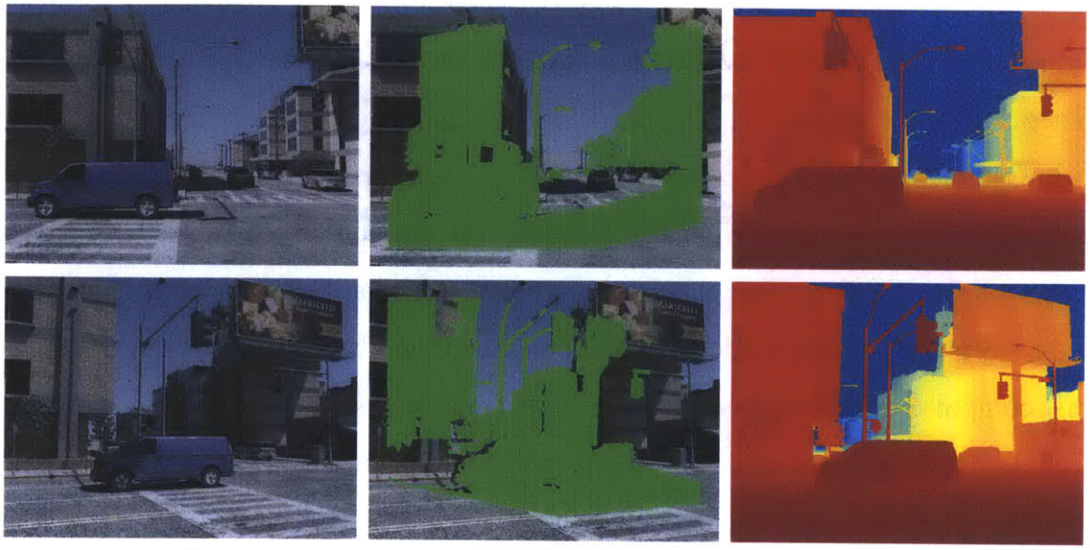



**Figure 3.6:** *Examples of images from our virtual city. a) Image pair of a scene under different viewpoint and illumination. b) The corresponding set of 3D points between the images in a). c) The corresponding depth maps of the images in a).*

camera at 22.5 degree steps (Top row of Fig. 3.4). Neighboring locations were close enough to capture the scene at the current camera position from a different viewpoint, e.g. the bottom row of figure 3.4 shows different viewpoints of the scene captured in the center image of the top row of figure 3.4. In this work, we used 3000 images from 200 different camera locations over 4 city blocks with 15 images taken at each location - three different camera orientations and five different illumination settings for each orientation. The images were rendered at resolution of 640x480 pixels. No noise or compression artifacts have been added to the images though they can be easily added as postprocessing step. These phenomena were studied previously in [67].

The benefit of using a photorealistic virtual world is that we know the geometry of the scene and can easily generate labeled ground truth data. In addition to the high quality rendered scenes, we also produce depth maps, normal maps, and the 3D co-

ordinates of each pixel in the virtual world. Figure 3.5 shows an example of the data generated for a single scene. The availability of the 3D geometry of the scene allows us to find the pixel to pixel correspondences between images of the same scene taken from different viewpoints. Figure 3.6 b) shows the pixel correspondences between the images in figure 3.6 a). Note that we do not have 3D coordinates for pixels belonging to the sky and therefore, we do not have correspondences between pixels of the sky. We only compute correspondences between pixels that are visible in both images. We also exclude pixels from the regions close to the boundary of the image since we cannot compute image descriptors accurately there.

### 3.2.2 Statue of Liberty

Since the photographic subject can influence feature performance, to study whether our photorealistic virtual world would be a good predictor for descriptor performance in the real world, we compared descriptor performance on a synthetically generated dataset of the Statue of Liberty to that on the real world Liberty dataset of [117] (see chapter 4). We purchased a high resolution 3D model of the Statue of Liberty and rendered 625 images at 640x480 resolution. We simulated the camera moving around the statue on the ground level in a circular arc centered at the statue. We rendered the scene at every 10 degrees for 250 degrees around the front of the statue and under five different locations of the sun, similar to our city dataset. We used 4 different camera lenses - 50mm, 85mm, 135mm, and 200mm - to acquire both distant and close up shots. We used the 135mm lens at two different angles - viewing the top and the base of the statue.

## 3.3 Conclusion

In this chapter, we described the photorealistic datasets used in chapters 4 and 5. We generated 3000 highly photorealistic images from 200 scenes of our virtual city model where each scene was captured from 5 different viewpoints and 5 different lighting conditions. We generated a dataset of 625 images of the Statue of Liberty from 25 different locations around the statue using different camera lenses and under 5 different locations of the sun. In chapter 4, we use the Statue of Liberty dataset to compare the relative performance of feature descriptors to that based on using photographic data of the Statue of Liberty provided by [117]. We then use the photorealistc virtual city dataset to measure the performance of different feature descriptors under controlled conditions. We consider changes in lighting, changes in viewpoint, and a combination of both. Finally, we propose using the available depth information for the scene to improve the performance of existing descriptors. In chapter 5, we use the photorealistic virtual city dataset to define what good features are and employ machine learning techniques to learn a model for detecting them.

# Chapter 4

# Evaluation of image descriptors

## 4.1 Introduction

IMAGE features play an important role in computer vision. They are used for tasks ranging from wide baseline stereo matching [76, 101, 107], panorama stitching [16] and 3D scene reconstruction [120] to object [24, 33, 34, 59], scene [54, 71, 121], texture [55] and gesture recognition [38]. Because of their importance and wide use, optimizing image features is a critical task.

The goal in designing features is that they must be robust, distinctive and invariant to various image and scene transformations. One of the challenges is acquiring ground truth data necessary for evaluating and comparing different image descriptors. Here, we use the two data sets proposed in chapter 3 from our highly photorealistic virtual world to evaluate the performance of image features. The data sets contain images taken under different illumination and from different viewpoints from high resolution 3D graphics models of a virtual city and of the Statue of Liberty.

We first seek to calibrate our virtual world evaluations against feature rankings made using photographic data. To control for image content, we compare the performance of feature descriptors on datasets based on real and synthetic images of the Statue of Liberty, and we find very similar feature rankings from the photorealistic and photographic datasets. We then exploit the flexibility of our virtual world to make

controlled evaluations that are very difficult to make from photographs. We use our controlled environment to evaluate the effects of changes in viewpoint and illumination on the performance of different feature descriptors. We can study the effect of augmenting the descriptors with depth information to improve performance.

## 4.2 Feature Descriptors

We used our dataset to evaluate the performance of a selection of commonly-used feature descriptors.

### 4.2.1 Scale Invariant Feature Transform (SIFT)

SIFT has been widely used in a variety of computer vision applications from object recognition to panorama stitching. We compute the descriptor similarly to [59]. After initial pre-smoothing of the image by $\sigma = 1.8$, we quantize the gradient orientation at each sample into $d$ directions and bin them in 4x4 spatial grid. Each gradient direction is weighted bilinearly according to its distance to the bin centers. The final descriptor is normalized using a threshold of 0.2 as in SIFT [59]. We used 4, 8 and 16 gradient directions thus creating three descriptors of dimension 64, 128, and 256 - these are referred to as T1a-S1-16, T1b-S1-16, and T1c-S1-16 in [118]. The descriptor was computed over a patch of 61x61 pixels centered at the sample.

### 4.2.2 Gradient Location and Orientation Histogram (GLOH)

GLOH was proposed as an extension to SIFT to improve robustness and distinctiveness of the descriptor [67]. We quantized the gradient orientations as in SIFT and then bin them in a log-polar histogram of 3 radial and 8 angular directions. Only the outer bins are divided into 8 directions, thus there are total of 17 bins. The size of the patch around the sample was 61x61 pixels and the final descriptor was normalized similarly to SIFT.

We used 4, 8 and 16 gradient directions resulting in 68, 136 and 272 dimensional feature vectors - these are similar to T1a-S2-17, T1b-S2-17, and T1c-S2-17 in [118]. Note that we do not reduce the size of the descriptors in our experiments, unlike [67].

### 4.2.3 DAISY

The DAISY descriptors is inspired by SIFT and GLOH, but designed for efficient computation [101]. Learning the best DAISY configuration was proposed by [117]. We compute $d$ gradient orientation maps and then convolve them with different Gaussian kernels depending on their distance from the center. The descriptor is then computed over a log-polar arrangement similar to GLOH. The vectors in each pooling region are normalized before concatenated in the final descriptor. We used three radial and eight angular directions for a total of 25 sample centers including the one at the center of the grid. The image patch is 61x61 pixels centered around the sample. We used 4, 8, and 16 gradient directions resulting in 100, 200, and 400 dimensional feature vectors - these are referred to as T1a-S4-25, T1b-S4-25, and T1c-S4-25 in [118].

### 4.2.4 Histograms of oriented gradients (HOG)

The HOG descriptor [24] and its variants [32] have demonstrated excellent performance for object and human detection. Similar to the SIFT [59], the HOG descriptor measures histograms of image gradient orientation but uses a different type of normalization. We use the same approach as described in [32]. However, we compute the descriptor for 4, 8, and 16 gradient orientation. The descriptor was computed over a patch of 61x61 pixels covering a neighborhood of 3x3 cells. We only use the descriptor for the cell centered at the sample resulting in very low dimensional feature vectors of 10, 16, and 28 dimensions. Low dimensional feature vectors are useful for efficient matching in tasks requiring large number of keypoints to be matched. Since computing

| Descriptor | HOG8 | SIFT16 | GLOH8 | DAISY16 |
|---|---|---|---|---|
| Notre Dame Real | 0.898 | 0.958 | 0.961 | **0.964** |
| Liberty Real | 0.885 | 0.947 | 0.950 | **0.953** |
| Liberty Synthetic | 0.896 | 0.950 | 0.955 | **0.959** |

**Table 4.1:** *Area under the ROC curve for different descriptors on the real Notre Dame and Liberty and the synthetic Liberty datasets. Note the feature rankings on both the real and synthetic datasets is the same despite the variation in individual performance. The feature ranking is the same even across datasets with different image content.*

HOG is similar to SIFT using only the center cell allows us to compare the performance of descriptors with different dimensionality.

### ■ 4.2.5 The self-similarity descriptor (SSIM)

The self-similarity descriptor [89] has been shown to perform well on matching objects of similar shape but vastly different local appearance. The idea is to represent the appearance in a local image area around a particular image patch by the "correlation map" of the patch with its neighborhood. The descriptor captures the local pattern of self-similarity. Each descriptor is obtained by computing the correlation map of a 5x5 patch in a window with radius equal to 30 pixels, then quantizing it using a log-polar histogram as in GLOH. We used 3 radial bins and either 8 or 16 angular bins, resulting in 24 or 48 dimensional feature vectors.

## ■ 4.3 Evaluation

Keypoints are the image locations where we compute descriptors. We computed keypoints using one of three different methods: spatial local maxima of a Difference of Gaussian (DoG) filter [59], the Harris corner detector [43], and a dense spatial grid at 5 pixel offset. We use the implementation of the keypoint detectors by [110]. For the

experiments presented here, we use the DoG keypoints from the first octave of the scale space. Since our dataset is synthetically generated, we know the complete geometry of the scene and therefore the pixel correspondences across images. Figure 3.6 a) shows a pair of images taken from different viewpoints and under different illumination. The overlapping part of the scene and the points in the images for which we have correspondences are shown in figure 3.6 b). Note that we do not match points in the sky for images from a different viewpoint since we do not have actual 3D coordinates for them. They may, however, be considered in experiments where the camera is static. For each image pair A and B, we compute the descriptors at each keypoint in image A and its corresponding 3D point in image B. We define the matching keypoints to be the true correspondences. The non-matching keypoints are defined to be keypoints that are at a distance of at least 10 pixels from the true correspondence in image space since neighboring keypoints in image space will have very similar descriptors and thus cause confusion in the matching. We follow the protocol of Winder et al. [118] to form an ROC curve of descriptor performance. We compute the Euclidean distance between the descriptors computed at each pair of matching and (randomly selected) non-matching keypoints. As a function of a distance threshold, we compute the number of correct and false matches that is the matching and non-matching keypoints with a descriptor distance below the threshold, respectively. Sweeping that computation over a descriptor distance threshold yields a receiver operating characteristic (ROC) curve. The correct match rate and the false positive rate for each discrimination threshold are:

$$\text{Correct Match Rate} = \frac{\#\textit{correct matches}}{\#\textit{matching keypoints}} \tag{4.1}$$

$$\text{False Positive Rate} = \frac{\#\textit{false matches}}{\#\textit{non-matching keypoints}} \tag{4.2}$$

The larger the area under the ROC curve, the better the performance of the descrip-

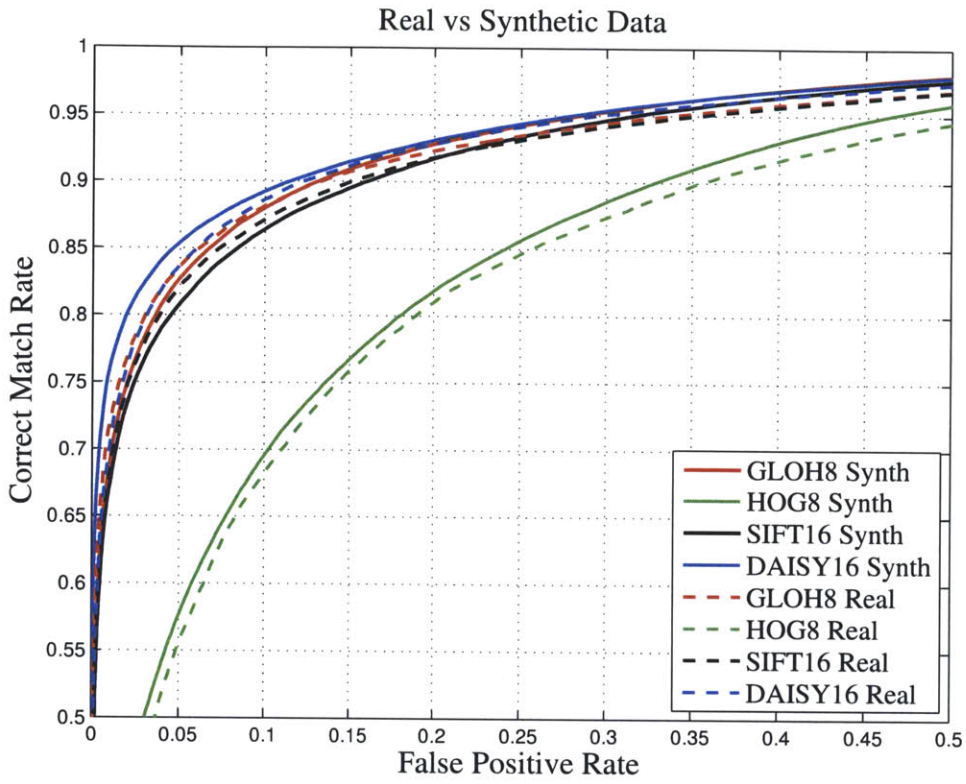



**Figure 4.1:** *Performance of the synthetic vs real world Statue of Liberty datasets on a set of descriptors. Note that the performance on both datasets is very similar and the relative ranking of the descriptors is the same.*

tor.

## 4.4 Experiments

### 4.4.1 Overview

To first confirm that our virtual world and the real world gave similar rankings, controlling for image content, we compare feature descriptors using the photographic Liberty patch dataset of [117] and our synthetic Statue of Liberty dataset. We find that the descriptors perform comparably on both datasets and the relative rank is the same. We

proceed to study the effect of changes in illumination of outdoor scenes and changes in camera viewpoint on the descriptor performance. Since our dataset is synthetically generated, we have full control of the scene and we can capture the exact same scene both under different illumination and different camera viewpoint and we have full knowledge of the geometry of the scene that allows to match keypoints accurately. We compare the degradation of all of the descriptors with changes in illumination and viewpoint. We find that the log-polar pooling scheme seems to perform better than the grid one for coping with changes in illumination, while the number of pooling regions has a bigger effect when there are changes in camera viewpoint. We also propose a 3D descriptor in the presence of depth map data and show that even a very low dimensional descriptor like HOG computed over the depth map can lead to improved feature matching performance.

### ■ 4.4.2 Real vs Synthetic Data

To calibrate our virtual world descriptor evaluations, we compared the performance on the Liberty patch dataset of [117] and our synthetic Statue of Liberty dataset, using 100000 patches/keypoints in both cases. The keypoints from the synthetic dataset were selected randomly from all the detected keypoints.

For this experiment, we only used images that have a partial or full view of the front of the statue as this seems to be the case for most of the images found online. Figure 4.1 a) shows performance of a set of the image descriptors on both the real and synthetic data. The ROC curves are very similar showing only slight variation and the ranking of the performance of the different descriptors is the same. The slightly worse performance of the descriptors on the real dataset could be due to inaccuracies in the patch matching. There can be some variation of the descriptor performance depending on the data they are applied to as shown in table 4.1. To study the change in feature

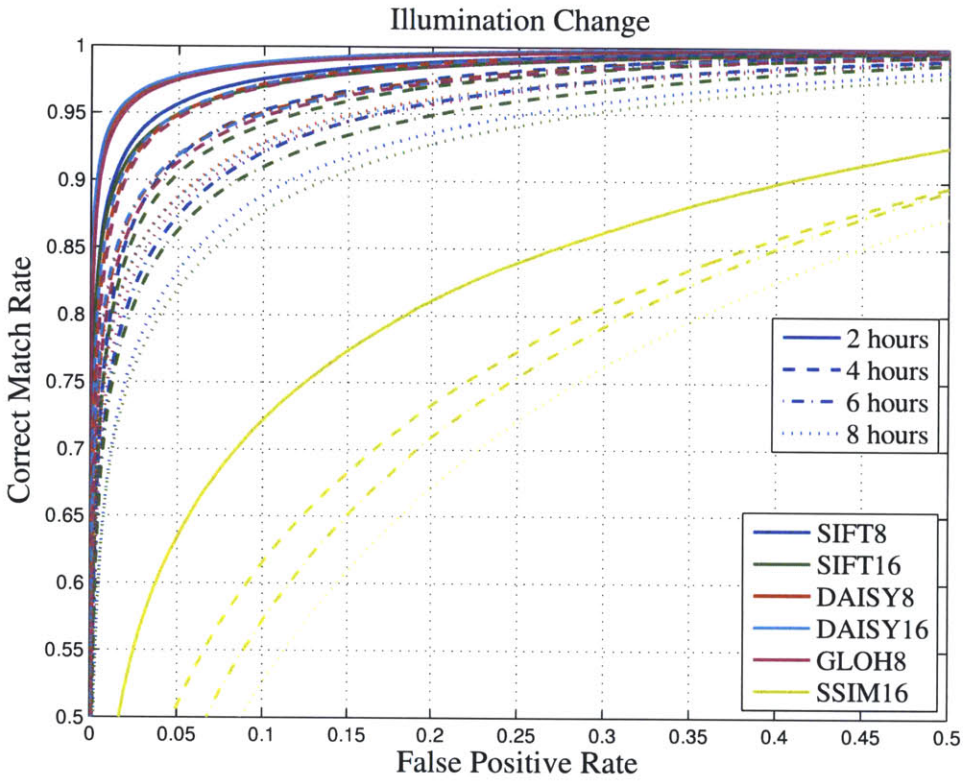



**Figure 4.2:** *Descriptor performance for a images from the virtual city taken with a static camera of a scene under different illumination (2,4,6 and 8 hour difference). The performance degrades with larger changes in illumination. DAISY8 and GLOH8 perform best in this context.*

rankings with image content, we kept the evaluation method fixed (photographic image patches) but compared the performance of features for the Notre Dame dataset [117]. The descriptors perform better on the Notre Dame dataset than on the Liberty one; however, even in this case the ranking of the descriptors is still the same. The better performance on the Notre Dame data set is probably due to the larger number of edge structures in the scene. These results show that (1) we can translate the relative performance of the descriptors on the synthetic data to that of the real data, and (2) the relative rankings appear to change very little across image content.

### 4.4.3 Illumination Change

Changes in illumination can result in large changes in the appearance of the scene due to shadows, specular reflections, etc. We compared the performance of the different descriptors under different illumination using our virtual city dataset. Each pair of matching keypoints belonged to images of the same scene taken with a static camera during two different times of day. We used 2.2 million keypoint pairs with peak threshold of 0.01 and edge threshold of 10. Figure 4.2 shows the performance of a subset of the descriptors for the same scene taken at 2, 4, 6, and 8 hour difference. The performance degrades with the increase of the time difference betwen the rendered images as the changes in illumination of the scene are more significant. The performance of the other descriptors followed a similar trend. The much worse performance of the SSIM descriptor is likely due to its smaller dimension and lack of distinctiveness as it was meant to be computed densely. The almost identical performance of the DAISY8 and DAISY16 descriptors shows that increasing the number of gradient orientation to 16 is not beneficial. In the case of SIFT, the performance even appears to degrade slightly. DAISY8 and GLOH8 perform very similarly to each other and better than SIFT in the presence of changes in illumination. That may be due to their use of the log-polar binning arrangement, common to DAISY8 and GLOH8.

### 4.4.4 Viewpoint Change

We performed a couple of experiments to evaluate the effects of viewpoint change on the different descriptors on both of our datasets - Statue of Liberty and Virtual City.

Our synthetic dataset of the Statue of Liberty contains images taken by moving the camera along a circle around the statue at 10 degree stops. We evaluated the performance of the descriptors as we move the camera up to 80 degrees from the reference image on images taken under the same lighting conditions. Figure 4.3 shows the perfor-

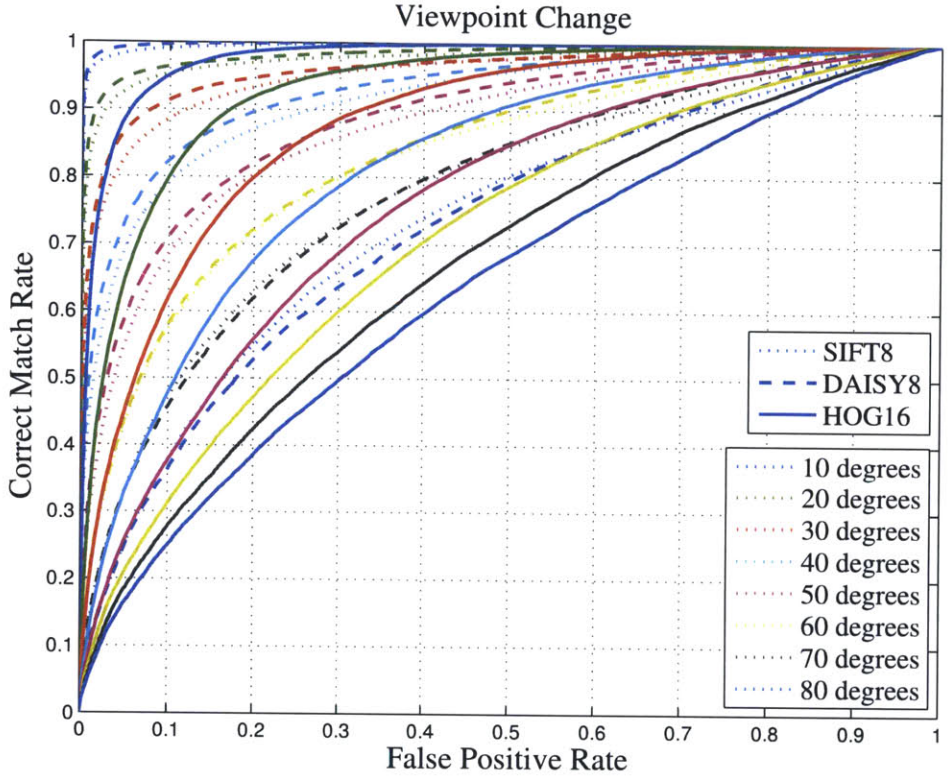



**Figure 4.3:** *Performance of descriptor on the virtual Statue of Liberty dataset for varying camera viewpoints (10-80 degrees rotation around the statue) under constant illumination. The performance of all descriptors degrades with larger change in viewpoint. DAISY8 performs better under small changes in viewpoint while SIFT8 performs better under larger changes.*

mance of several descriptors and how it degrades with the increase in angle between the camera locations. The performance of the DAISY8 descriptor degrades faster after 50 degrees and the performance of the HOG16 descriptors almost reaches chances level. The much worse performance of HOG16 may be related to its lower dimensionality (28) in comparison to the SIFT8 (128) and DAISY8 (200) descriptors.

We evaluated the performance of the descriptors on our virtual city dataset for keypoints in images taken under different viewpoint (Fig. 3.4) but under the same illumination using 1.3 milliion keypoint pairs. All images were taken at 1pm. The ranking

for the descriptors was similar to that under changes in illumination (section 4.4.3) except for GLOH8 (Fig. 4.4). Under viewpoint changes, the performance of the GLOH8 descriptor is similar to that of SIFT8, not to DAISY8 as in section 4.4.3. This could be explained by the larger number of pooling regions in DAISY8, 25 versus 17 in GLOH8 and 16 in SIFT8. It appears that the arrangement of the pooling regions is important for illumination changes in the scene while the number of pooling regions matters in scenes captured from different viewpoints. Here, again the performance of HOG16 and SSIM descriptors may be related to the descriptor dimensionality.

### ■ 4.4.5 Viewpoint and Illumination Change

In sections 4.4.3 and 4.4.4, we considered the effects of illumination change on a scene taken with a static camera and the effects of viewpoint change under constant illumination. Here, we compare the effects of camera position location under different illumination for one of the descriptors DAISY8. The relative performance of the other descriptors was similar. We considered the performance of DAISY8 for scenes taken under different illumination (2, 4, 6, and 8 hours apart) with a static camera, with a camera at the same location at rotation stops of 22.5 degrees (Fig. 3.4 top row)) and camera from different locations (Fig. 3.4 bottom row). The performance with the panning camera (Cam1) is not as good as that of the static camera and the difference in performance grows with larger changes in illumination (Fig. 4.5). The task of matching keypoints in images taken from different camera position and rotation (Cam2) is a lot more challenging and the descriptor performance is considerably worse. This is because here the changes in perspective, occlusions, etc. play much larger role. It is especially true for keypoints around contour boundaries, where the background could significantly change due to changes in viewpoint.

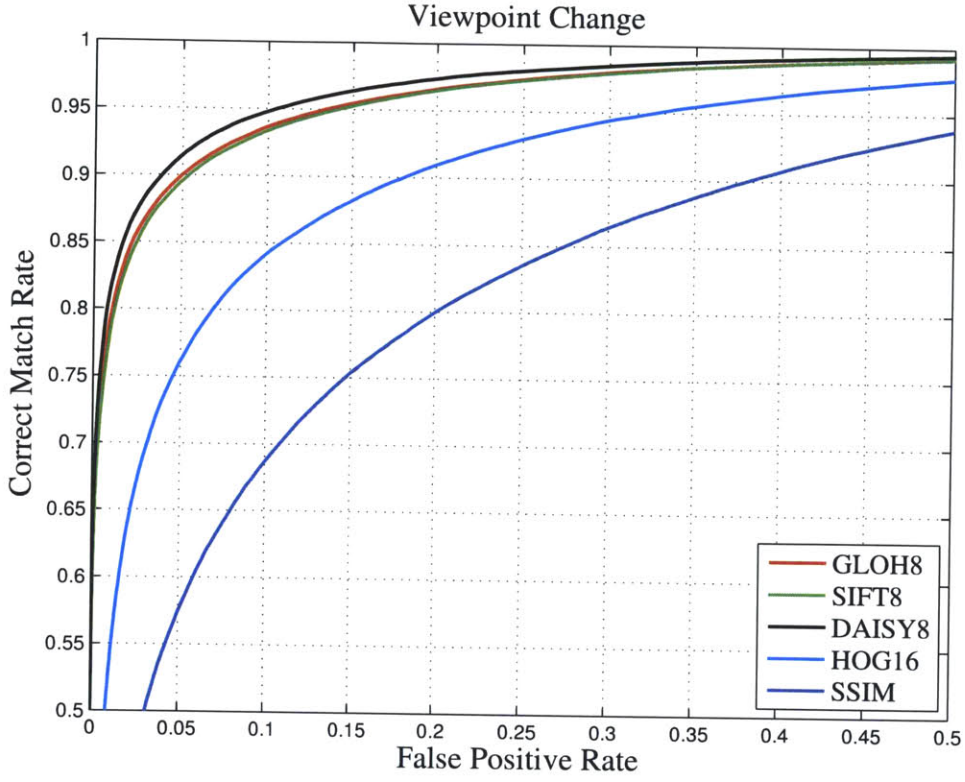



**Figure 4.4:** *Performance of descriptors under different viewpoint for scenes from the virtual city under constant illumination. Note here GLOH8 and SIFT8 perform similarly, where as GLOH8 performed better than SIFT8 under changes in illumination.*

### 4.4.6 3D Descriptors

Suppose that in addition to the RGB information, we also have depth information available. Depth can be acquired by many different means, at a range of quality levels. Since we know the full geometry of each scene in our virtual city, we have depth maps easily available (Fig. 3.6 c)), and we can assess the utility of incorporating depth information into feature descriptors. Since acquiring high resolution depth maps is difficult, we quantized the depth maps from our virtual city to $n$ depth levels to approximate a depth map acquired in a real world setting. We expect that knowing depth will be particularly helpful in two scenarios. For images of a scene under different illumination, it can dis-

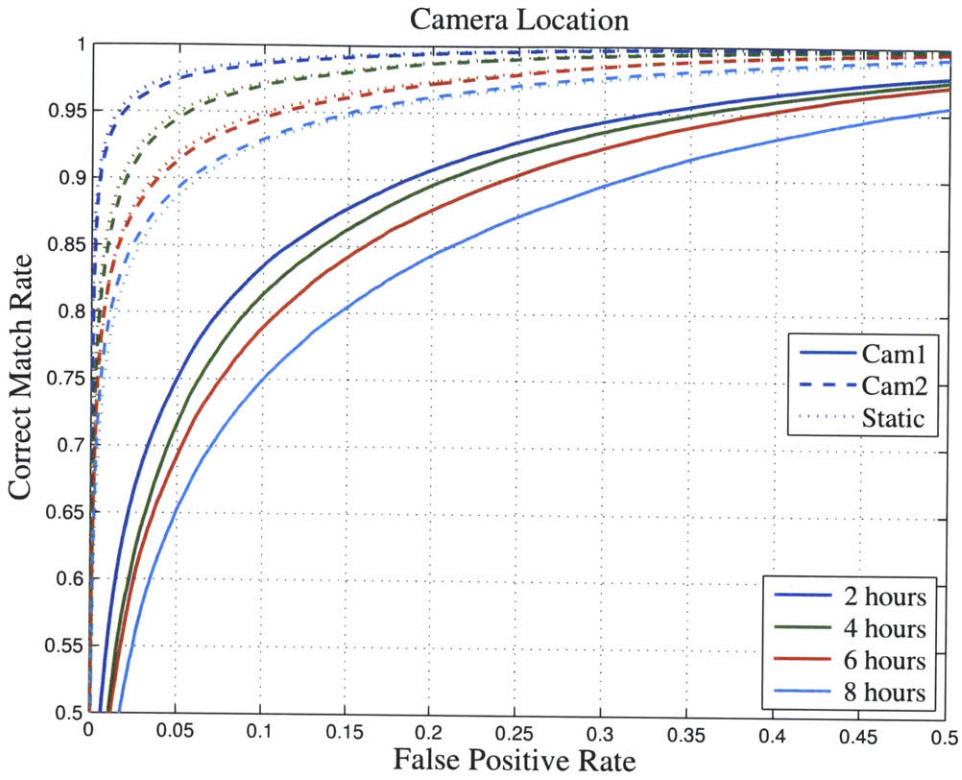



**Figure 4.5:** *Performance of the DAISY8 descriptor for scenes taken under different illumination (2, 4, 6, and 8 hours apart) with a static camera, with a camera (Cam2) at the same location at rotation stops of 22.5 degrees (Fig. 3.4 Top row) and a camera (Cam1) from different locations (Fig. 3.4 Bottom row). The descriptor has most difficulty with large changes in viewpoint.*

tinguish between edges due to depth discontinuities and due to shadows. For images under different viewpoint, it can help match keypoints on contour boundaries despite significant changes in the appearance of the background.

We propose to augment the feature descriptors in the following way. For each keypoint, we compute the descriptor, $F_{rgb}$, using the RGB image (Fig. 3.6 a)) and the descriptor, $F_{depth}$, using the depth map (Fig. 3.6 c)). Thus, the final descriptor is $[F_{rgb}; F_{depth}]$. We experimented with different combinations of descriptors for $F_{rgb}$ and $F_{depth}$ and different depth resolutions, $n = 16, 32, 64, 128$, and 256. We found that using

descriptors based on histograms of oriented gradients for $F_{depth}$ produced best results as they capture the information about the relative depth of the pixels in the neighborhood around the keypoint. To evaluate whether two keypoints match, we compute the weighted sum of the Euclidean distance between the descriptors from the RGB image, $D_{rgb}$ and the Euclidean distance between the descriptors from the depth map, $D_{depth}$.

$$D_{desc} = \alpha D_{depth} + (1 - \alpha) D_{rgb} \tag{4.3}$$

We performed different experiments with various values of alpha. We see greater improvement in performance for larger changes in viewpoint and illumination. Figure 4.6 shows the performance of the SIFT8 descriptor for the RGB image, HOG16 descriptor for the depth map quantized to 64 depth levels and alpha value of 0.3 in comparison to using the SIFT8 descriptor alone. Even a very low dimensional descriptor as HOG16 (28) that adds minimal computational overhead produces a significant improvement in the performance of descriptors in challenging illumination and viewpoint conditions. Using higher dimensional descriptors like GLOH or SIFT for the depth map descriptor improves the performance further but at the expense of higher computational cost. Even depth maps with a resolution as low as 16 depth levels produce improvement in performance. Higher resolution depth maps (greater than 64 levels) improve the performance further but not significantly.

## 4.5 Conclusion

We used a photorealistic virtual world to evaluate the performance of image features. We used two datasets of photorealistic images – one from a virtual city and the other of a model of the Statue of Liberty. We showed that the performance of the descriptors on similar datasets from the real world and virtual Statue of Liberty is similar and results in the same ranking of the descriptors. Working in a virtual world allows complete

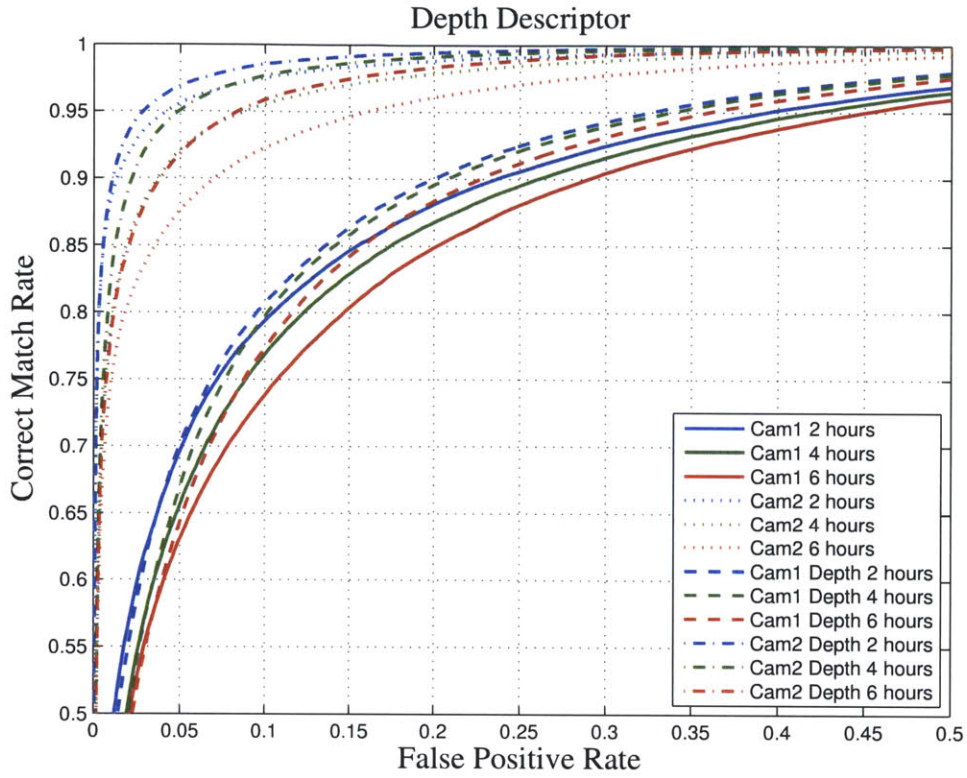



**Figure 4.6:** *The performance of the SIFT8 descriptor in comparison with the combined SIFT8 on the RGB image plus the HOG16 on the depth map (64 depth levels) 3D descriptor under different camera viewpoint and varying illumination conditions. Note the performance of the 3D descriptor has a larger performance gain for larger changes in viewpoint (Cam1).*

knowledge of the geometry of the scene and full control of the environment, thus allowing to study the impact of different parts of the environment on the descriptors in isolation.

Our experiments on the dataset of our virtual city show that the DAISY descriptor performs best overall both under viewpoint and illumination changes. We found that the spatial arrangement of the pooling regions in the gradient descriptors has an impact on the descriptor performance for matching keypoints in images taken under different illumination. The number of pooling regions on the other hand needs to be

considered for images taken from different camera viewpoint. The lower dimensional feature descriptors generally performed worse due to lack of distinctiveness. However, we showed that using a low dimensional descriptor such as HOG can help improve descriptor performance if applied to the depth map of the scene and used in conjunction with a feature descriptor over the RGB image. We ranked features with regard to specific image transformations (viewpoint, and lighting variations over time-of-day).

Using high quality 3D computer graphics models as we have here allows for controlled and specific evaluation of image features, and may allow new features to be designed and optimized for specific computer vision tasks.

# Chapter 5

# Learning interest point detectors

## 5.1 Introduction

IN the previous chapter, we explored the effects of lighting and viewpoint changes on different feature descriptors. Here we take a step back and investigate the design and performance of feature detectors for a specific descriptor, the SIFT descriptor. Identifying the best locations to extract local image features is an important first step for many computer vision algorithms e.g. wide baseline stereo matching [101, 107], panorama stitching [16], object recognition[24, 59]. The quality of the image features and their selected location has a large influence on the subsequent computer vision tasks [59, 86]. The extraction of local features is generally formulated as a two stage process. First an interest point detector [8, 43, 50, 59] locates regions in the image that are likely to provide reliable correspondences with a second image of the same object or place. Then a descriptor [24, 59, 67, 101] is used to represent the local patch. There have been a number of works comparing different approaches existing in the literature [67, 68, 86]. There has also been an attempt to learn what are the best parameters of the SIFT descriptor [118], but there has been no work trying to learn a detector that extracts the best feature locations. Those detectors are generally based on heuristics about what is supposed to be an interesting image region.

Here, we propose using the descriptor to measure the "interestingness" of the im-

age region given a set of images with ground truth pixel correspondences. Collecting a ground truth dataset of real world complex 3D scenes that contain changes in viewpoint and illumination is difficult. Datasets previously used [66, 68, 105, 118] are of planar images or small set of simple 3D models that do no contain phenomena such as occlusions, cast shadows, inter-reflections. Therefore, we choose to use the dataset of images from a highly photorealistic virtual city described in chapter 3 that provides a controlled environment to study the effect of changes in viewpoint and illumination. We use the descriptor based measure of how interesting an image region is to select a set of distinctive points and use those to train a model for an interest point detector. Machine learning has been used in the space of feature detection by Rosten et al. who propose techniques for optimizing the speed of a detector [81] and by Trujillo et al. who use genetic programming to synthesize new low-level operators for point detection starting from a set of existing ones [105].

In this chapter, we propose a method for selecting good interest points given a feature descriptor. We require the points to have the following properties: distinctiveness with respect to a local neighborhood of points, global separability and repeatability. We use 1800 images taken under different viewpoint and illumination from the virtual city dataset presented in chapter 3. Using a 3D graphics model gives us complete knowledge of the geometry of the scene and, thereby, ground truth pixel correspondences between images. Our method selects a set of good interest points based on how similar the feature description of the corresponding points is to each other and how distinctive it is from a local neighborhood around the points. We created a database of over 3 million positive and 3 million negative examples of good interest points for the dense SIFT descriptor. We used a linear support vector machine to train a model that detects good interest points suitable for use with the SIFT descriptor. We proposed using different sets of features to define the interest point locations and found that the descriptor itself is the best predictor. Finally, we compare the learned models to two of the widely used

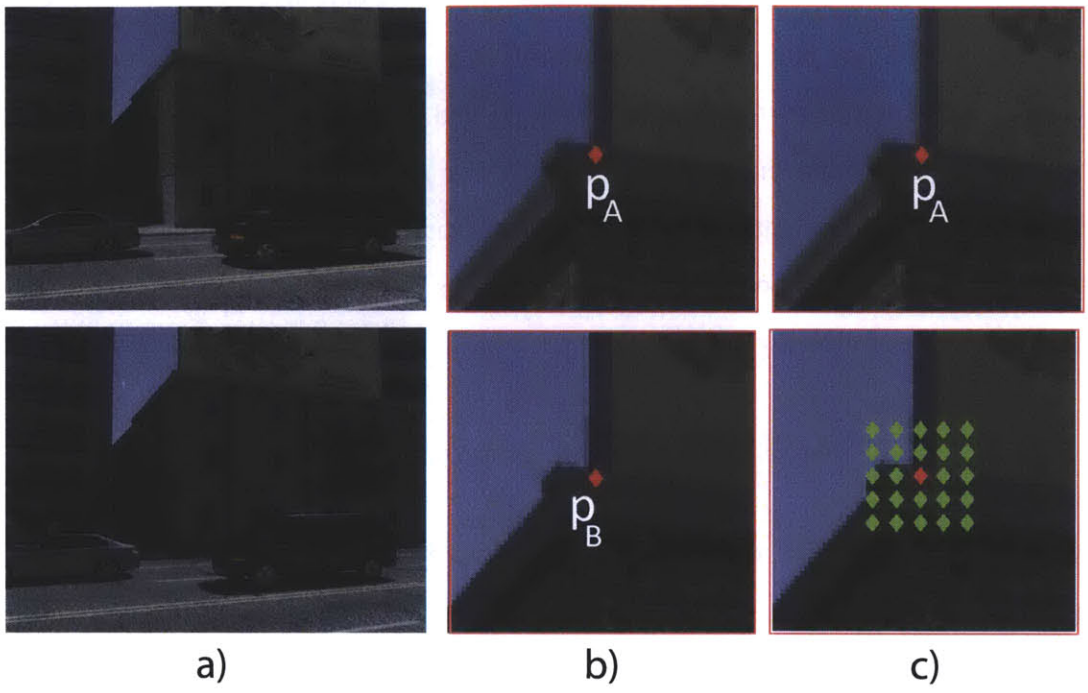



**Figure 5.1:** *a) Images **A** (top row) and **B** (bottom row) of the same scene taken at different time of the day. b) Zoomed in portion of the images **A** (top row) and **B**, showing the pixels $p_A$ and $p_B$ (in red) corresponding to the same 3D point. c) The bottom image shows the pixels (in green) in the neighborhood around $p_B$ that are used for comparing the distance in descriptors space with respect to the distance in descriptors space between $p_A$ and $p_B$ .*

interest point detectors.

## ■ 5.2 Defining good interest points

In this section, we define the notion of a good interest point. The basic task in which local features are used is to be matched across different images. A point $p$ in the image is described by the vector $D(p)$ based on some transformation of the local patch around $p$. Given an image pair **A** and **B** (fig. 5.1 a)), we want to find a set of corresponding 3D points in the two images, $p_A$ and $p_B$ (fig. 5.1 b)), such that they are distinctive with respect to their local neighborhood and can be matched uniquely using the de-

scription $D$. We first compute the description $D(p)$ at every pixel $p$ in the two images. Then for each pair of corresponding points, $p_A$ and $p_B$, we compute the Euclidean distance, $E(p_A, p_B)$, between $D(p_A)$ and $D(p_B)$. For $p_A$ to be an interest point, we want $E(p_A, p_B)$ to be sufficiently smaller than the distance in descriptor space from $p_A$ to points in the neighborhood around $p_B$. Then for a point $p_A$ to be an interest point, it must satisfy the following criteria:

$$E(p_A, p_B) < E(p_A, q_B) \quad | \quad \forall q_B \in W, q_B \neq p_B \tag{5.1}$$

where $W$ is a square grid of size $n$ around $p_B$ (fig. 5.1 c)). Since the descriptor is computed over a local patch around the point, often 20 to 40 pixels in width, neighboring points are likely to have similar description. Therefore, we sample the neighborhood around $p_B$ at 5x5 grid at 5 pixel offset.

To ensure distinctiveness and reduce ambiguities, we need a measure of how different a point is from its surrounding neighborhood. We fit a Gaussian $\mathcal{N}(\mu, \sigma^2)$ to $\{E(p_A, q_B) | \forall q_B \in W, q_B \neq p_B\}$. Then, for each pair $(p_A, p_B)$ that satisfies equation (5.1), we compute the standard score based on this distribution.

$$Z_{p_A} = \frac{E(p_A, p_B) - \mu}{\sigma} \tag{5.2}$$

This gives us a measure of how distinctive $p_B$ is from the other points in its local neighborhood. We consider only points with $Z_{p_A} < z_{thresh}$ as possible candidates for good interest points where $z_{thresh}$ is a parameter that can vary depending on the number of interest points required by the application.

After we compute the standard score at each pixel, we can create a score map for the image as shown in the top row of figure 5.2 a). Notice that there are clusters of points that have low scores. In general, points that are close by will be hard to match,

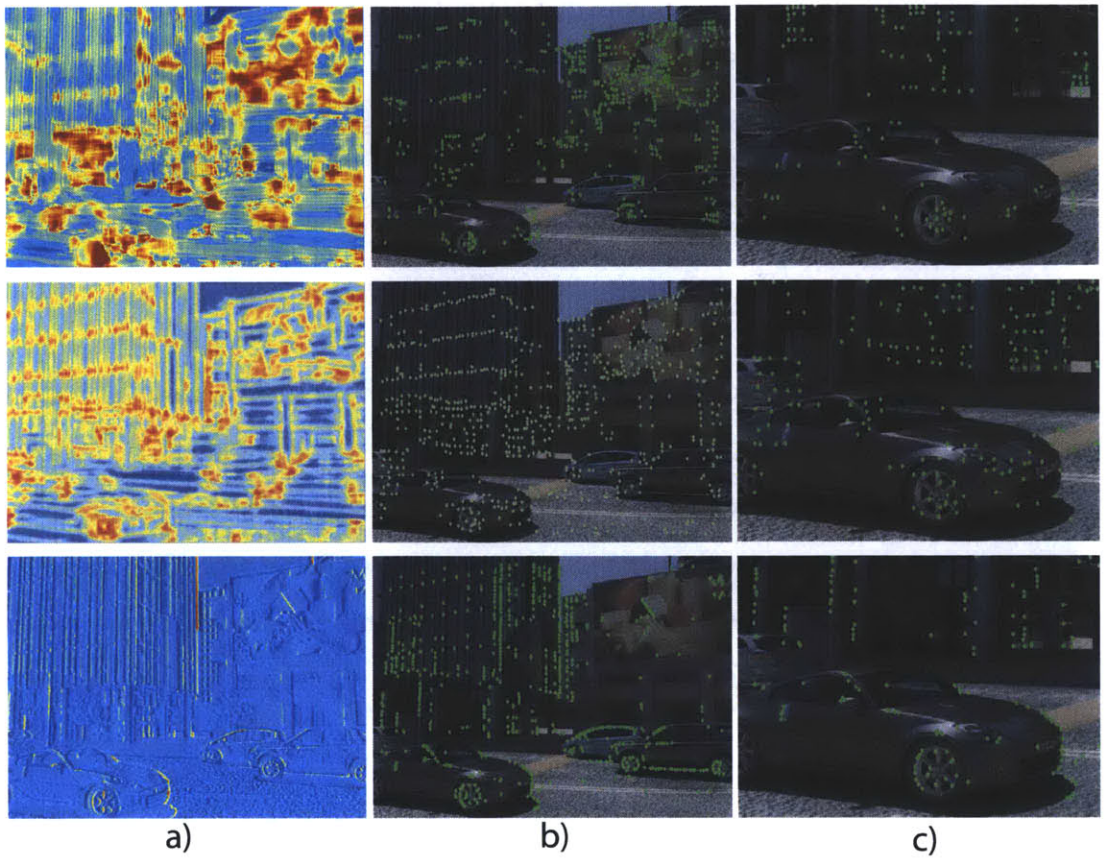



**Figure 5.2: Top row:** *a) The standard score map for good points based on a pair of images of the same scene taken under different illumination. The color map is from blue to red and indicates low to high information content respectively. b) 700 points selected based on the score map in a). Note that smooth regions, e.g. the sky and road have no points. The regions around the cast shadows from the building and the cars also do not contain points since they cannot be matched in images under different illumination. Most of the selected points are of textured or corner like regions. c) A close up of the image in b).* **Middle row:** *a) The classifier score map for the same image based on the model using the SIFT descriptor features. Note that it is fairly similar to the standard score map above. b) 700 points detected by the model. The distribution of the points is more uniform but with the exception of detecting some points on the road surface, it finds points in similar regions as the points in a). c) A close up of the image in b).* **Bottom row:** *a) The classifier score map for the same image based on the model using simple features. b) 700 points detected by the model. It favors points on edges. c) A close up of the image in b).*

so a good detector will produce points that are not crowded together. To ensure that the points are sampled in more uniform manner across the image and not clustered together, we apply a non-minimum suppression to the score map and only consider points that satisfy the condition:

$$Z_{p_A} < \min\{Z_{q_A} | \forall q_A \in W, q_A \neq p_A\}, \tag{5.3}$$

where W is a 3x3 window around $p_A$.The top row of figure 5.2 b) shows a set of interest points selected by the above method. These points are distinctive and can be matched uniquely with respect to a local neighborhood of points given the descriptor D.

Finally, we want to ensure that the points selected by the above procedure are found in both images **A** and **B**, i.e. they are stable. We perform the above selection procedure to both image A and B and then we prune the set of interest points to only those that appear as good interest point candidates in both images.

## ■ 5.3 Learning an interest point detector

The benefit of a dataset of complex scenes with ground truth correspondences is that we can easily generate a set of good interest points. Moreover, the dataset we used is large enough to allow us to employ machine learning techniques to learn an interest point detector for a given pixel description. The dataset provides us with 15600 image pairs - 3600 image pairs of scenes taken from the same viewpoint but under different illumination and 12000 image pairs of scenes taken from different viewpoints both under the same and under different illumination. Using the method described in section 5.2, we produced a training set that contains over 3 million positive and 3 million negative examples of good interest points for the dense SIFT descriptor. We use the SIFT descriptor in our experiments; however, in principle our method can be used for

any choice of descriptor. We then train a classifier that can tell us if a pixel is a good interest point candidate.

### 5.3.1 Features used for learning

We consider using both complex features such as the descriptor used for generating the training set and a set of simple features that are fast to compute.

**Dense SIFT Descriptor** Natural choice of features is the dense SIFT descriptor that was used for generating the training set. We compute the SIFT descriptor at each point in our training set. After initial pre-smoothing of the image by $\sigma = 1.8$, we quantize the gradient orientation at each sample into $d$ directions and bin them in 4x4 spatial grid. Each gradient direction is weighted bilinearly according to its distance to the bin centers. The final descriptor is normalized using a threshold of 0.2 as in SIFT [59]. We used 8 gradient directions and patch size of 21x21 pixels centered at the interest point.

**Simple features** One downside to using a descriptor vector like SIFT is that it can increase the computational complexity of the detector. We considered creating a number of smaller feature vectors from simple features computed at the interest point. These allows for faster classification at detection time.

*Color* The simplest feature is to use the color information itself. We convert the image to CIE L*a*b* color space and use the three components at the given pixel as a feature vector.

*Gradients* Since the SIFT descriptor is based on histograms of gradient orientations, we compute the image gradient at each pixel and create a feature vector $[I_x, I_y, ||\nabla I||, \theta_{\nabla I}]$ where $I_x$ and $I_y$ are the image derivatives and $\theta_{\nabla I}$ is the gradient direction.

*Blobs* Some of the popular keypoint detectors use the responses from the Difference of Gaussian (DoG) [59] or Laplacian of Gaussians (LoG) filters and the determinant of the Hessian [8, 57]. These are generally referred to as blob detectors. We create a feature vector from the filter responses at the highest scale at each pixel.

*Corners* Corner detectors are often used to provide interest points. Here, we create a feature vector from the filter responses of the corner detectors proposed by [37, 43, 50, 91, 115].

### 5.3.2 Training

For training our model, we divided our set into 70% for training and 20% for testing. We use images from different parts of our virtual city for training and testing. Since we used a fly through camera animation to generate the images, neighboring camera locations capture parts of the same scene. To avoid the presence of images of the same city scene in both training and test dataset, we did not use 10% of the dataset. This ensures there is no overlap between the two sets. We split the dataset into three different sets - image pairs 1) taken from the same viewpoint but under different illumination (Light) 2) taken under same illumination but from different viewpoint (Viewpoint) and 3) taken both from different viewpoint and illumination (LV) - and trained a model for each one. For each image pair, we generate examples of good interest points using the method described in section 5.2 and select 800 points for each image. In order to have examples that are strongly positive and strongly negative, we select a random sample of 800 points from the points that have a high standard score $Z_{p_A} >> z_{thresh}$ after applying a non-maximum suppression to the score map. We do this to avoid samples on the boundary between the two set and to get both a more uniform sampling and distinctly negative examples. Since we use the SIFT descriptor to determine the training samples, we do not compute descriptors of points that are at a distance of

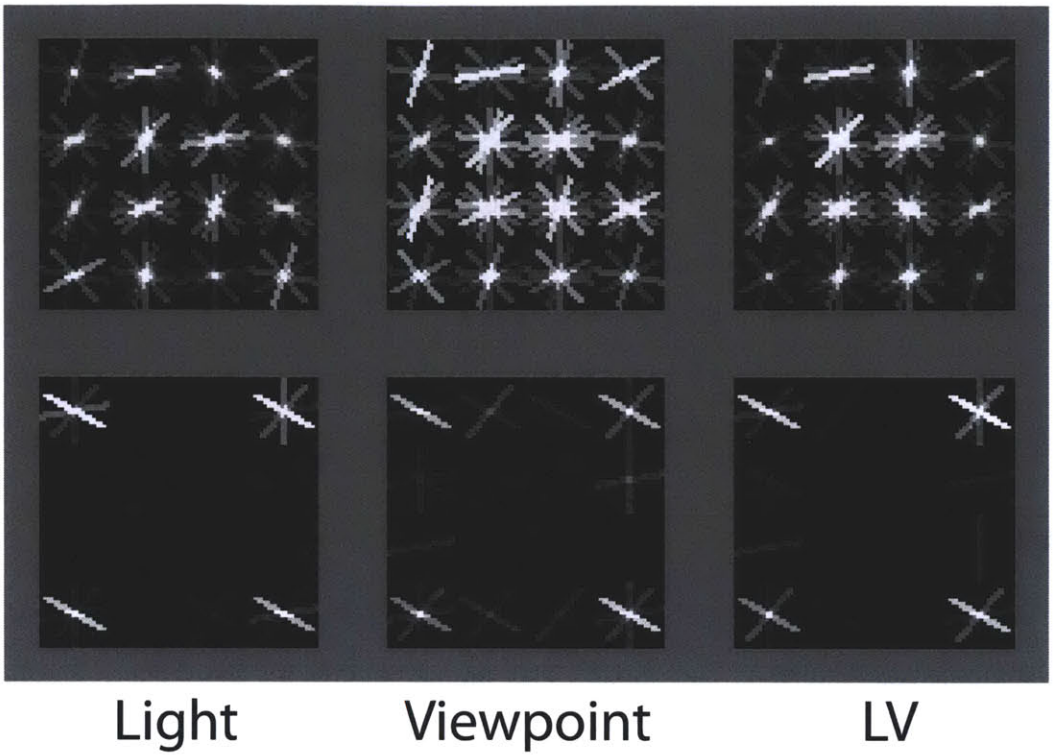



**Figure 5.3:** *Visualization of the SIFT descriptor weights learned by the SVM for the models based on the three different datasets (Light, Viewpoint, and LV).* **Top row:** *Positive weights.* **Bottom row:** *Negative weights.*

n pixels from the boundary where n is half the patch size used from computing the descriptors and therefore we do not use them for testing or training.

We used a linear support vector machine [31] to train models for each of the three training sets using 3.25 million positive and 3.25 million negative examples for the Light and LV datasets and 0.75 million positive and 0.75 million negative examples for the Viewpoint dataset. We trained models using the SIFT descriptor as a feature vector (referred to as the descriptor model) and models using each of the sets of simple features and a combination of all the simple features (referred to as the simple model). The features in our training set were normalized to have zero mean and unit variance

and the same normalization parameters were used to normalize the test data. We chose models with linear kernels because of their faster performance and the large size of our training and feature set since one of the requirements of feature detectors is that they are fast to compute. We set the misclassification cost c to 1. Figure 5.3 shows the learned model weights for the SIFT descriptor with a visualization similar to that of [32]. There seems to be a preference towards many orientations in the center region and fewer toward the edges for the positive weights. The opposite effect seems to be present for the negative weights.

### 5.3.3 Detection

To detect pixels that are good interest points using the learned models, we first need to compute the features at each pixel. The simple features are very fast to compute. Computing SIFT densely is also very efficient. It takes 0.3 seconds for an image of 640x480 pixels in MATLAB and this can be further optimized if necessary. The linear model provides fast classification. We don't use the predicted labels directly as indicated by the sign of $w^T x + b$ where $x$ is the feature vector and $w$ and $b$ are the learned model parameters; instead, we use the value of $w^T x + b$ as a continuous score of how "interesting" a pixel is, a measure of whether it will be a good interest point.

Rows 3 and 5 in figure 5.7 show a visualization of the resulting classifier scores on several test examples for the simple and descriptor models on the dataset with varying viewpoint and illumination (LV). Note that there are many pixels considered as interesting that are clustered together. This is particularly visible in the visualization for the model using the SIFT descriptor as a feature. Therefore, just applying a threshold on the classifier score to determine good interest points is not enough to produce a globally separable set and avoid confusion in the matching stage. Our trained detector produces the final set of interest points by apply non-maximum suppression on

the classifier score map. To handle features at different scales, one could use a similar approach to [66].

## 5.4 Performance evaluation

We use several measures to evaluate and compare the performance of our learned detectors. The widely accepted performance measure of feature detectors is the repeatability criterion proposed by [86]. It signifies the level of robustness of the detector to changes in the imaging conditions by measuring the percentage of 3D points detected in both images. The repeatability criterion, however, is only one side of the coin. Higher repeatability does not necessarily translate to good performance in the feature matching task. In particular, if the detector finds points that are clustered together in certain parts of the image, the repeatability rate will be high but the number of ambiguous matches will be high as well; thereby reducing the usefulness of the detected interest points. Schmid et al. use entropy as a measure of distinctiveness of the interest points based on gray level descriptors. In a similar vein, we propose to measure the performance of the SIFT descriptor using the detected interest points. Better descriptor performance corresponds to more distinctive set of points selected by the detector. Finally, we consider how close the detected interest points are to the set of ground truth points selected by the procedure described in section 5.2.

We compare the performance of our detectors to two of the popular interest point detectors - the Difference of Gaussians (DoG) [59] and the Harris corner [43] detectors. For the DoG detector, we used the implementation by Vedaldi [111] as it allowed us to control the number of points returned by the detector. Instead of varying the thresholds, we select a set number of top ranked keypoints by each detector. This allows us to measure their performance in a comparable way. We found that varying the number of interest points used in our experiments does not change the relative performance of

| Detectors | SIFT | DoG | Harris | Color | Corner | Blob | Gradient | Simple |
|---|---|---|---|---|---|---|---|---|
| Light | 0.317 | 0.265 | 0.363 | **0.437** | 0.272 | 0.309 | 0.387 | 0.380 |
| Viewpoint | 0.304 | 0.481 | 0.518 | 0.433 | **0.520** | 0.506 | 0.496 | 0.495 |
| LV | 0.222 | 0.313 | 0.349 | **0.416** | 0.294 | 0.336 | 0.327 | 0.317 |

**Table 5.1:** *Repeatability rates ($n = 2$) for the DoG, Harris and learned detector models on the three different datasets.*

the detectors. For most of the results, we used a maximum of 600 interest points per image. In the cases of image pairs with viewpoint changes, the number of points in the overlapping region could be lower. In those cases, we used the minimum number of points across all the detectors for the given image.

### ■ 5.4.1 Repeatability rate

The repeatability rate is defined as the number of corresponding 3D points detected in both images **A** and **B** with respect to the total number of detected points. To determine the repeated points, we need to consider only points detected within the region of overlap between the two images. Moreover, due to ambiguities in the detection we say that a point is repeated if it is found within a neighborhood of size $n$ around its true correspondence. Since the number of points detected in both images may vary, we consider the total number of detected points to be the minimum between the number of points detected $N_A$ and $N_B$ in **A** and **B**, respectively.

$$Repeatability = \frac{\#\text{ of correspondences}}{min(N_A, N_B)} \tag{5.4}$$

Note that because we are using a synthetic dataset, we have true correspondences between pixels and can determine the region of overlap even for non-planar scenes unlike [86].

We computed the repeatability for DoG, Harris and the learned detectors on each

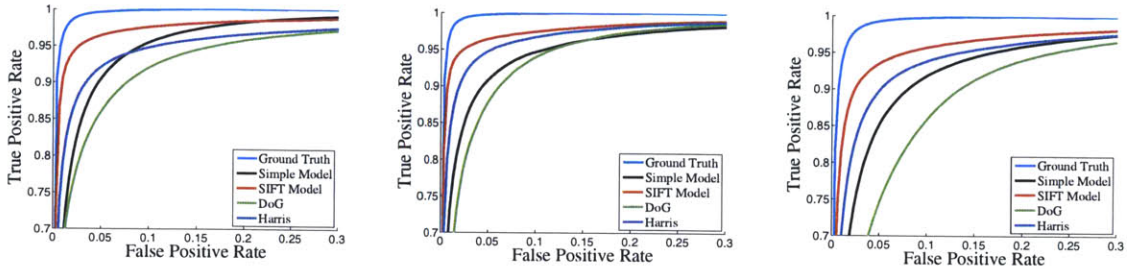



**Figure 5.4:** *ROC curve for the SIFT descriptor performance. a) Lighting dataset. b) Viewpoint dataset. c) Viewpoint and Lighting dataset.*

of the three datasets. Table 5.1 shows the repeatability rates for all the detectors for $n = 2$. We found that the simple detector model has similar repeatability to the DoG and Harris detectors while the descriptor detector has lower repeatability than Harris and DoG for the Viewpoint and LV datasets. The model based only on the color features had the highest repeatability for the datasets with changes in illumination. Inspecting the results visually, we found that this model resulted in many of the detected points being clustered together.

### ■ 5.4.2 Descriptor matching

Since ultimately the detected points will be used for matching across different images, we evaluate the performance of the models based on their ability to detect points that result in good matches over the set of repeatable interest points. We use an ROC curve to measure the descriptor performance given a set of interest points [118], similar to the measure used in chapter 4. Given two images, we define the matched pairs to be the pairs of keypoints corresponding to the same 3D point and the non-matched pairs to be all the other pairs. We then compute the Euclidean distance between the descriptors of each pair and plot the correctly matched pairs as a fraction of all true matches (true positive rate) against the incorrectly matched pairs as a fraction of all the non-matching

pairs (false positive rate).

$$\text{True Positive Rate} = \frac{\#correct\ matches}{\#matching\ keypoints} \tag{5.5}$$

$$\text{False Positive Rate} = \frac{\#false\ matches}{\#non\text{-}matching\ keypoints} \tag{5.6}$$

The larger the area under the ROC curve, the better the performance of the descriptor. In our case, the descriptor performance is dependent on the interest point selection. Therefore, good descriptor performance corresponds to better interest point detection.

Figure 5.4 shows the ROC curves for the SIFT descriptor performance using sets of interest points from different detectors on the three dataset. We only plot the performance of two of the trained models for readability. We also plot the performance of the ground truth interest points found by the method described in section 5.2 to make sure that we indeed select a reasonable set of points. Our selection of ground truth interest points is based on the distinctiveness of a point with respect to the local neighborhood around it, therefore it is possible to select points that could be ambiguous with respect to other parts of the image. In practice, it seems that the locally distinctive criterion works well in selecting points that are likely to be distinctive globally as well. The descriptor model performs best in the matching task. We report the results for the other models and datasets in table 5.2 in terms of the 95% error rate, that is the percentage of false matches when 95% of all correct matches are detected. The descriptor model has the lowest error rate on all three datasets; however, the LV dataset including both viewpoint and illumination changes appears to be more challenging. Note that the model based on color features only performs very poorly despite its high repeatability rate. The simple model and the model using only the gradient based features have a similar performance and it is significantly better than that of the other simple features. This seems reasonable given that the training set for the models was acquired using a

| Keypoints | GT | SIFT | DoG | Harris | Color | Corner | Blob | Gradient | Simple |
|---|---|---|---|---|---|---|---|---|---|
| Light | 0.010 | **0.030** | 0.174 | 0.110 | 0.104 | 0.278 | 0.231 | 0.097 | 0.094 |
| Viewpoint | 0.007 | **0.027** | 0.120 | 0.054 | 0.234 | 0.161 | 0.134 | 0.117 | 0.110 |
| LV | 0.013 | **0.080** | 0.231 | 0.134 | 0.552 | 0.258 | 0.291 | 0.177 | 0.167 |

**Table 5.2:** *Error rates at 95% correct detection in the SIFT descriptors matching for the DoG, Harris and learned detector models on the three different datasets. The first column shows the performance of the SIFT descriptors using the interest point selected by the procedure described in section 5.2.*

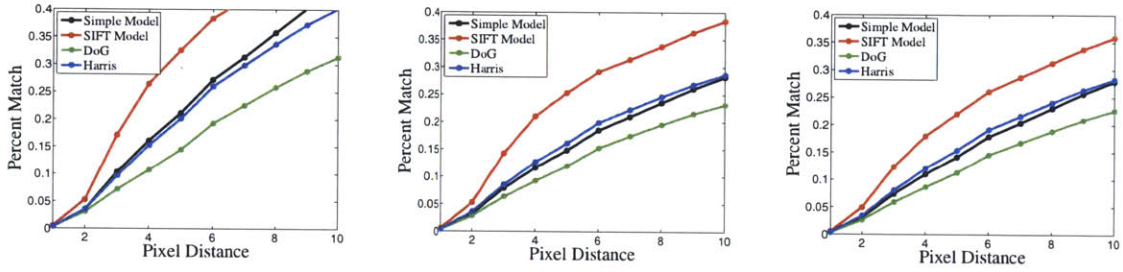



**Figure 5.5:** *Repeatability rate between detected and ground truth interest points. a) Lighting dataset. b) Viewpoint dataset. c) Viewpoint and Lighting dataset.*

descriptor based on histograms of gradient orientations.

### 5.4.3 Comparison to ground truth

A natural question to ask is: how similar are the detected interest points with respect to the ground truth set acquired by the method in section 5.2? We used two different measures to answer this question. First, we compute the repeatability rate between the detected points and the ground truth points as a function of the neighborhood size $n$. Figure 5.5 shows the results for several of the detectors on the three datasets. The repeatability rate is fairly low for all of the detectors but the descriptor model detects more interest points that are within the neighborhood of the ground truth set. In our method for selecting good interest points, there are other potential candidates before applying the non-minimum suppression that also could be good points if not selected

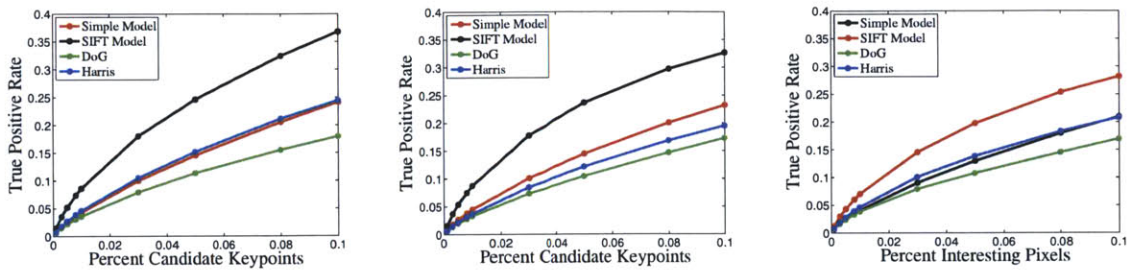



**Figure 5.6:** *ROC curve for the percent detect points in the set of potential interesting point candidates. a) Lighting dataset. b) Viewpoint dataset. c) Viewpoint and Lighting dataset.*

in clusters.

Another way to measure how well a detector can predict the distinctiveness of a point is to consider the percentage of detected interest points within the areas of potential interest points in the standard score map produced by our point selection procedure. We determine those areas by using different values for $z_{thresh}$ and selecting the pixels that satisfy $Z_{p_A} < z_{thresh}$. Note that as the percentage of the image considered to be interesting goes to 100%, so will the predictability of the detected interest points. Figure 5.6 shows the resulting ROC curves. The performance of all the detectors is above chance and the relative performance is the same as the repeatability rate. However, there are still many points detected that are not among the most distinctive features locally.

### ■ 5.4.4 Visual results

We show a visual comparison of the detected interest points for the descriptor and simple models trained on the LV dataset and the DoG and Harris detectors (fig 5.7). The first two columns show examples from the test set in our database. The last column is of a real image from a street in Valladolid. Each of the detectors was given a budget

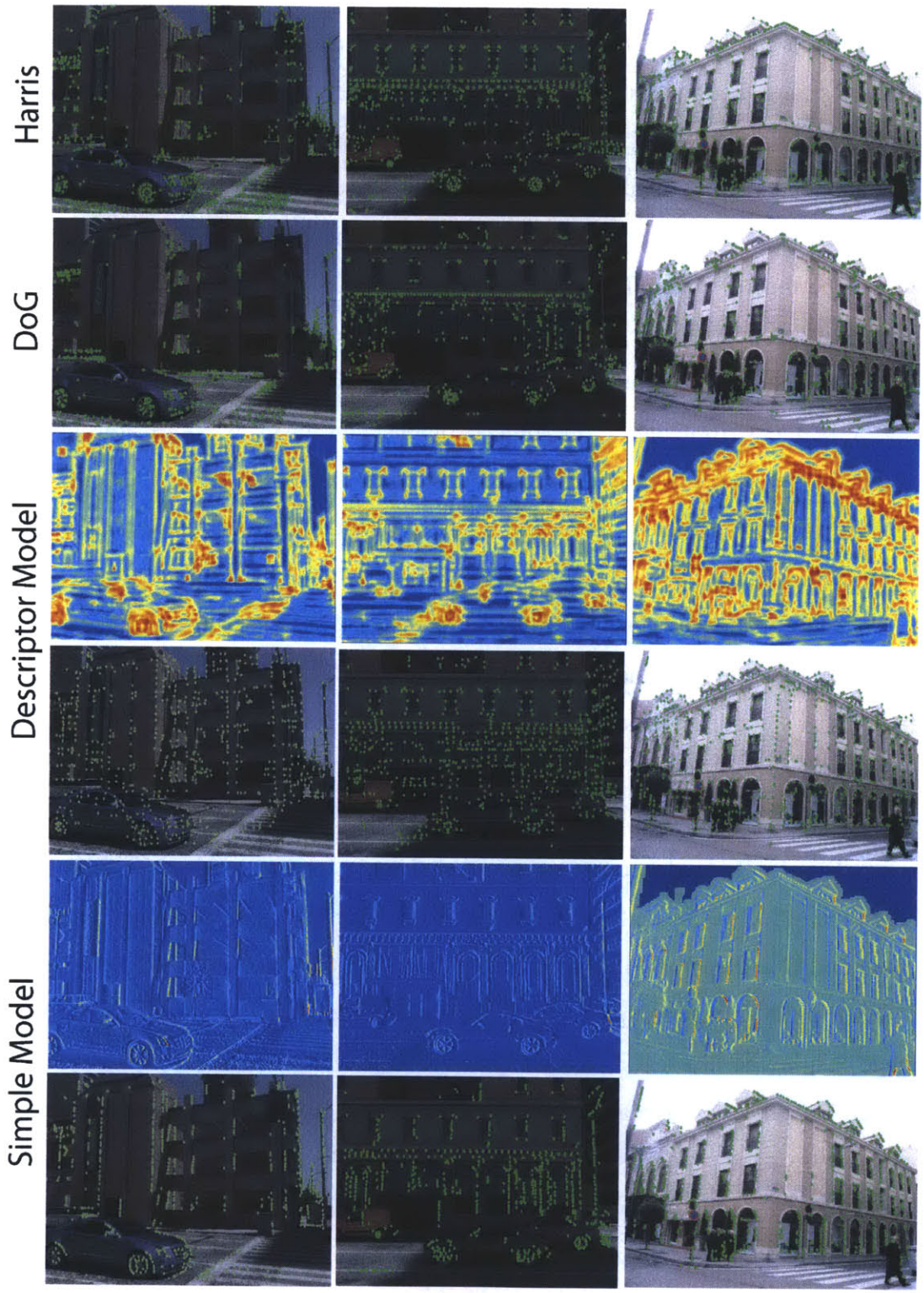



**Figure 5.7: Visual results (LV dataset)** *Each column shows the performance of the Harris, DoG, descriptor and simple trained model detectors trained on the LV dataset. The first two columns are examples from the synthetic test set. The last example is from a real image from a street in Valladolid. The Descriptor model produces more globally separable set of points and has preference for corner like regions. The simple model prefers strong edges.*

of 600 pixels and the neighborhood around the detected points is marked in green. Note the strong shadows on the car in the first example in the first column, our models and in particular the descriptor model have learned that despite the strong edge there, these are generally not good interest points. More specifically, it has learned to ignore edges with large gradient magnitude and, in particular, horizontal ones, as they are likely to be shadow edges. The points detected by the descriptor model have the global separability property. There are almost no clusters of points unlike the results produced by the DoG and Harris detectors. This could explain both the lower repeatability rate and the better performance in the descriptor matching task. If points are clustered together in a region, it is easier to achieve a high repeatability rate than if they are more uniformly distributed across the image. Clustered points, however, makes it harder to match features uniquely.

Visual inspection suggests that the descriptor model has a strong preference for corner like regions and finds more of those than either DoG or Harris detectors. It also has learned that smooth regions with little texture are not good interest points. Naturally, our definition of an interest point was made using a descriptor based on histograms of gradient orientations and it does not provide a distinctive description of uniform regions. The simple model seems to have a preference for strong edges. This is likely due to the influence of the features based on the filter responses for image gradients, corner and blob detectors since they largely depend on the local edge strength. It also suffers from the clustering problem. The trained models work well on the real world image despite the fact that it is overexposed compared to the images in our synthetic dataset. Note that we have no people in our virtual city dataset but the descriptor model finds points on the people especially ones with corner like appearance.

Figures 5.8 and 5.9 show qualitative results from the descriptor and simple models trained on the light and viewpoint datasets respectively. The descriptor model trained on the viewpoint dataset is able to isolate corner like regions with greater confidence

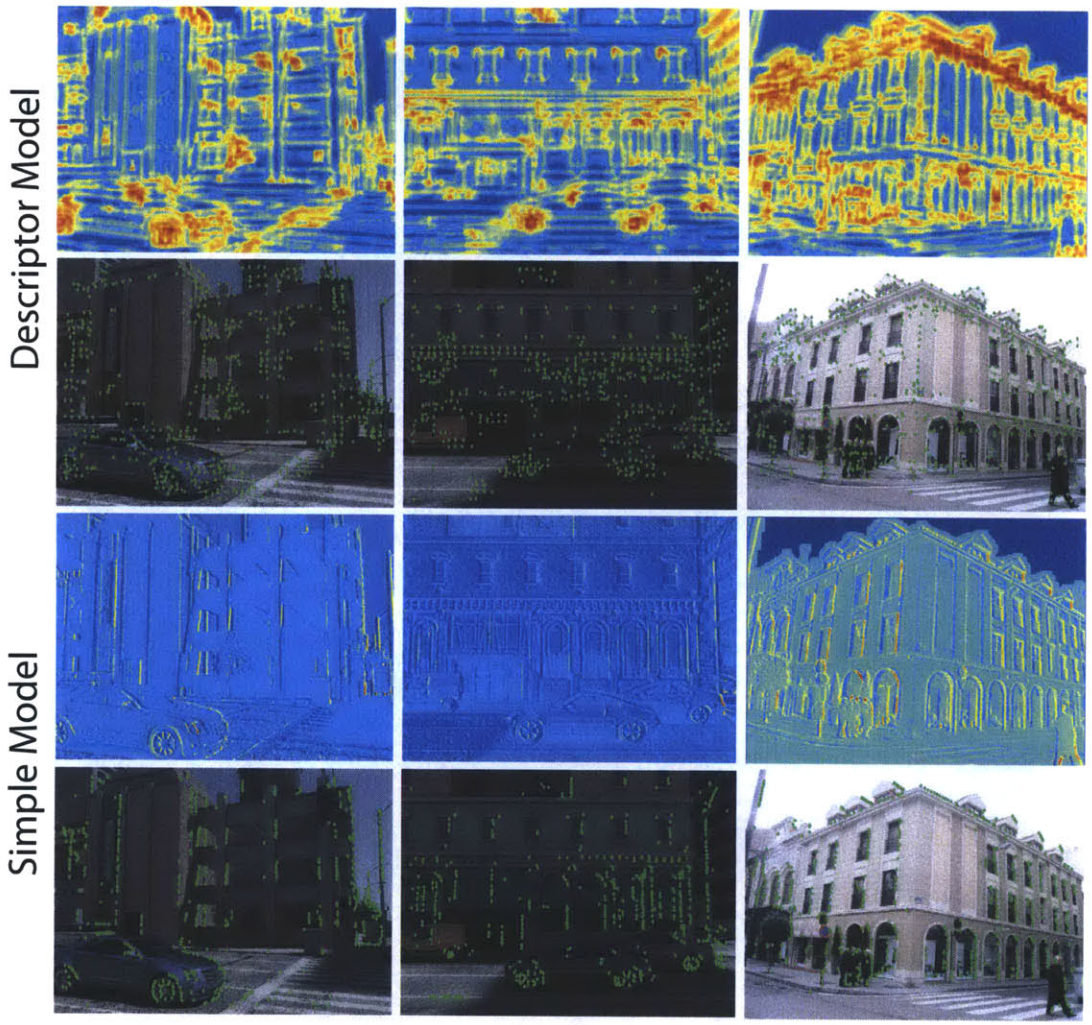



**Figure 5.8: Visual results (Light)** *Each column shows the performance of the descriptor and simple model detectors trained on the Light datatet. The first two columns are examples from the synthetic test set. The last example is from a real image from a street in Valladolid. The Descriptor model produces more globally separable set of points and has preference for corner like regions. The simple model prefers strong edges.*

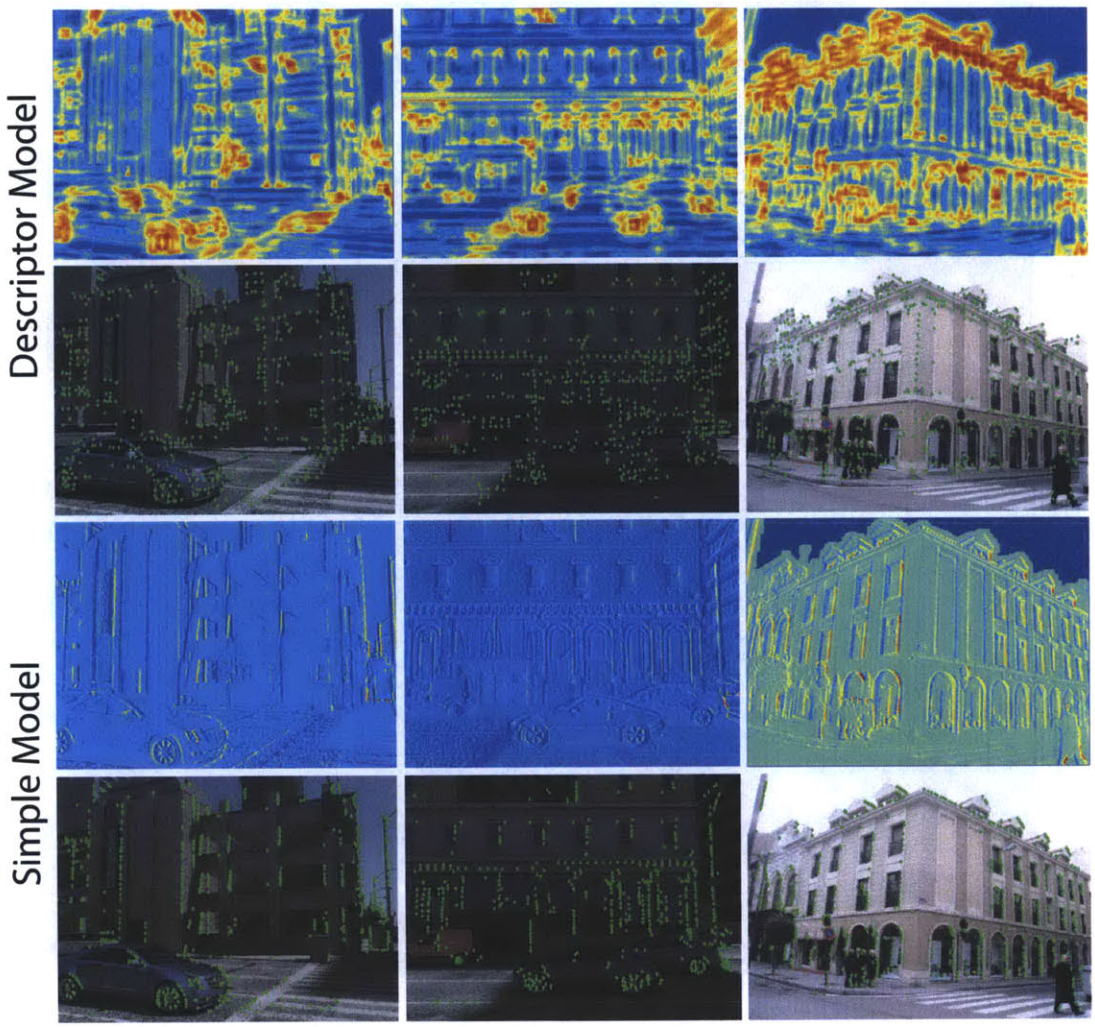



**Figure 5.9: Visual results (Viewpoint)** *Each column shows the performance of the descriptor and simple model detectors trained on the Viewpoint dataset. The first two columns are examples from the synthetic test set. The last example is from a real image from a street in Valladolid. The Descriptor model produces more globally separable set of points and has preference for corner like regions. The simple model prefers strong edges.*

(see the fig 5.9 top row middle image) while the one trained on the light dataset has fewer clusters close to the edges of the shadow regions (see the fig 5.8 top row middle image).

## 5.5 Conclusion

We proposed a method for learning a feature detector that would optimize the performance of a given descriptor in the matching task. We presented a procedure for selecting a set of good interest points based on a given descriptor from image pairs with ground truth pixel correspondences. In our experiments, we used a dataset of 15600 image pairs from a highly photorealistic virtual city taken under different viewpoint and illumination. Using the dense SIFT descriptor, we generated a set of over 6 million positive and negative examples and trained a model for interest point detection using a linear support vector machine. For training, we used two sets of features: the SIFT descriptor itself and a set of simple features based on color, image gradients and the response from filters used in corner and blob detection at the pixel location of the interest point. We found that the model trained using the descriptor as a feature performed better in all of our experiments than the one using the simple features; however, even the latter had performance comparable to that of the DoG and Harris detectors on our set.

We trained and tested the models on three different datasets of image pairs: with illumination variation only, with viewpoint variation only and with both viewpoint and illumination changes. We found that the model trained using the SIFT descriptor as a feature vector had a lower repeatability rate for the datasets with viewpoint changes in comparison to the DoG and Harris detectors, but a much better performance in the actual matching task. From visual inspection of the selection points, we found that DoG and Harris have more of a tendency to cluster points in interesting regions while

our model seemed to produce a more globally separable set. This could explain both the higher repeatability rate and the lower performance in the descriptor matching task of the DoG and Harris detectors. If points are clustered together, it is more likely to find repeatable matches but also harder to match them unambiguously.

We also show results of our detector on an image from a city in the real world with different imaging conditions and visually the nature of the detected interest points compared to those on our synthetic test images is very similar. Both trained models learned that smooth regions do not present distinctive features for the SIFT descriptor. The simple model has a particular preference for edges that comes from the gradient features while the descriptor model seems to prefer corner like features.

# Chapter 6

# Looking beyond image boundaries

## 6.1 Introduction

HUMANS are much better than computers in analyzing and understanding the content of images. In fact, humans can also reason about the 3D world that extends beyond the image boundaries. Our past experience, coupled with the regular nature of the world, allows us to make such predictions. Our goal here is to allow computers to make similar predictions.

Being able to predict the world beyond image boundaries has many potential applications. One example would be a robot moving through the environment. Such a robot can improve its motion planning capabilities if it can predict how the world would look from a new point of view before actually moving there. The ability to extend the field of view beyond the image boundaries can also be useful in image editing applications where, for artistic purposes, we might need to extend the field of view of a given image. In addition, the prediction task provides another way to ask a question of importance in the object recognition community, how well can context predict image content [27, 102]?

There have been informal studies of matching or prediction for individual applications [44, 92], but systematic study of these properties have not been undertaken. In the past, large image databases were used to fill in holes in an image [44]. The

evaluation of [44] did not address the tasks of matching or prediction, and the method was user-assisted, not automatic as ours is. Prior work on multi-perspective panoramas [2, 77, 119] focuses on using large number of images all taken from the same location but from different view points and stitching them together. Similarly, the PhotoTourism interface [96] places a large collection of images of one particular geographic location in a common 3D space. In contrast, we use a large collection of images that do not necessarily come from the same location in the real world and we explore the regularities present in them to make predictions.

### 6.1.1 Matching and Prediction

Our working assumption is that a large database of images can be used to synthesize the world beyond the boundaries of a given image. To do this, we introduce the notion of transformed image retrieval. In a traditional image retrieval system the query image is used as is to query the database. In contrast, we first apply a 2D transformation to the query image that simulates the desired camera motion and use the transformed image to retrieve matching images that extend beyond the boundaries of the original query image. For example, suppose we want to see the world to the right of the given query image. We first create the transformed query image as in Fig. 6.1 and then search the database matching only based on the known part of the transformed query image to the images in the database. We then use the best matched image to extend the query image to the right.

In our experiments, we focus on a database of outdoor city scenes from many different parts of the world that were downloaded from various photo sharing websites. To measure how well the system works, we use a ground truth database synthesized from 10,000 geo-referenced images from the Google Street View data set of Pittsburgh [40].

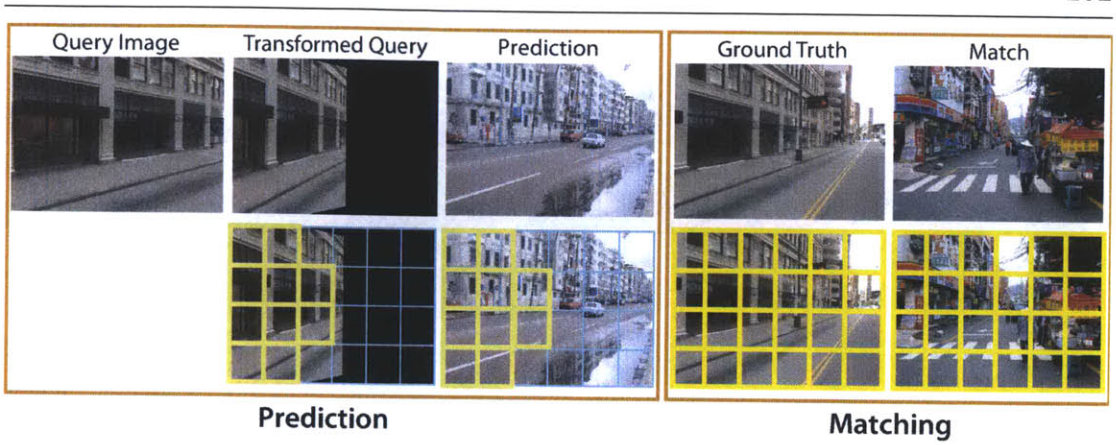



**Figure 6.1: Prediction and matching.** *Top row: Query Image. Transformed query image for rotate right motion. The prediction obtained by finding the nearest neighbor using the transformed query (the prediction task). Ground truth (the actual view seen after the camera rotated from the query image). Top match to the ground truth image (the matching task). Bottom row: The transformed query image and the top prediction image retrieved with the bins used for retrieval overlaid. The ground truth and the top match images and the bins used for the retrieval overlaid.*

During system evaluation, given the image database and the ground truth Pittsburgh street view data set, we study the capabilities of both matching and prediction.The matching task asks how well a collection of test images can match a dataset of images, and indicates the coverage of a set of test images by the reference database. The more interesting question is: how well can images from one camera position and orientation predict the image taken by a camera at a neighboring position or orientation? This is the *prediction* task.

In the matching task, we ask how well can we explain street-level images from Pittsburgh from outdoor images taken in any city other than Pittsburgh? In the prediction task, we ask what regularities are in common across images of Pittsburgh and of other cities in the world such that we can predict images of Pittsburgh under different camera transformations? We conduct large scale experiments to evaluate both the matching task and the prediction task using various image representation meth-

**Figure 6.2:** *Examples images from our Flickr database.*

ods, various database sizes and various camera motions (for prediction). We evaluate the quality of the matches and predictions in a user study performed using Amazon's Mechanical Turk web service.

## ■ 6.2 Image database

We collected a dataset of more than 6 million images from Flickr by querying for suitable image tags and relevant groups. About 1 million of those were of city scenes. We queried for particular locations in conjunction with tags such as 'New York street', or 'California beach', and also downloaded images from Flickr groups returned by search on terms relevant to particular themes, such as 'Landscape' or 'Forest'. The typical resolution of the downloaded images is $500 \times 375$ pixels. One million jpeg compressed images takes about 120GB of hard-disk space. Currently, we only consider images in landscape format with aspect ratio close to 4:3. Figure 6.2 shows example

images from our database.

### 6.2.1 Organizing images into themes

The issue of semantic mis-matches in retrieved images is especially significant in the case of transformed image retrieval where the information available for matching is weaker than the original scene descriptor (due to the smaller image overlap after the transformation). This can result in semantically incoherent transitions between images.

To augment the scene descriptor, we train a classifier to identify images within some semantic theme. The theme of an image is generally invariant to camera motions. We can dramatically improve retrieval results by matching only pictures that belong to the same theme. This also lowers the memory and CPU requirements as only part of the database needs to be searched. Examples of themes we consider in this work are shown in figure 6.3.

To obtain a visually and semantically coherent set of images for each theme we train theme classifiers from manually labeled training data in a manner similar to [13, 54]. For each theme we train 1-vs-all non-linear Support Vector Machine classifier [87] from about 1,000 positive and 2,000 negative training images. We have developed a suitable labeling interface so that the classifier can be trained interactively. At each iteration, the most uncertain images are shown to the user for labeling. The interface also allows the user to visually assess the classifier performance and label additional images if needed.

## 6.3 Matching and prediction

We evaluate the performance of our system using different image representations described in this section. In the matching task, we use traditional image retrieval employing the entire scene descriptor in the search process. In the prediction task, we


**Figure 6.3:** *Example of images belonging to different scene themes. Partitioning a large collection of images improves the quality of the results. The classification is done automatically. When navigating through the image collection, it is important to keep the themes constant when moving from one picture to another to avoid undesired transitions. The user can also allow transitions across themes.*

use transformed image retrieval where we first apply a 2D transformation to the query image and then we use only the valid portion of the scene descriptor for query in the database. This process is described in detail here.

### ■ 6.3.1 Image Representation

Here we describe the different image representations we investigate for the matching and prediction tasks. Similar representations were used by [121] for the tasks of scene detection and classification.

**i. GIST descriptor:** The GIST descriptor has been shown to work well for scene classification [71]. It measures the oriented edge energy at different scale levels aggregated into coarse spatial bins. We create a 480 dimensional feature vector by first converting

the image to grayscale and then dividing it in 6x4 bins and applying 8, 8, and 4 orientation filters at three difference scales (coarse to fine) to each bin. The GIST descriptor for each image is normalized to have unit L2 norm.

**ii. Histograms of oriented gradients (HOG):** The HOG descriptor [24] and its variants [32] have demonstrated excellent performance for object and human detection. Similar to the SIFT [59], the HOG descriptor measures histograms of image gradient orientation at a coarse grid of spatial locations. First, HOG descriptors are densely extracted on a regular grid at steps of 8 pixels using the code available online provided by [32]. This gives a 31-dimension descriptor for each node of the grid. Then, 2x2 neighboring HOG descriptors are stacked together to form a descriptor with 124 dimensions. The stacking has overlapping on the grid, and the 124-dimension descriptors are quantized into 300 visual words by k-means. The quantized descriptors are then spatially binned into a coarse grid of 6x4 bins, similar to the GIST descriptor. The resulting histogram is normalized to have unit L1 norm.

**iii. The self-similarity descriptor (SSIM):** The self-similarity descriptor [89] has been shown to perform well on matching objects of similar shape but vastly different local appearance. The idea is to represent the appearance in a local image area around a particular image patch by the "correlation map" of the patch with its neighborhood. The descriptor captures the local pattern of self-similaritiy. Such self-similarity patterns can then be matched despite a very different appearance of the central patch. We employ the self-similarity descriptor for scene representation.

The self-similarity descriptors are computed on a regular grid at steps of five pixels. Each descriptor is obtained by computing the correlation map of a 5x5 patch in a window with radius equal to 40 pixels, then quantizing it in 3 radial bins and 10 angular bins, obtaining 30 dimensional descriptor vectors. The descriptors are then quantized into 300 visual words by k-means. Similar to HOG, the self-similarity descriptors are

spatially binned into a coarse grid of 6x4 bins, and normalized to have unit L1 norm.

**iv. Dense SIFT:** Densely extracted SIFT descriptors have been shown to work well in both object and scene recognition [54]. Here, we extract the descriptor on a regular grid at steps of 5 pixels using a window at two scales (4x4 and 8x8). The descriptors are stacked together for each HSV color channel and quantized into 300 visual words by k-means. The quantized descriptors are then spatially binned into a coarse grid of 6x4 bins, similar to the GIST descriptor. The resulting histogram is normalized to have unit L1 norm.

**v. Geometric context (GC):** The geometric context classifiers [46] estimate the coarse geometric layout of natural scenes with respect to the camera. It has been shown to work well as a spatial prior for object detectors [47]. Here we investigate it for the task of scene matching and prediction. We use only the (i) ground, (ii) sky, (iii) vertical and (iv) porous classes as they are more reliably detected. We reduce the probability outputs of each of the four geometric classes to 32x32 pixels resulting in a 256-dimensional descriptor. The resulting descriptor is normalized to have unit L2 norm.

**vi. Tiny images (Tiny32):** As a baseline we also include the tiny image representation of Torralba *et al.* [103]. The most trivial way to match scenes is to compare them directly in color image space. Reducing drastically the image dimensions makes this approach more computationally feasible and less sensitive to exact alignment. This method of image matching has been examined thoroughly [103] for the purpose of object recognition and scene classification. Each of the RGB channels is subsampled to 32x32 pixels resulting in a 3,072 dimensional descriptor. The images are then normalized to have unit L2 norm.

**vii. Texton histogram:** Textons as elementary units in image analysis were introduced in [63]. We use a 512 entry universal texton dictionary built by clustering the responses

of filters at 8 different orientations, 2 different scales and elongations. We then build a 512-dimensional histograms spatially binned into a coarse grid of 6x4 bins by assigning the result of the filter responses at each pixel to the nearest texton dictionary entry. The histogram is normalized to have unit L1 norm.

**viii. Color histogram:** We compute color histograms in CIE L*a*b* color space with 4, 14, and 14 bins in L, a, and b, similar to [121]. The histograms are then spatially binned into a coarse grid of 6x4 bins and normalized to unit L1 norm.

Next we describe the matching and description procedures.

### 6.3.2 Nearest neighbor image matching

In the matching task, the goal is to find the best matching image in the street scene database given the query. Recently, there has been some success in semantic scene matching using nearest neighbor techniques in large databases of millions of images [44, 103] and we follow this work here. The matching is performed using an exhaustive nearest neighbor search, though faster but approximate indexing techniques could be used [116]. For the (normalized) GIST, tiny image and geometric context descriptors we use the Euclidean distance. For the HOG, self-similarity, dense SIFT, texton and color histograms we use the $\chi^2$ distance.

We also investigate a combination of all descriptors (denoted All). This is achieved by a simple weighted average of the distances using individual descriptors. Here we investigate uniformly set weights, but weights learned on a separate training set can also be used [6]. Uniform weights are a reasonable choice in our case, as the range of distances for each of the normalized descriptors is between 0 and 1.

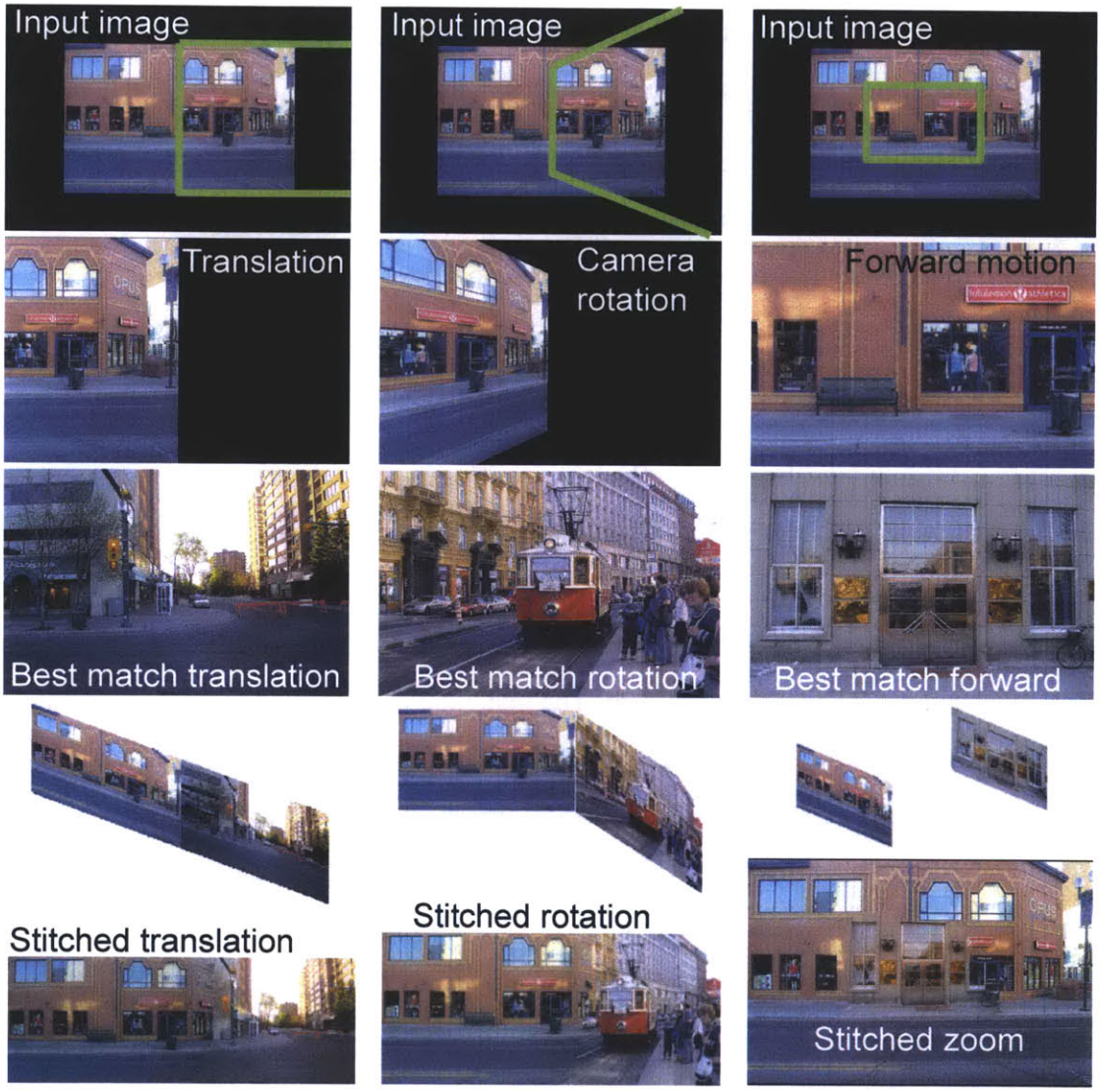



**Figure 6.4:** *Scene matching with camera view transformations. First row: The input image with the desired camera view overlaid in green. Second row: The synthesized view from the new camera. The goal is to find images which can fill-in the unseen portion of the image (shown in black) while matching the visible portion. The third row shows the top matching image found in the dataset of street scenes for each motion. The fourth row illustrates the induced camera motion between the two pictures. The final row shows the composite after Poisson blending.*

### 6.3.3 Transformed image retrieval

In the prediction task, we would like to predict what we will see if we move the camera. Given a query image, we want to find possible candidates describing the parts of the scene not captured by the camera. We extend traditional image retrieval to find matches that simulate a particular camera motion without explicitly modeling the geometry of the scene. Instead, we apply 2D transformations to the query image. We achieve horizontal camera rotation by warping the image using the appropriate homography [98](page 11)

$$H = KRK^{-1}, \tag{6.1}$$

where K is the internal calibration matrix and R the camera rotation matrix. For specifying K, we set the unknown focal length to be half the image width. For specifying R we assume no vertical or in-plane rotation. The 3D zoom motion (or forward/backward motion) is approximated by scaling the image and camera translation is approximated by a translation in the image plane, ignoring the parallax effects. The camera motions are illustrated in figure 6.4.

The transformed query image has missing pixels because not all the information of what the camera would see after the transformation is available in the original query image. Therefore, we can only use a subset of the feature vector to perform the transformed image retrieval (Fig. 6.1). Given the observed portion of the transformed query image, we can now find semantically similar images that approximate the new camera point of view. For translation and rotation, we take about half of the overlapping image. For zoom, we use a scaled version of the image depending on the amount of zoom. The retrieval is performed using the nearest neighbor search as outlined in section 6.3.2 but only using the observed portion of the transformed query image.

We seek to find semantically similar images in the database coarsely matching the geometry of the observed portion of the synthesized view. For example, when the

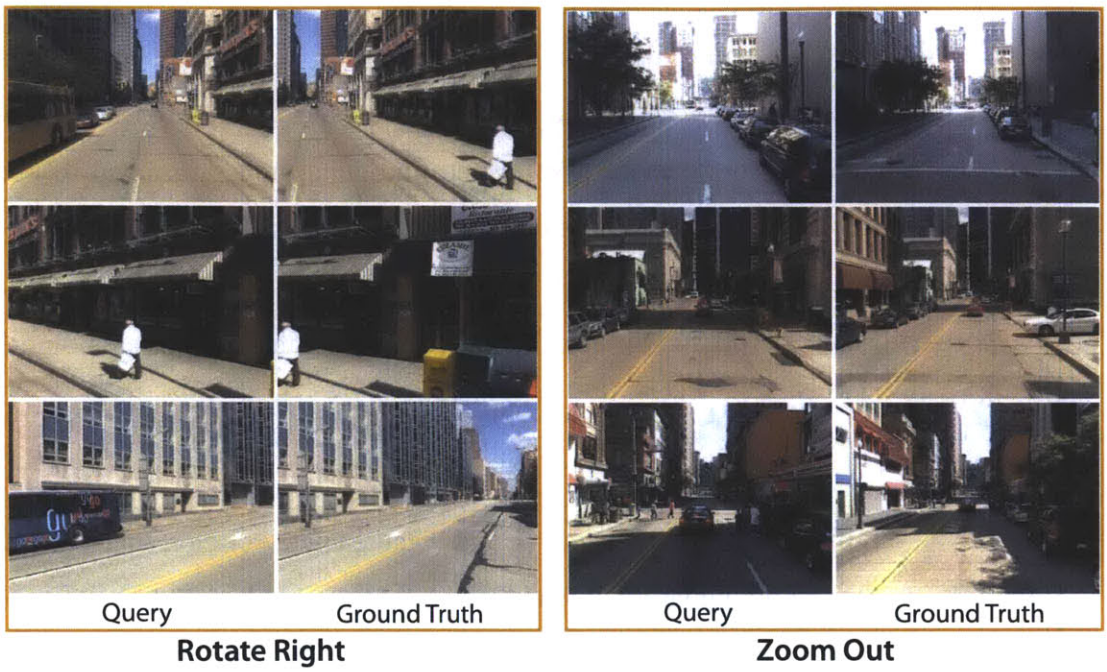



**Figure 6.5:** *Example images from our ground truth dataset for two of the motions - rotate right and zoom out. The figure shows the query image in each pair and the ground truth image that would have been captured after performing the corresponding camera motion.*

camera rotation changes a fronto-parallel view of a building to a view with a strong perspective (as shown in the middle column of figure 6.4), we find most retrieved images depict scenes looking down a street.

## ■ 6.4 Evaluation method

To compare the different descriptors for the matching and prediction tasks we have collected the following ground truth image test set and designed an evaluation procedure based on human judgment of visual similarity. We use the outdoor street city scene for this evaluation.

### 6.4.1 Ground truth dataset

We gathered ground truth data from the Google Street View Pittsburgh data set [40], which is provided by Google for research purposes. The data set contains about 10,000 high-resolution spherical panoramas collected by a moving vehicle for the Street View feature of Google Maps. Each panorama has a field of view of 360 degrees horizontally and 180 degrees vertically. To synthesize images similar to those in our Flickr city database, we unwarp portions of the spherical panoramas simulating a camera with 45 degree field of view with rotation stops at every 22.5 degrees [98]. From the synthesized images, we selected 300 representative pairs for each motion - rotate right and zoom out - for our ground truth data set. Each pair contains the query image for our test set and its corresponding ground truth match based on the camera motion as shown in figure 6.5. Having ground truth data allows us to measure the quality of the different image representations used for matching and making predictions about what the camera would see if it performed a given camera motion.

### 6.4.2 Performance measures

As a performance measure we use human judgment of visual similarity of the matched / predicted image to the ground truth. We sought to evaluate the visual quality of the images retrieved using a variety of different feature descriptors. Rather than assessing pairwise comparisons between each possible descriptor's results, we compared the results from each feature descriptor against a random selection from the image database. This allowed us to evaluate performance of all the descriptors with respect to a common scale and reduced the complexity of the evaluation task.

We designed and conducted experiments using the Amazon Mechanical Turk to compare the performance of the different features for the matching and prediction tasks. We divided each experiment of 300 test query images into 10 tasks and each

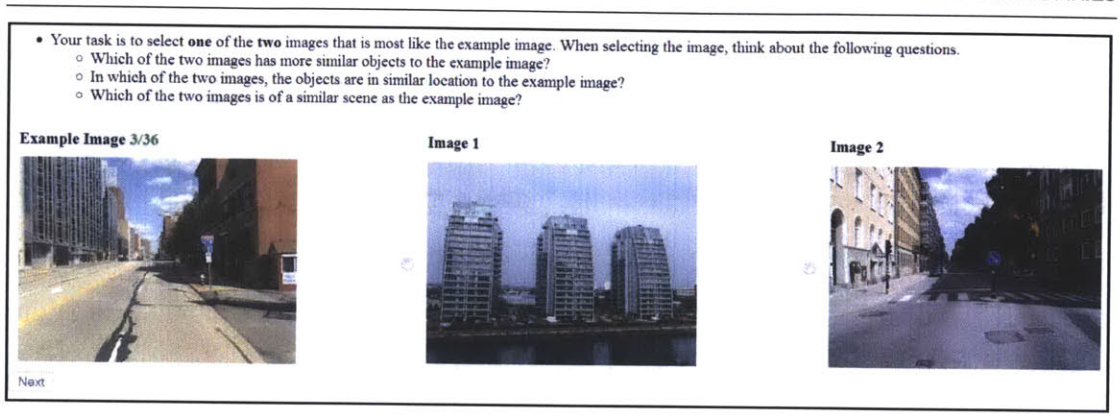



**Figure 6.6:** *Single example from the user study showing the ground truth G (Example Image), random R (Image 1) and prediction P/matching M (Image 2) images presented to the user. Comparison to the random guess allows for evaluation and ranking of the different image representations (feature sets) relative to each other. The random and predicted images are presented in random order during the task to ensure there is no selection bias.*

task contained 30 examples.

To evaluate the prediction performance, an example included (i) the ground truth image $G$ (the actual view after the camera transformation), (ii) the predicted image $P$ from the database obtained using the transformed image retrieval and (iii) a random image $R$ from the database (Fig. 6.6). The users were then asked to select which one of the (ii) predicted image or the (iii) random image was most like the (i) ground truth image, based on visual similarity of the scene type, common objects and their locations in the images. The predicted and random images were presented in a random order to eliminate possible bias if users had a preference in selecting the first or the second image more often. The random image in the triplet is kept the same across different evaluated features. The chance performance for this task is 50%. The best performance, i.e. if the users prefer the predicted image over the random image for all 300 test queries, would be 100%.

Similarly, to evaluate the matching task an example included: (i) the ground truth

image $G$, (ii) the best match from the database $M$ (obtained by matching directly on $G$ without a transformation), and (iii) a random image $R$ from the database.

To ensure the quality of the work, we set a qualification requirement that allowed only users with 95% task acceptance rate to participate. Furthermore, we added six control examples to each task presented in a random order. The control examples had a clear correct answer and allowed us to remove users that were not paying attention throughout the entire task. Tasks with less than 100% correct responses to the control examples were discarded, approximately 10-15% of the data. Each task was completed by 10 different users.

We performed a number of experiments using test examples generated from different descriptors and different database sizes. Over 100 different users participated in the user study. For each example, we had between 8 and 10 votes for the image that best predicted the ground truth image. The results for each experiment were evaluated by the majority vote. If more than 50% of the users selected the predicted image as the image that explained best the ground truth image, then we say the users preferred the predicted image. Note that the random images are kept the same across different users.

## 6.5 Experiments

We use the ground truth data to evaluate (i) the matching performance and (ii) the prediction performance for each camera motion. To evaluate prediction, we use the transformed query $Q$ to find the best match $P$ that should predict the ground truth image $G$ that the camera would see after performing a particular motion. To evaluate matching, we use the ground truth image $G$ to find the best match $B$ in the database. We consider the following two motions - rotate right by 22.5 degrees, and 2x zoom out. Performance is measured on the ground truth dataset of 300 queries by the user preference rate vs. the random image as outlined in section 6.4.

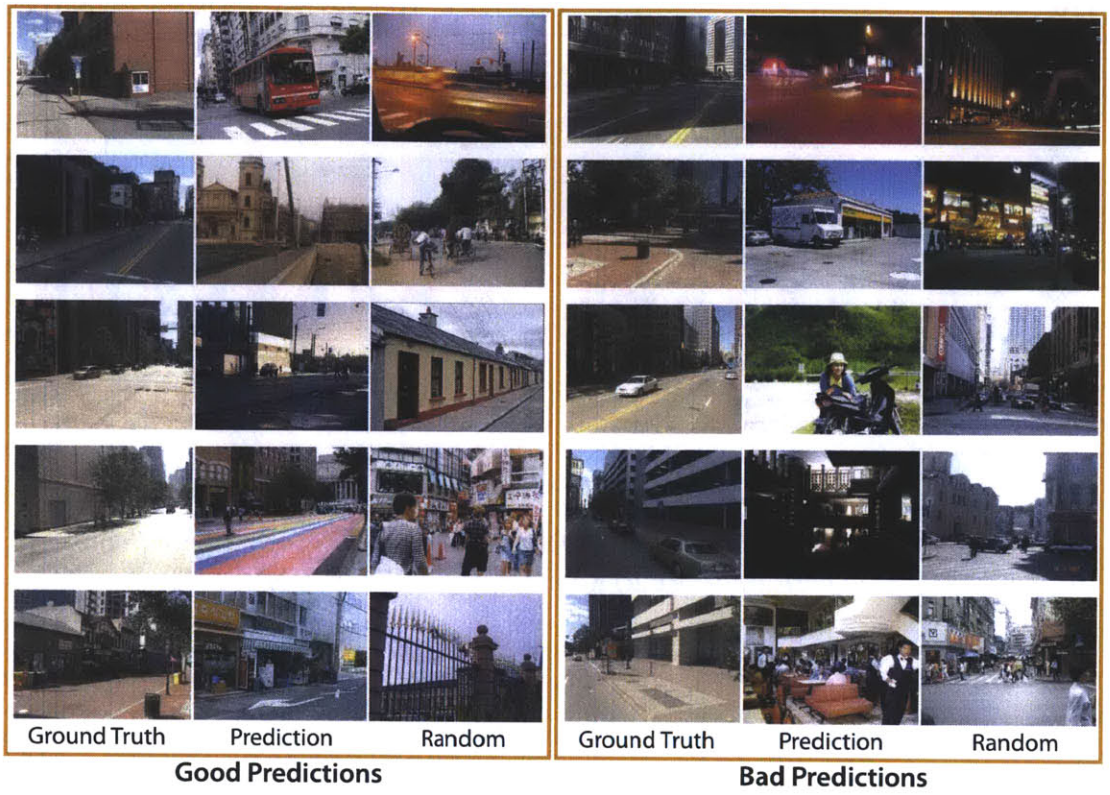
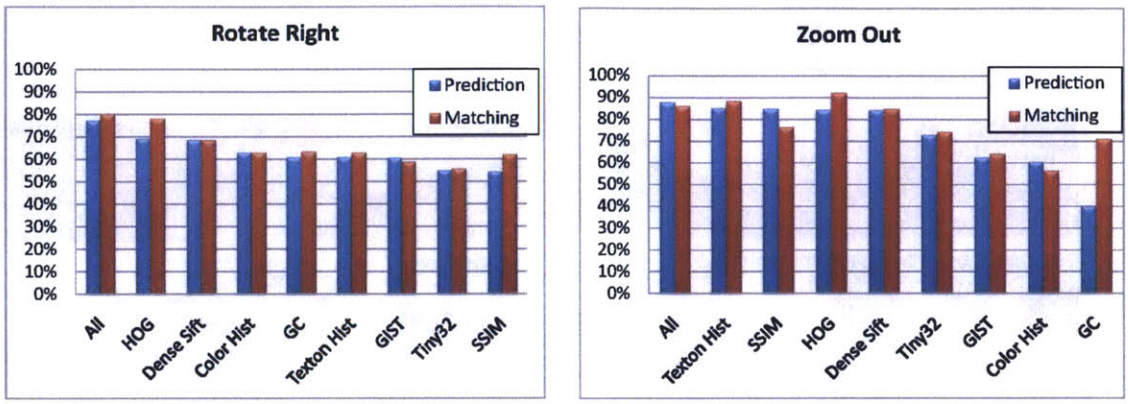



**Figure 6.7:** *Examples of image triplets shown to the user. The first three columns show examples where the users picked the prediction based on the HOG feature descriptor from a 100K database. The second three columns show examples where the users preferred the random image.*

of the HOG descriptor for both types of motions, however the HOG descriptor performs better in the matching task. For the rotate right motion, the geometric context (GC), self-similarity descriptor (SSIM), GIST and the color and texton histograms perform about the same – around 60%. The tiny image descriptor performs slightly above the chance performance (50%).

**Matching vs. prediction performance:** In the majority of the cases, the descriptors performed better at the matching task. This is to be expected since there is more information available in this case as we use the entire ground truth image to retrieve similar



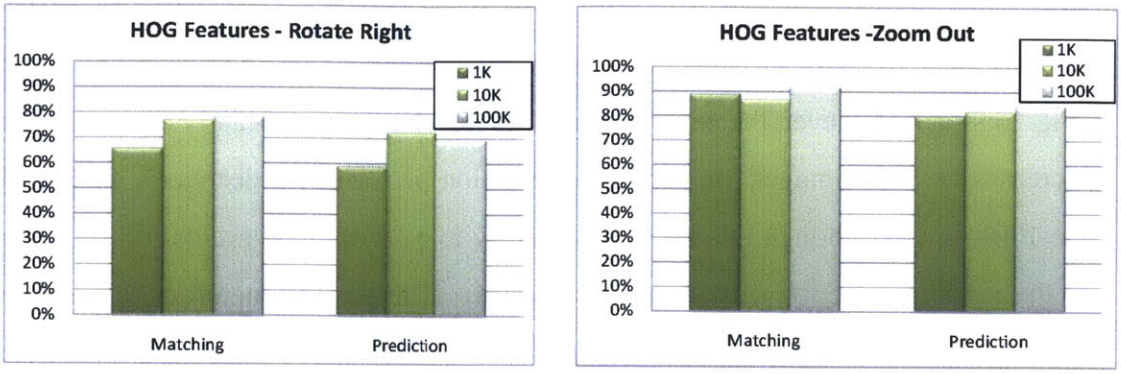

(a) (b)

**Figure 6.8:** *Performance of the different features in the matching and prediction tasks (sorted by matching performance). a) Rotate Right. b) Zoom Out.*

images from the database. For prediction, only the visible portion of the transformed query image is used (Fig. 6.1). Nevertheless, the significantly above chance (50%) results on the prediction task (77% for rotate-right and 88% for zoom-out) suggest it is possible, to some extent, to predict the appearance outside the boundary of the image for the street scenes considered in this work.

Interestingly, the performance of the self-similarity descriptor (SSIM) in the prediction task is significantly better than that for the matching task. One possible explanation are the matches on night images (see the fourth column of the top row of Fig. 6.10 (b) for the SSIM descriptor). The self-similarity descriptor is largely invariant to illumination. The users were generally influenced by color when selecting similar images even though they were not explicitly asked to compare the images based on color appearance.

For zoom-out motion, the geometric context (GC) is performing significantly worse for prediction than matching. This might be due to the fact that for the zoom-out prediction the transformed retrieval uses only the middle portion of the database images. This typically excludes the ground plane region – an important feature for geometric




(a) (b)

**Figure 6.9:** *Performance of HOG feature descriptor on different database sizes. a) Rotate Right. b) Zoom Out.*

context.

**Rotate vs. zoom-out motion:** Most of the descriptors perform better in the matching task for the zoom out motion than for the rotate right motion, despite the fact that the matching task is the same for both motions (i.e. no transformation is applied to the query). This is likely due to the nature of our ground truth test datasets where most of the images for the zoom out motion are perspective views of streets (see Fig. 6.10 (b)), which are well represented in the database of Flickr images. For the rotate motion the test set is more varied making the task considerably harder. Similar behavior is observed for the prediction task, further emphasizing the easier nature of the zoom-out motion test data.

**The effect of increasing database size:** We performed an additional experiment varying the size of the database of Flickr images between 1K, 10K, and 100K. For this experiment, we used the best performing feature descriptor on the 100K database (HOG) with the rotate right and zoom out motions. We selected a random subset of 1K and 10K images from our 100K database. The matching and prediction performance is shown in figure 6.9. Increasing the database size from 1K to 10K results in a marked

improvement for both the matching and prediction tasks for the rotate right motion (Fig. 6.9 (a)). However, the performance does not change significantly as the database increases to 100K images. In fact, the prediction performance decreases slightly for the 100K dataset, though this decrease is likely to be within the evaluation error (due to the random selection of comparison images). The performance for the zoom out motion for both tasks does not change significantly with the change in database size. Others have observed a steady increase in image matching performance with database size for recognition [103] or computer graphics [44] applications. In our case, the lack of observable improvement for the zoom out motion and when going from the 10K to the 100K dataset for the rotate right motion might be due to the fact that our Flickr database has been carefully pre-filtered (section 6.2.1) to contain mostly street scenes. The use of a scene classifier to select the images in our dataset may have reduced the variability in the data where adding images beyond 1K (for zoom out) or 10K (for rotate right) does not provide any additional information to make the tasks easier to solve. The reason for the lower bound for zoom out motion may be due to the easier nature of the ground truth data set where most of the scenes are of perspective street views.

improvement for both the matching and prediction tasks for the rotate right motion (Fig. 6.9 (a)). However, the performance does not change significantly as the database increases to 100K images. In fact, the prediction performance decreases slightly for the 100K dataset, though this decrease is likely to be within the evaluation error (due to the random selection of comparison images). The performance for the zoom out motion for both tasks does not change significantly with the change in database size. Others have observed a steady increase in image matching performance with database size for recognition [103] or computer graphics [44] applications. In our case, the lack of observable improvement for the zoom out motion and when going from the 10K to the 100K dataset for the rotate right motion might be due to the fact that our Flickr database has been carefully pre-filtered (section 6.2.1) to contain mostly street scenes. The use of a scene classifier to select the images in our dataset may have reduced the variability in the data where adding images beyond 1K (for zoom out) or 10K (for rotate right) does not provide any additional information to make the tasks easier to solve. The reason for the lower bound for zoom out motion may be due to the easier nature of the ground truth data set where most of the scenes are of perspective street views.

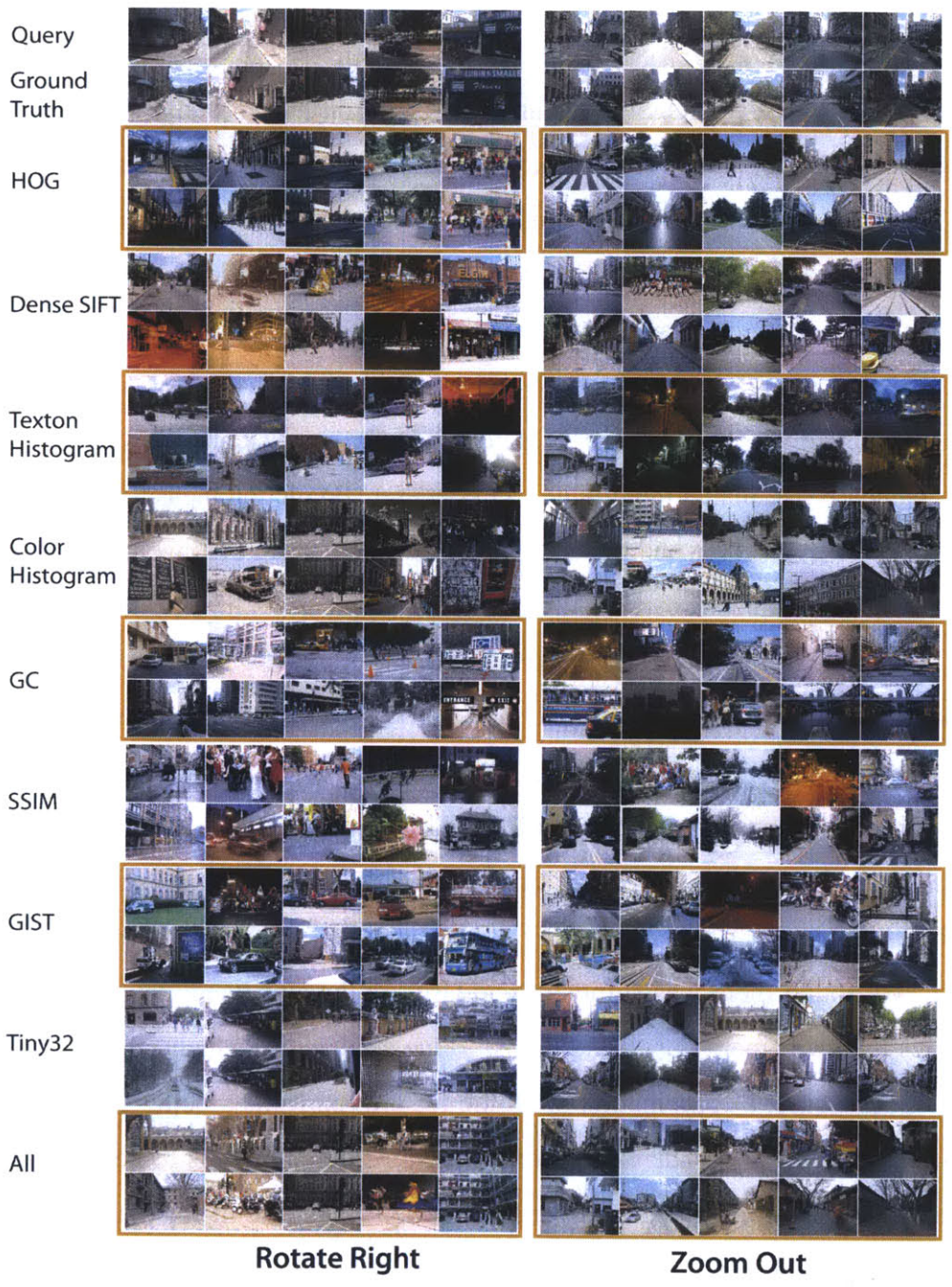



**Figure 6.10:** *Example matches and predictions on a 100K database. a) Rotate Right. b) Zoom Out. For each descriptor the images shown in: Top row - Matches to the ground truth; Bottom row - Predictions of the ground truth image based on the query image.*

## 6.6 Conclusion

We presented an extensive quantitative analysis based on a user study, showing that it is possible to predict visual content beyond image boundaries. Given a query image, we first transform it to simulate camera motion and then use it to retrieve images from the database as possible candidates for extending the field of view of the original query image. We conducted large scale experiments, using various image representation methods, on a large image database and ground truth data set. We investigated matching and prediction performance. Matching is the process of retrieving images from the database that match the ground truth data set. Prediction measures how well we can predict the world beyond image boundaries. To measure that, we use a ground truth geo-referenced data set that allows us to measure the accuracy of our prediction. We use Amazon Mechanical Turk workers to judge image matching and prediction performance relative to a random image choice as a way to assess the relative power of different algorithm choices.

We find that histograms of quantized HOG features worked best of the eight individual categories of features we tested. A dataset size of 100,000 images was sufficient to match images with 78 to 92 percent favoring the retrieved image over a random selection. We found there was sufficient regularity in the streetview datasets to predict images at new camera positions, almost to the accuracy to which images can be matched. In this case, a combination of all the features performed better than any individual feature set alone. The experiments described here allow us to calibrate and characterize datasets for the matching and prediction tasks, useful for dataset driven recognition and navigation.

# Chapter 7

# Infinite images: The image graph

## 7.1 Introduction

ONE application of the system described in 6 is creating a photorealistic virtual space to explore a large collection of images using intuitive 3D controls (fig. 7.1). We represent the space as an *image graph* where each node is an image and different types of edges correspond to different types of motions between nodes. Exploring a photo collection thus reduces to traversing the graph. The image graph can be used for other applications such as creating infinitely long panoramas by constantly panning the camera, or creating a photorealistic transition between two images by finding the shortest path connecting them in the graph. We describe how to construct the image graph, what are its properties and show a number of applications that are based on this representation.

The key challenge is to establish an edge between two nodes (images) if there is a smooth, photorealistic 3D motion between them. We do this using the transformation retrieval system described in 6.1.1. We approximate a number of 3D camera motions with simple 2D transformations and compute, for each image in the graph, its top matched candidates under various motions. The result is a sparse graph where every node is connected by a small number of edges (corresponding to different motion types) to other nodes in the graph.

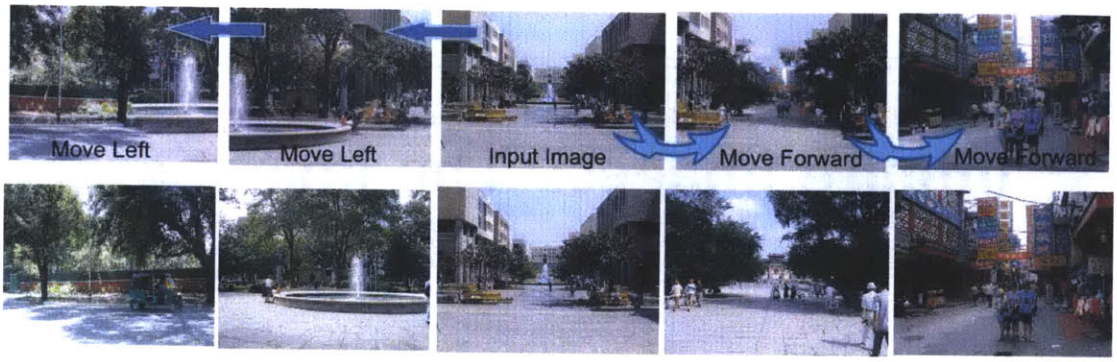



**Figure 7.1:** *Given the user supplied starting image (middle), our system lets the user navigate through a collection of images as if in a 3D world. Top row: Snapshots of the synthesized virtual scene. Bottom row: Original images of different real 3D locations automatically found in the image collection which were blended together to synthesize the virtual scene.*

The structure of the image graph impacts our ability to create photorealistic tours of the virtual space. If, for example, the image graph consists of a center node that all nodes are directly connected to, then all paths from one image to another will be extremely short and boring. If, on the other hand, the graph consists of many isolated connected components, then it will not be possible to reach many nodes in the graph. We analyze the graph and suggest ways to improve its structure to make paths more appealing.

Once we construct and analyze the image graph, we show that several different and useful applications can be reduced to finding paths in the graph. The basic application is web based and lets users explore the image collection interactively using intuitive 3D controls. In another application we show how to create an infinite panorama that corresponds to starting with the query image and following the camera panning edge from image to image for as long as needed. We can also create a video clip that is made of the images along the path. This can be used as a new transition effect between images. For example, we can create infinite zoom effects that resemble the "Droste

effect"[1] and has been used by various artistic groups to generate infinite zooms [2]. In the future, we believe that the image graph, and transformed image retrieval, can be used to create large photorealistic environment for applications like games or Second Life.

## 7.2 Related work

We represent the virtual photorealistic environment with the image graph and *not* with a 3D model. As in other examples of image-based rendering [41, 56], we generate the output images directly from input images, bypassing completely the 3D model as an intermediate step. This is because capturing images is easy, but constructing 3D models is time consuming and prone to errors.

We construct the image graph using image search methods. These methods can be roughly divided in two groups, those that use metadata cues such as timestamp, tags or geo-tagging, and those that rely purely on visual data. For example, time can be used to cluster images around events ("'Mike's birthday"', "'Summer vacation of 1997"') and, with the recent introduction of geo-tagging, images can be organized based on location ("'Trip to Italy"'). Google Street View and Microsoft Virtual Earth rely on geo-tagging to let users take a virtual trip around the streets of major cities around the world. Some image search engines organize images by tags that are either entered manually by users or collected automatically from the image caption or the web page containing the image. The intersection of metadata cues lets users query images on a semantic level ("'Find all images taken in New-York in 2004"'). Recently, there is great progress in the field of object [75] and, in particular, face detection [112]. Finding faces and automatically adding the corresponding names as tags to each image extends

[1] http://en.wikipedia.org/wiki/Droste_effect
[2] http://zoomquilt2.madmindworx.com/zoomquilt2.swf

the range of queries ("'Find pictures of Jonathan taken in Spain on November 2004"').

In a visual image search system, on the other hand, the user supplies a query image that is then matched to images in the database using low level image features such as color, shape and texture [35, 109]. Image features can be also extracted with a controlled degree of invariance to viewpoint and illumination thus enabling viewpoint invariant object and scene retrieval [93].

Our system is decidedly appearance based as we only rely on image content to retrieve similar images. But unlike other image retrieval systems we are not interested in retrieving similar images. Instead, we are looking for transformed image retrieval, where the retrieved image should match the query image after some transformation.

The retrieved images are often displayed as thumbnails, which is a practical but not very engaging way to browse images. If all images are collected from the same point of view then a big Gigapixel image can be constructed and viewed interactively using familiar 2D image controls [51].

Another option for browsing large collections of images is to treat images as points in a high dimensional space, compute the distances between them and use multi-dimensional scaling to display them in the image plane for the user to navigate through [84]. However, there is no effort to create a virtual 3D world and as a result there is no sense of "being there".

If the image dataset consists of a random collection of images of different scenes, taken at different times, one can construct an AutoCollage [82]. This gives visually pleasing collages, but fails to scale to large collections where thousands or millions of images are involved.

Alternatively, one can rely on 3D context to organize images. This was underscored by the success of the PhotoTourism system [97]. First, images are retrieved using tags ("'Find all images of Old City Prague"') and then calibrated to a common 3D space. Users can then browse the image collection by moving in 3D space. This was

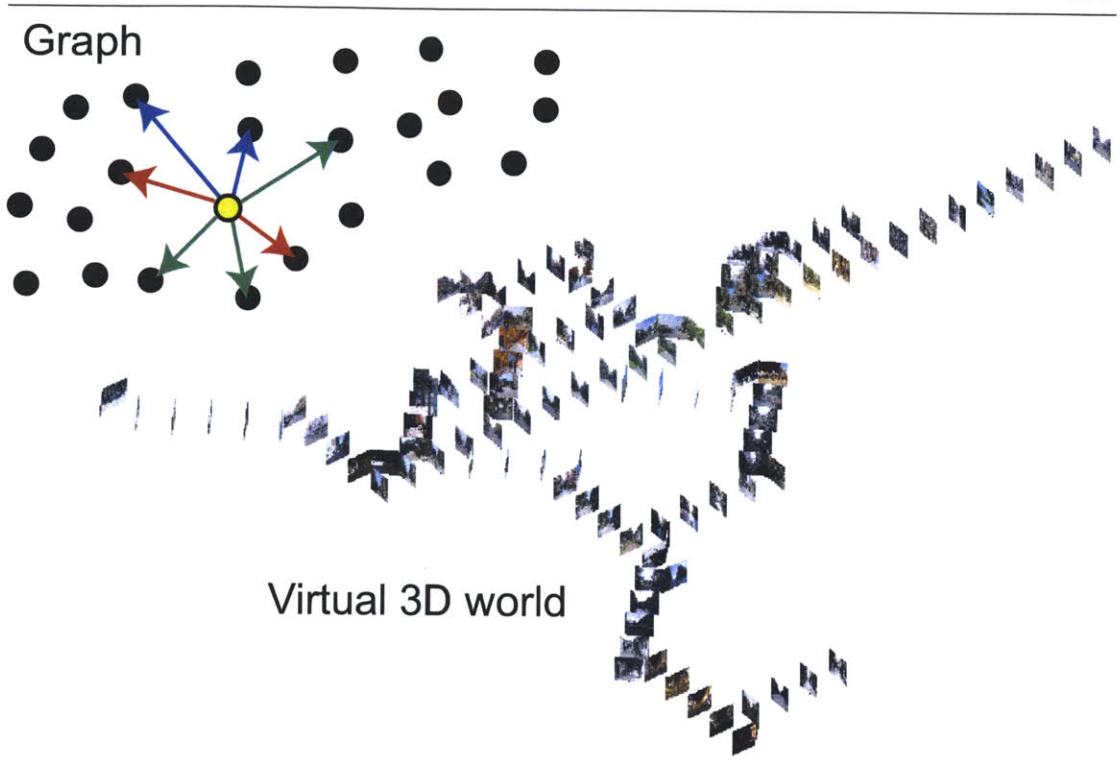



**Figure 7.2:** *Top: In the image graph, each image is a node and there are different types of edges (color coded in the illustration) that correspond to different camera motions. For each motion, there are several possible transitions to different images and we typically keep just the top 10 transitions for each edge/motion type. Bottom: Portion of an image graph laid out in a virtual 3D world. Relative locations of images are given by the camera motions. For example, for the forward motion the next image is displayed in front of the query image. Here only some edges of the graph are shown.*

later extended to detect paths in the image collection which gives a better control in navigating the 3D scene [95].

## ■ 7.3 The Image Graph

In this section we describe how we collect and arrange a large collection of images in an image graph. Each image in the collection corresponds to a node in the graph and an edge in the graph represents a transition between a pair of images. There are

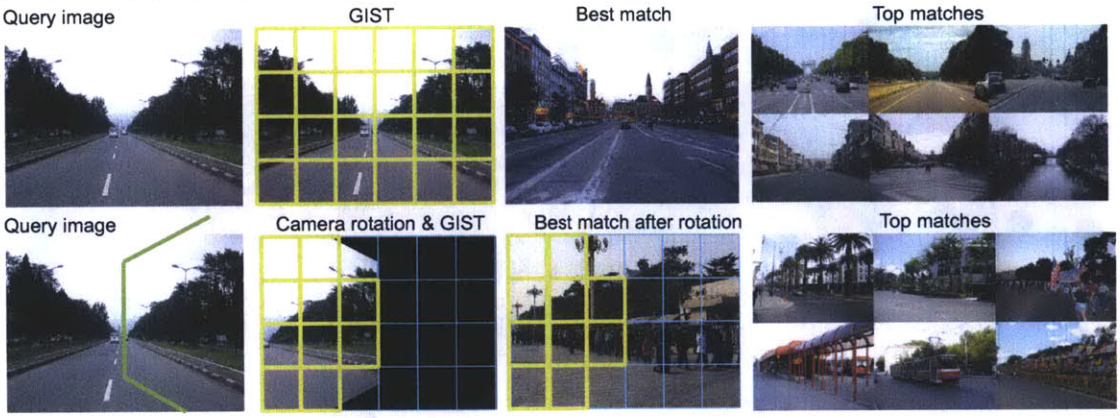



**Figure 7.3:** *Each row shows the input image, the $6 \times 4$ spatial bins used to compute the GIST descriptor, the best match on a dataset of 10,000 images, and the next 6 best matches. Top row: we look for images that match the full GIST descriptor. Bottom row: Result of a query after simulating a camera rotation. The returned images contain a new perspective, similar to the one that we will have obtained by rotating the camera 45 degrees to the right.*

different types of edges corresponding to different camera motions. The image graph is illustrated in figure 7.2. The images do not need to correspond to a real unique 3D space as in Phototourism [97]. Instead, our images are expected to correspond to unrelated, but visually similar, places.

To build the image graph we need to find for each image in the collection a set of candidate images suitable for a transition using a particular camera motion. These transitions will then form the edges in the graph. We use the *transformed image retrieval* method described in section 6.3.3 to find the set of candidate images. We hope that it contains images that can be seamlessly blended with the query image after applying the appropriate camera transformation.

## ■ 7.4 Image representation

Here, the images are represented using the GIST descriptor, which was found to perform well for scene classification [71]. The GIST descriptor measures the oriented

edge energy at multiple scales aggregated into coarse spatial bins. In this work we use three scales with (from coarse to fine) 4, 8 and 8 filter orientations aggregated into $6 \times 4$ spatial bins, resulting in a 480 dimensional image descriptor. While images of similar scenes, for example a street in New York and a street in London, can have very different colors and image intensities, we expect the coarse layout and orientation of image edges to be similar. The GIST descriptor can match similar scenes imaged under vastly different illuminations; for example, a street scene in a bright daylight can be matched reasonably well with a street scene in night. However, as we want to blend the matched images into seamless transitions, we would like to find matching scenes with similar illuminations. Hence, we also represent a rough spatial layout of colors by down-sampling each of the RGB channels to $8 \times 8$ pixels. We normalize both GIST and the color layout vectors to have unit norm to make both measures comparable. Similar image descriptors were used for scene matching in [45].

As illustrated in figure 6.4, not all pixels are observed in the transformed image and hence only descriptor values extracted from the observed descriptor cells in the query image are considered for matching. In this case, the GIST and the color layout vectors are renormalized to have unit norm over the visible region.

The distance between two images is evaluated as the sum of square differences between the GIST descriptors and the low-resolution color layouts of the two images. We set the weight between GIST and color to 0.5, which we found is a good compromise between matching on the geometric structure of the scene captured by GIST and the layout of colors. Currently we consider only images in landscape format with an aspect ratio close to 4:3. This represents about 75% of all collected images.

Figure 7.3 shows an example of a query image, the bins used to compute the image descriptor and the closest matching images from a dataset of 10,000 street images. The images returned belong to similar scene categories and have similar camera properties (same perspective pattern and similar depth).

## 7.5 Creating seamless transitions

We want to simulate transitions between pairs of images depicting different places, such as a street in New York and a street in London. This is a challenging task as image structures (such as windows on buildings) would not necessarily be aligned and can have different shapes and colors. Moreover, images returned by the scene matcher can still be misaligned as the GIST descriptors matches only the rough spatial structure given by the $6 \times 4$ grid of cells. For example, in the case of city skylines, the horizon line can be at a slightly different height.

To address these issues we first align the matched image with the query using a global image transformation (translation and scale change) to compensate for gross misalignments. Next we find a good seam between the aligned images. The goal is to transition between the two images at places where their image intensities are similar. Finally, to reduce the remaining misalignment artifacts and differences in color the two images are blended together in the gradient domain.

### 7.5.1 Horizon line estimation

Although we do not perform a full estimation of the 3D geometry of the scene structure, estimation of the horizon line can improve the quality of image transitions, as illustrated in Figure 7.4. In this example, we want forward motion to represent a person walking on the ground plane. As shown in the top row, if we zoom into the picture by using the image center as the focus of expansion, we move into the sky region. However, if we zoom in on the horizon line we simulate a person moving on the ground plane. It is important to keep the horizon line within the queried region.

We show how to estimate the horizon line from a single image by learning a regression function on the GIST descriptor [53, 104]. In contrast to [22, 26] in which camera orientation is estimated by an explicit model of the scene geometry, we use

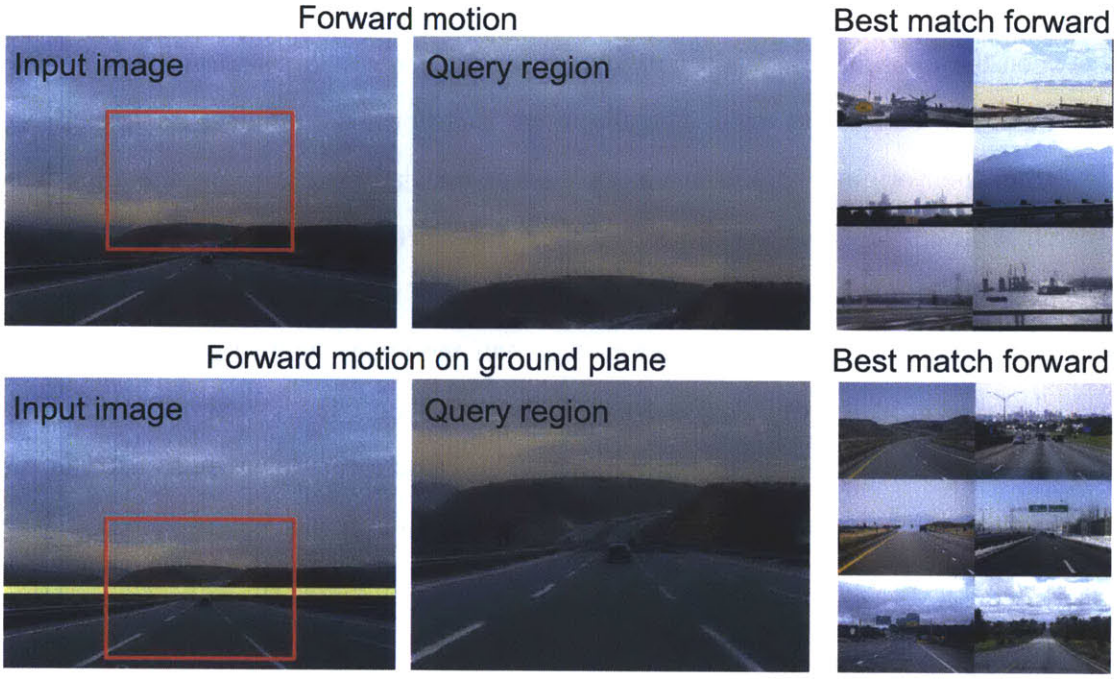



**Figure 7.4:** *The top row shows the queried region when we take the central image portion. The bottom row shows the results obtained when centering the queried region on the horizon line. The retrieved images contain roads with similar perspective to the input image.*

machine learning to train a regressor. Our method works even when the scene lacks clear 3D features such as a vanishing point. We collected 3,000 training images for which we entered the location of the horizon line manually (for pictures taken by a person standing on the ground, the horizon line can be approximated by the average vertical location of all the faces present in the image). We use a weighted mixture of linear regressors [39] to estimate the location of the horizon line from the GIST features as described in [104].

### 7.5.2 Alignment

To perform the alignment, we apply a standard gradient descent alignment [61, 98] minimizing the mean of the sum of squared pixel color differences in the overlap region

between the two images. We search over three parameters: translation offset in both the $x$ and $y$ direction and scale. The alignment is performed on images down-sampled to $1/6$ of their original resolution. In the case of translations and zoom, the alignment search is initialized by the synthesized view transformation. In the case of camera rotation we initialize the alignment with a translation in the x direction matching the image overlap induced by the rotation, e.g. half the image width for the example in middle column of figure 6.4. As a result, the camera rotation is approximated by a translation and scale in the image domain. The camera rotation is only used in the scene matching to induce a change in the geometry of the scene.

### 7.5.3 Compositing

The aligned images are blended along a seam in their region of overlap using Poisson blending [73]. In the case of camera translation and rotation we find a seam running from the top to the bottom of the overlap region minimizing the sum of absolute pixel color differences by solving a dynamic program [29]. In the case of zoom, where images are within each other, we find four seams, one for each side of the overlap region. In addition, to preserve a significant portion of the zoomed-in image, we constrain each seam to lie close to the boundary of the overlap region. Finally, images are blended in the gradient domain using the Poisson solver of [3]. Examples of composited images are shown in the bottom row of figure 6.4.

## 7.6 Building an image graph for a single motion

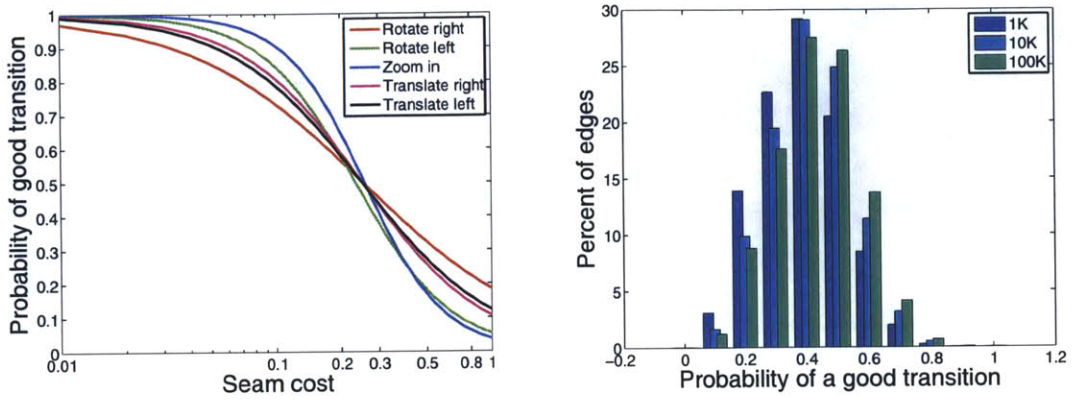
We process the image database and create a graph for each type of camera motion: (i) rotate left, (ii) rotate right, (iii) move left, (iv) move right and (v) move forward. We call graphs for one type of camera motion a motion graph. Motion graphs for different motions are combined into a single image graph, which we refer to as the combined


**Figure 7.5:** *Examples of good (top row) and poor (bottom row) quality transitions for different motion graphs. (a) Rotate right graph. (b) Move forward graph. (c) Move right graph.*

graph. For each transition between a pair of images (corresponding to an edge in the graph) we store the transformation aligning the images, the GIST matching cost of the two images, and the cost of the seam. In section 7.8 we show how to convert these costs into probabilities by manually labeling several examples of good and poor quality transitions. We typically retain only the top 10 matches measured by the GIST matching cost for each motion. This gives us a sparse graph that can be easily stored and allows for further efficient processing. Currently, we create a separate graph for each theme.

Note that edges in the image graph for each motion are directed, describing transitions from the query image to the (top 10) best matches in the database. (However, the direction of each edge could be reversed by reversing its image transformation. For example, a "rotate left" edge from image $A$ to image $B$, could be reversed to a "rotate right" edge from $B$ to $A$. Note however, that image matching is not symmetric, as the fact that $B$ is among the top 10 nearest neighbors of $A$ does not imply that $A$ is among the top 10 nearest neighbors of $B$.)

In terms of computation cost, computing GIST descriptors takes about 0.2 seconds




(a) (b)

**Figure 7.6:** *(a) Fitted models mapping seam cost to probability of a good transition based on 300 labeled examples for each motion graph. (b) Histograms of edge probabilities in 1K, 10K, 100K graphs.*

per image, querying a database of 100K images takes about a second in our Matlab implementation, and it takes a couple of seconds to align and blend the retrieved images with the query image. These timings are for a 2GHz machine.

In our current implementation we do not enforce the image graph to respect constraints of a real physical space. Each transition depends only on the particular query image and there are no additional dependencies on other edges in the graph. As a result, it is perfectly legal in our image graph to keep rotating without coming back to the query image. Similarly, it is possible to start with a query image, move left, then move right and not get back to the original query image.

## ■ 7.7 Combining image graphs for different motions

Each motion graph provides the user with the option to explore the image space using one particular camera motion, but to create interesting tours we would like to simulate different camera motions. As described in section 7.6, the edges in the graph can be assigned weights based on the GIST matching cost or the seam cost. Since these

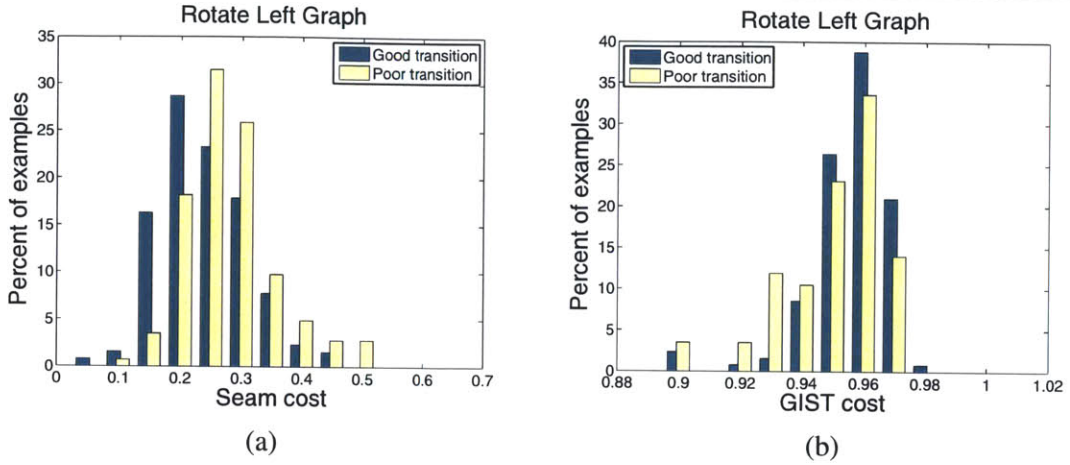



**Figure 7.7:** *Edge cost distribution for the 300 manually labelled good and poor quality transitions in the "rotate left" graph. (a) Seam cost. (b) GIST matching cost. Lower cost should correspond to better quality transitions.*

are computed differently for each motion graph, they are not necessarily comparable. Instead of using the previously described edge costs for merging the graphs, we fit a model using logistic regression to convert the edge cost to a probability of a good transition for each motion based on 300 hand-labeled training examples. As training data we randomly picked 300 pairs of nodes from each motion graph. After stitching and blending the images using the appropriate motion, we labeled the results as good or poor quality. Figure 7.5 shows examples of good and poor quality transitions for rotate right, move right, and move forward camera motions. The labeling is subjective and was based on the labels of a single user. In our training data, we have about 50% positive and 50% negative examples for all motions except move forward where there are only 35% positive examples. The distribution of the GIST matching costs of the positive and negative examples show more overlap than that of the seam costs (Fig. 7.7) making it harder to learn a good model. For the rest of our experiments, we use models mapping the seam cost (rather than the GIST cost) to estimate the probability of a good transition as shown in Fig. 7.6 (a). The difference in models for different motions shows

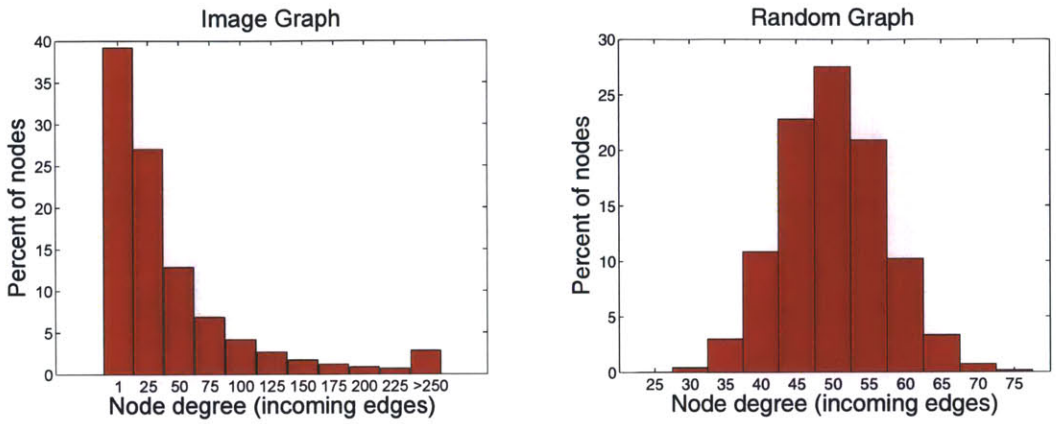



**Figure 7.8:** *Histograms of incoming edges. (a) 100K image graph. (b) 100K random graph.*

that using the seam cost directly to determine transition probabilities in a combined graph would give unequal weights for different camera motions.

We use the learned models for mapping the seam cost to the probability of a good quality transition to assign new weights to the edges of the different motion graphs. With comparable edge weights, we then can merge the motion graphs into a single graph. The directed edges in the new image graph correspond to the different types of motion and the weights refer to the probability that the given motion produces a good quality transition. If there is an edge between a given pair of images in multiple motion graphs, we retain only the edge for the motion that results in the best quality transition. We consider the distribution of the edge weights in graphs with 1K, 10K, and 100K nodes. The images of the 1K and 10K graphs were a random subset of the images of the 100K graph. Intuitively, we expect that adding more images to the graph will increase the probability of finding good quality transitions between images. The distribution of the edge weights, indeed, shifts with the increase in graph size (Fig. 7.6(b)). The 100K graph contains a larger proportion of edges with higher probability of producing a good transition than do the smaller image graphs.

## 7.8 Properties of the image graph

The motion graphs allow us to create photorealistic tours of the virtual image space using a single 3D motion. To add versatility to our system and make the tours more interesting, we merge the separate motion graphs into a single graph where the directed edges correspond to a particular type of motion and their weights represent the probability that the edge produces a good quality transition (Fig. 7.2). The properties of the resulting image graph impact the quality of the tours through the image space. For example, if the graph has nodes with a large number of incoming edges (hubs), it is likely that most tours would pass through the same subset of images and the paths between any pair of nodes would contain only a small number of transitions. In that case, exploring the image space can quickly become boring. We discuss the presence of such high degree nodes in the graph and contrast the structure of the image graph to that of a random graph. Edges with low transition probabilities can also affect the quality of the tours because they can result in paths connecting semantically incoherent images. We propose several ways to augment the image graph in order to produce better tours and discuss how they affect the connectivity. If the graph has multiple isolated connected components, some portions of the graph would never be reached. A path leading to a small connected component could force the user to visit the same nodes over and over again when trying to generate longer tours. In discussing graph properties in this section, we analyze a graph constructed based on the "streets" theme that combines all five types of motion - rotate right, rotate left, move right, move left, and move forward.

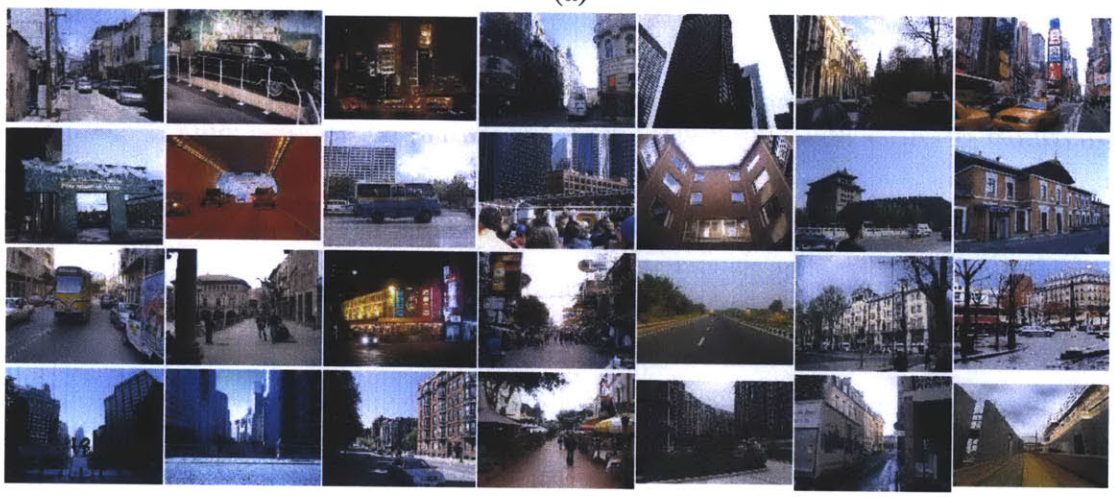
### 7.8.1 Popular images

An interesting aspect of the graph is the node degree for the incoming edges. The number of outgoing edges in the combined graph is almost constant because for each motion graph we only keep a subset of the edges corresponding to the top 10 matches

measured by the GIST matching cost. The number of incoming edges, however, varies significantly (Fig. 7.8 (a)). The majority of the nodes have 50 or fewer incoming edges with 5-8% of nodes having one or none. There are also a number of very well connected nodes with thousands of incoming edges in a graph with 100K nodes. A sample of the popular images in different motion graphs is shown in figure 7.9 (a). Most of the images are very textured, especially those from the forward motion graph, or have a lot of structure. They also have very similar color layout. It is likely that if there is no good match in the database for a given query image, due to their regularity and uniform lighting these images can give good matching costs even though they may not be semantically correct. Even if the transitions to a popular image are semantically correct, too many paths going through one image would make the image space exploration uninteresting. The popular images from the forward motion graph are clearly incorrect semantically. Interestingly, in the combined graph, none of them are amongst the top most high-degree nodes. Most of the popular nodes seem to come from the rotate and translate motion graphs. This can be explained by the fact that we have two graphs for both rotate and translate motions and all four of them favor images with similar structure thereby preserving the connectivity of their popular nodes. A sample of images with no incoming edges are juxtaposed with the popular ones in figure 7.9. The images with no connections show wide color and lighting variation in comparison to the dull colors and uniform lighting of their popular counterparts.

To further understand the distribution of edges in the image graph, we constructed a random graph. First we created a graph for each motion by adding 10 randomly chosen outgoing edges to each node and assigning the edge weights to values between 0 and 1 by randomly drawing from the uniform distribution. The graphs were then merged together to create the random combined graph. The distribution of the incoming edges in the random graph is quite different from that in the image graph. Here we have no nodes with fewer than 20 and with more than 80 incoming edges (Fig. 7.8 (b)). The

(a)



(b)

**Figure 7.9:** *Sample images with (a) many or (b) no incoming edges in different motion graphs. Top row: Forward motion. Second row: Rotate right motion. Third row: Move left motion. Bottom row: Combined motions.*

properties of similar types of random graphs have been studied in e.g. [30].

### ■ 7.8.2 Graph connectivity

To ensure versatility of the tours of the virtual space, we would like to be able to reach a node from any other node in the image graph. In graph theory terms, the image graph

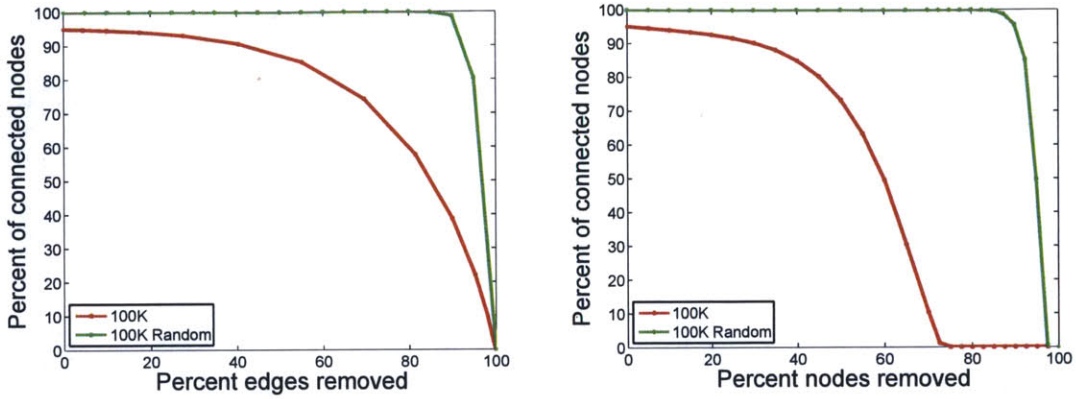



**Figure 7.10:** *Percentage of nodes in the largest connected component of the 100K image and random graphs. (a) After removing edges with low probability of a good transition. (b) After removing nodes with the highest number of incoming edges.*

should be strongly connected [21]. We already saw that there is a small percentage of nodes in the image graph that have no incoming edges. Unless we start the exploration of the image space from one of those nodes, we will never be able to visit them. It is possible that the image graph is not well connected, i.e. it could have several isolated strongly connected subgraphs. Our tours will then be limited to one portion of the graph which would be undesirable. We consider the size of the largest connected component in the graph. Ideally, most nodes, with the exception of those with no incoming edges, will be part of it. Figure 7.10 shows that over 90% of the nodes lie in the largest strongly connected subgraph.

We wish to improve the quality of the transitions but at the same time retain the ability to travel through a large portion of the image graph. One option is to remove edges with low probability of producing a good quality transition. It appears that we can safely discard about 30% of those edges without reducing the size of the largest connected component (Fig. 7.10 (a)). In contrast, we can remove almost 90% of the poor quality edges in the random graph, before the graph becomes disconnected at all

(Fig. 7.10 (a)). This behavior can be attributed to the uniform distribution of edges and edge weights in the random graph. In contrast, edge occurrences (and their weights) in the image graph are likely to be correlated and dependent on the type and content of the query image.

Another option for improving the tours through the image space is to reduce the number of times we pass through the same node. To achieve this, we can remove nodes with many incoming edges. At the same time, we would like to preserve the proportion of nodes that belong to the largest connected component in the resulting subgraph. Figure 7.10 (b) shows the percentage of nodes left in the largest connected component after removing a given number of high-degree nodes from the graph. We can safely discard about 20% of the most popular nodes without affecting the connectedness of the image graph. This suggests that the high-degree nodes lie in the periphery of the graph and are not central to its connectivity. As we observed earlier, the random graph is much better connected than the image graph. Due to the uniform distribution of edges in the random graph, even removing 80% of the high-degree nodes still leaves the remaining graph well connected.

### ■ 7.8.3 Path properties

Our system can be used for finding tours between a given pair of images ("image taxi" described in section 7.9.2) or providing transitions between images of a personal collection. In both scenarios, we would like to find a path with edges that have high probability of providing a good transition. We use the Dijkstra's shortest path algorithm [21] to find the best path between a pair of nodes selecting the best quality edge at each step. In the image taxi scenario in particular, we prefer longer but still good quality paths. We already know that our graph is well connected as discussed in section 7.8.2. The question we ask now is: How far away each node is from all other

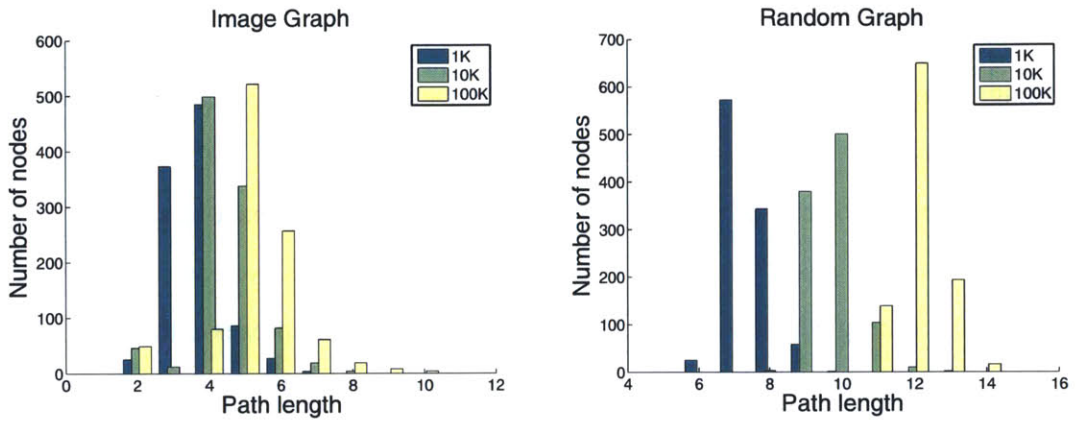
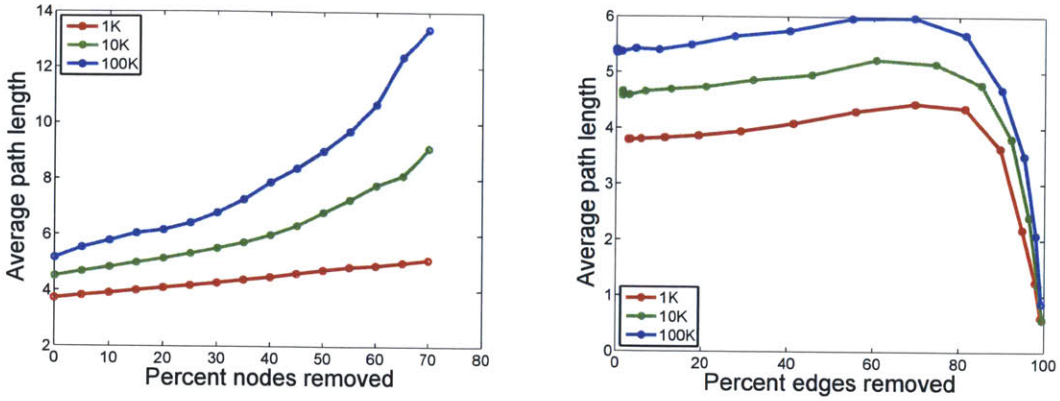



**Figure 7.11:** *Average path length to a given node from all other nodes in the graph (sample of 1000 nodes). (a) Image graph. (b) Random graph.*

nodes?

We computed the average length of the best path to 1000 randomly sampled nodes in image graphs with 1K, 10K, and 100K nodes (Fig.7.11 (a)). It takes on average 4-6 transitions to reach a node from any given node. This suggests that no two nodes lie very far from each other, which can be explained by the portion of very high-degree nodes in the image graph. In the random graph, on the other hand, the average path has about twice as many transitions as those in the image graph. Despite the fact that the random graph is better connected, its nodes lie farther apart from each other because there are no hubs in it. The number of transitions also increases as we add more nodes to the graph since the number of outgoing edges for each node does not vary. This is more pronounced in the random graph but it can be also seen in the image graph.

The effect that high-degree nodes have on the image graph can be further observed in figure 7.12 (a). Removing the hubs leads to an increase in the average number of transitions required to reach one node from another. The trend is more visible in the larger graphs due to the fact that the number of outgoing edges does not scale with the graph size. The figure does not capture the possibility that the graph may become




(a) (b)

**Figure 7.12:** *Average path length to a given node from all other nodes in the graph. (a) After removing high-degree nodes. (b) After pruning the low quality edges.*

fragmented with lots of isolated connected components as we ignore nodes that are not connected when estimating the average path length. Thus, one has to be careful with removing high-degree nodes in an attempt to create longer paths because, while the number of necessary transitions may grow, the largest strongly connected component will become smaller. The image graph starts to become disconnected after 30% of the nodes are removed and becomes completely disconnected when the number of discarded nodes goes to 70%. A small increase in the average path length could also be achieved by pruning edges that have low probability of producing good quality transitions (Fig. 7.12 (b)).

## ■ 7.9 Application of the image graph

Once the image graph is constructed, we can use it to create different applications using the same basic machinery - finding paths in the graph. We present an interactive web interface that allows the user to navigate the image space using intuitive 3D controls. We also use the graph to create "infinite" panoramas starting from a given query image

**Figure 7.13:** *Various panoramas created by our system. The top two panoramas were created automatically from the "landscape" and "skyline" themes respectively. The bottom three panoramas were created interactively from the "people", "forest" and "graffiti" themes respectively.*

and traversing the graph using a particular camera motion. The image taxi tool allows the user to generate a tour between two images by finding a good quality path between the corresponding nodes in the graph. Finally, the graph could be augmented to include a personal photo collection that can later be viewed by using images from the graph as transition "filler" images.

We use three different camera motions in the proposed applications - translation, rotation, and forward motion. The camera translation is simulated by translating with constant per pixel speed between the pair of input images (also applying small scale changes if the consecutive images are of different size). The case of camera rotation is similar to translation but, in addition, we display the resulting images on a cylinder. In the case of forward motion we apply constant scale change between the two images to create an impression of a person walking at a constant speed in the 3D world. The focus

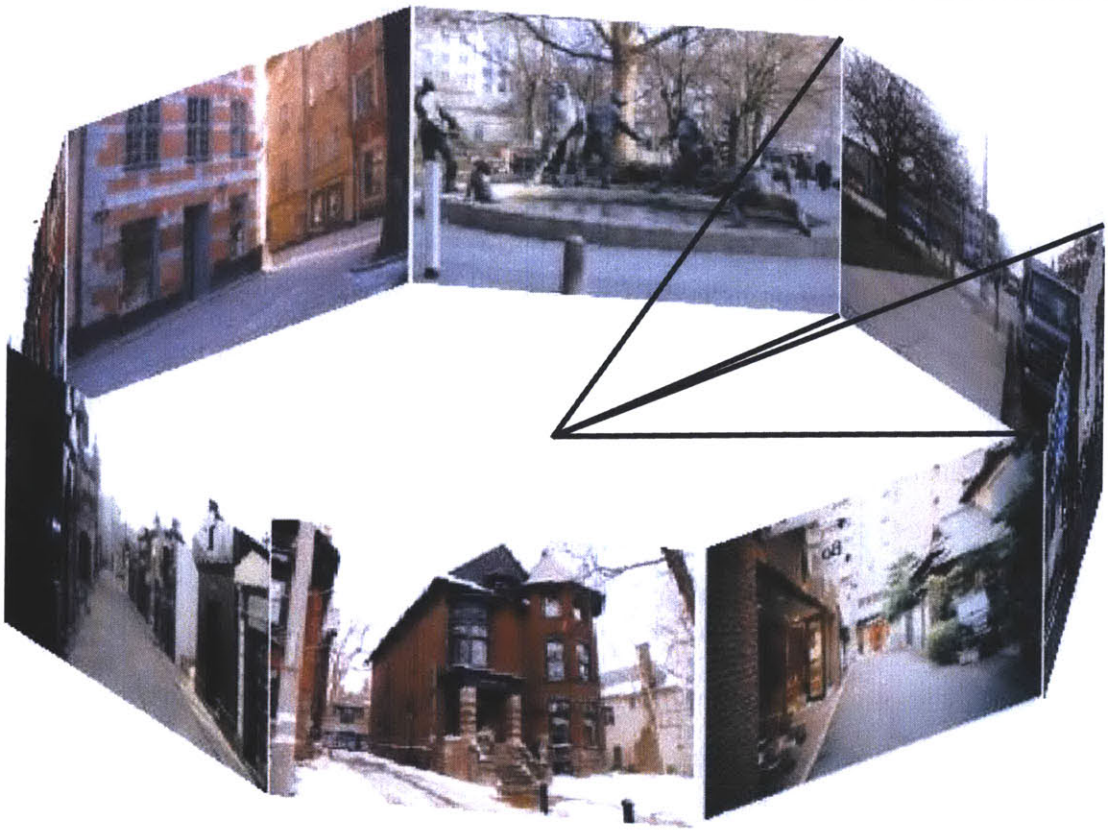


**Figure 7.14:** *A camera rotation can be used to generate from a single picture a good guess of the surrounding environment not covered by the camera. Here, the system is hallucinating what could be behind the camera (original image marked with a frustum). Note that the back of the camera is also a perspective street, aligned with the camera view.*

of expansion for the forward motion is at the center of the next aligned image, which can result in (typically small) changes in forward motion direction between consecutive images.

### 7.9.1 Infinite panoramas

One application of the image graph is to create an infinite panorama by starting with a query image and following the "camera translate" edge from image node to image

node. Each retrieved image is then blended with the query image to create an ever larger panorama. Figure 7.13 shows examples of long panoramas created for various semantic themes.

We can create panoramas for each of the three camera motions: translation, rotation and forward motion as is demonstrated in figures 7.15, 7.16 and 7.17, respectively. These panoramas were created from the "streets" theme by starting with a query image and following the same camera motion edge from node to node. Note that the left/right translation motion tends to preserve the camera orientation with respect to the scene (the scene remains roughly fronto-parallel), while the rotation motion induces a change in perspective between consecutive images (Fig 7.14). The translation and rotation sequences were produced fully automatically. The forward motion sequence was produced interactively by letting the user specify the direction of the motion - toward the horizon of the image. Because it is difficult to display an infinite zoom on paper, please refer to this video (`http://people.csail.mit.edu/biliana/papers/pieee2009/pieee_video.avi`). As mentioned earlier, there is no restriction in our representation for a rotation panorama to go beyond 360 degrees *without* returning to the first query image.

### 7.9.2 The image taxi: finding a path between two images

The image taxi tool lets the user specify the first and last images of a tour and then computes the shortest path in the image graph that connects the two end images. This option can be used to generate smooth transitions between images from a private photo collection, using a large, generic image database, or to generate a smooth transition between two video shots. Given two input images, we first connect them to the graph and then find the shortest path between them using the Dijkstra algorithm [21]. We then follow the path creating a tour based on the different edges along the way. We

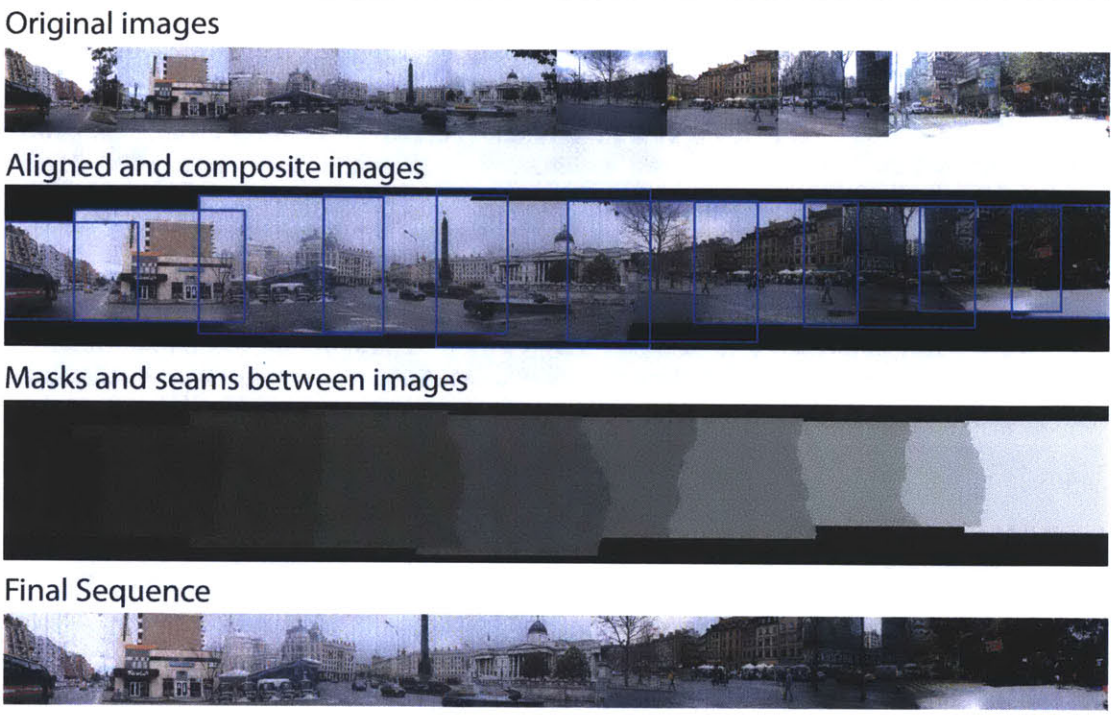



**Figure 7.15:** *Sequence generated from a large database of street images by inducing camera translation. Each consecutive image was obtained by inducing camera translation by half the image size as illustrated in figure 6.4.*

demonstrate the image taxi in the accompanying video.

The video also contains the different tours generated by our system. All the sequences presented there were generated automatically, except for the "Hollywood" and "Windows" sequences where the user had to choose the best match out of the top 5 candidates.

### 7.9.3 The interactive system

We have designed an interactive viewer that allows exploring the image graph using intuitive 3D commands (move left/right, zoom, rotate left/right). The viewer is implemented in Flex and runs using Flash Player 10 in a standard web browser. The user can select a particular image graph (theme) to explore. After choosing an image that serves


**Figure 7.16:** *Sequence generated from a large database of street images by inducing camera rotation. Each consecutive image was obtained by inducing camera rotation of 45 degrees as illustrated in figure 6.4. This produces changes in the perspective between consecutive images.*

as a gateway to the virtual 3D space (Fig. 7.18 (a)), s/he can start navigating interactively. Five regions of the screen correspond to the different motions the user can take. Moving the mouse displays the top matching image for each motion overlaid on the current image (Fig. 7.18 (b)). The system automatically moves to the image selected by the user (Fig. 7.18 (f,g,h)). If the user is not satisfied with the top matching image he can choose a different image from the top ten matching images (Fig. 7.18 (e)). The system also allows to undo a transition providing the user with opportunity to explore alternate routes (Fig. 7.18 (d)).

The transition between every pair of images must be smooth to give the impression of "being there". Thus it is important to be able to synthesize each intermediate

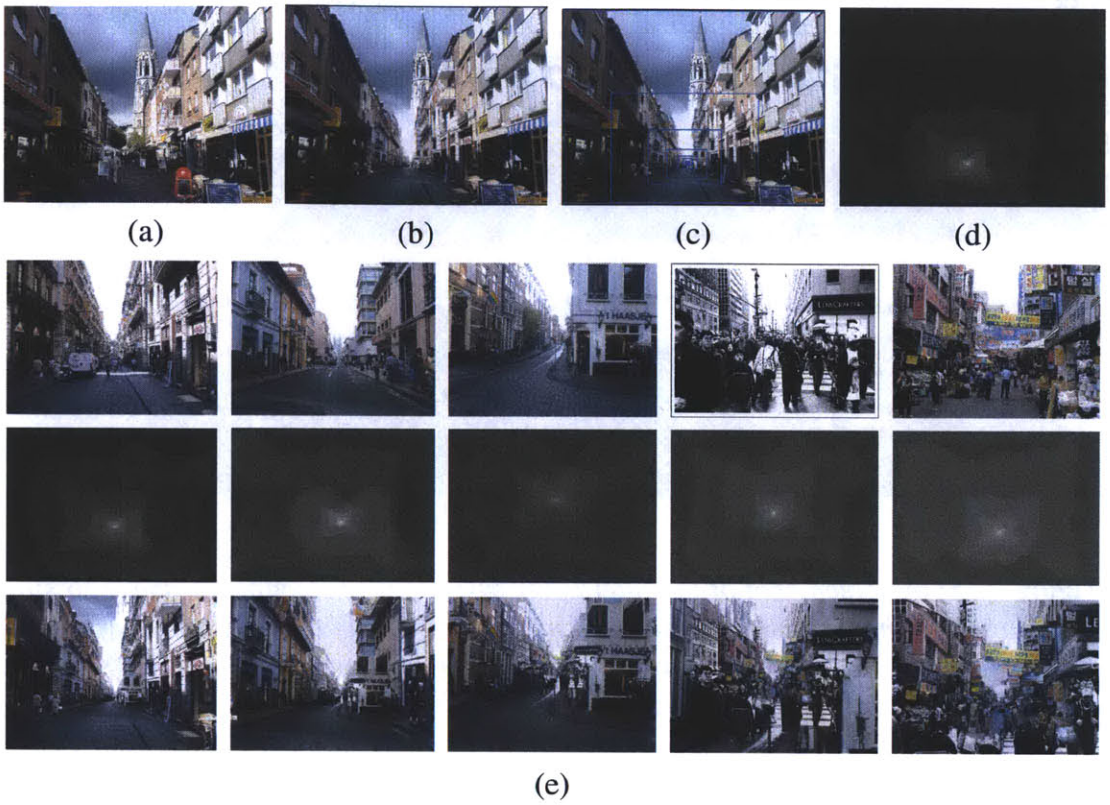



**Figure 7.17:** *Forward motion sequence of 16 images. (a) The query image. (b) The query image composited with the consecutive images in the sequence. (c) Image boundaries of the following images in the sequence overlaid over the composited image (b). (d) Masks indicating areas in (b) coming from different images. (e) Images 2–6 of the forward motion sequence. Top row: Original images. Middle row: Masks. Bottom row: Composited images. Note that the entire sequence contains perspective views of streets. Each consecutive image was obtained by inducing camera forward motion as illustrated in figure 6.4.*

image frame efficiently. The images in the graph have already been aligned offline (section 7.5.2), therefore, to stitch two images we only need to compute on demand the best seam in their region of overlap and blend them. To ensure real-time transitions, we use the fast pyramid blending instead of the more expensive Poisson blending to construct the blended composite [17] and we apply the blending only on the region of overlap between the two images. The intermediate frames of the motion are syn-


**Figure 7.18:** *A screen shot of the interactive system. (a) Selecting an image to start browsing. (b) The current image is shown in the center and the next image for each motion (forward, rotate left/right, translate left/right) is shown as an inset. (c) Selecting rotate right motion. (d) The user can undo the current transition. (e) The next image for a particular motion (here move right) can be also interactively chosen from the top 10 matching images displayed on the side. (f-h) show the first, middle, and last frame of a rotate right motion sequence.*

thesized from the composite image. To achieve smooth transitions, we use kernels written in Adobe's Pixel Bender language for hardware-independent description of image processing algorithms [1] to perform the pyramid blending and the synthesis of the intermediate frames. The Pixel Bender kernels can be compiled efficiently for the CPU and/or the GPU and allow significant speed-up of computationally intensive image processing tasks. Currently, Flash Player supports only limited set of features and does not support kernels running on the GPU but if the extra features are added, it will lead to even faster transitions. The computations are memory intensive and are performed on the client machine, therefore they require about 500MB-1GB available memory to run smoothly.

After a transition is complete, the image displayed to the user is a blend of two images from the database - the original query and the retrieved image. To produce the

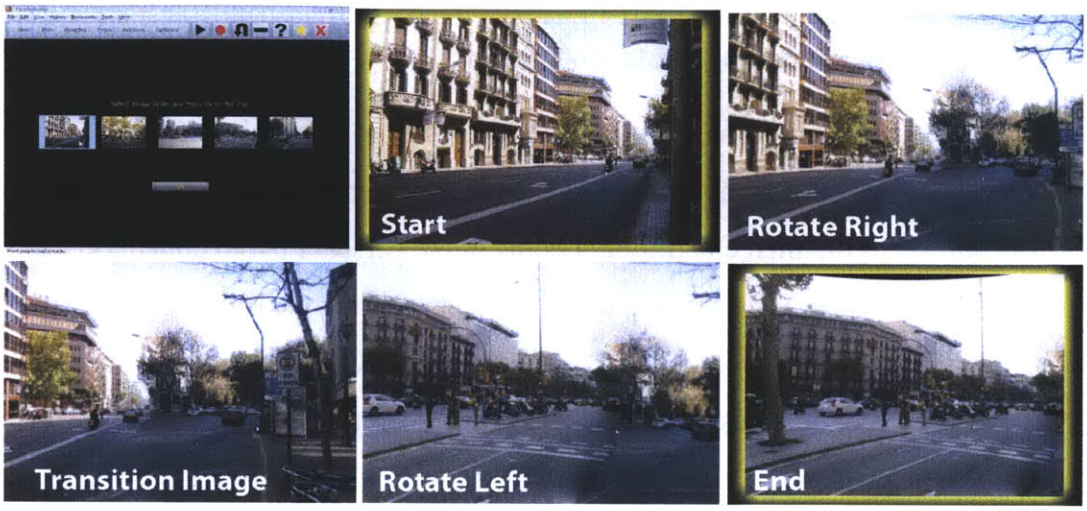



**Figure 7.19:** *Viewing a personal photo collection in the interactive tool. Column 1 shows the interface for placing the images from the personal collection in the desired viewing order. Columns 2-5 show a path between two of the images in the personal collection containing one transition "filler" image. Note the second transition (rotate left) does not bring us back to the starting image since we do not enforce spatial consistency in the graph.*

next transition, we use the actual retrieved image rather then the displayed blend as the next query. This allows us to take advantage of the pre-computed image graph. Despite the fact that the query image is not identical to the one displayed to the user, we found that it serves as a good enough proxy.

The system also has a non-interactive mode where the next image is chosen automatically by selecting the edge and camera motion type with the highest probability of a good transition. If the user likes a particular sequence, it can be saved as a "trip". The trips can be chained together to create a much larger tour that can also be played automatically. This way, the user can build a library of trips and then mix and match them to achieve a particular tour.

### 7.9.4 Touring personal photo collections

We can also augment the image graph with images from a personal photo collection. The user could then view their photo collection in the interactive web interface in a manner similar to the image taxi tool. Photos in the personal photo collection can be arranged in any order. Then a new tour is created by finding the shortest path between each pair of images in the collection. The tour can be played automatically in the interactive web interface. Images from the personal photo collection are highlighted while those from the original image graph serve as transition "filler" images. Augmenting the image graph needs to be done offline before the photo collection can be viewed. Figure 7.19 provides an example of a five image personal photo collection from street scenes of Barcelona. The image in the first column shows the interface for placing the images in the personal collection in the desired viewing order. The other columns in figure 7.19 show a path between two of the images in the personal collection that only contains one transition "filler" image. Notice that the motions used to travel from the start to the end image are rotate right and rotate left. As mentioned in section 7.6, the image graph does not preserve spatial consistency. The rotate left motion does not bring us back to the start image but instead we arrive at a new location.

## 7.10 Limitations

In the course of developing the system we have learned about some of its limitations. First, the system is as good as the underlying database is. The larger the database, the better the results. We also found that there are three typical failure modes. The first is when a semantically wrong image is retrieved. This is mitigated by the use of themes but still sometimes occurs. Second, compositing two distinct images is always a challenge and at times, the seam is quite visible. Finally, there are cases in which the seam runs through important objects in the image which produces noticeable artifacts.

## 7.11 Conclusion

We proposed creating a large photorealistic virtual space using the transformed image retrieval method as an intuitive 3D-based navigation approach to browsing large photo collections. Our system arranges the images into semantic themes and performs transformed image retrieval to simulate various camera motions within the large image database. Matches are pre-computed offline and the image collection is organized into an image graph, where images correspond to nodes and edges in the graph correspond to transitions between images using different camera motions. We examined properties of the image graph and their effect on tours of the image database. We used the image graph to create synthetic panoramas, interactively tour the virtual 3D space, or create an image taxi tour from one image to another through the dataset. These tools enable intuitive interaction with large image datasets and may enhance interactive games, movies, photo-sharing web sites and personal photo collections.

# Chapter 8

# Conclusion

IN this thesis, we used large databases of real and synthetic images to evaluate local feature descriptors, learn new feature detectors, predict what the camera would have seen under a given motion and synthesize images based on those predictions.

One of the difficulties in comparing and evaluating feature descriptors is the ability to acquire ground truth labeled data, specifically pixel to pixel correspondences between the images, for complex 3D scenes under different scene and image transformations in a controlled way. In chapter 3, we proposed a dataset based on highly photorealistic images of a virtual city model. Previous datasets used in the evaluation of image descriptors and detectors were limited as they did not include scenes with changes in perspective, presence of occlusions, cast shadows and specularities, and controlled changes in illumination. Using a computer generated model provides us with complete knowledge of the geometry of the scene and allows for easy generation of ground truth data even in complex scenes.

In chapter 4, we calibrated the performance of image descriptors using synthetic and real world images of the Statue of Liberty, showing that the relative performance ranking is the same. We used the virtual city dataset proposed in the previous chapter to study the performance of different image descriptors under controlled lighting and viewpoint conditions. We found that the DAISY descriptors performs best under changes of viewpoint and illumination. Comparing the DAISY, SIFT, and GLOH

descriptors, we found that the number of the pooling regions has an impact on the performance when there are changes in viewpoint, whereas their spatial arrangement affects the performance under changes in illumination. Finally, we proposed augmenting the descriptors in the presence of depth information and showed that even quantized low-resolution depth maps can lead to a significant improvement in performance.

The performance of the image descriptors is closely coupled to the selection of good interest points, especially in applications that require the use of a sparse set of distinctive points. In the past, detectors have been hand crafted based on heuristics of the "interestingness" of the point or region. In chapter 5, we proposed to use machine learning techniques to train detector models that would be fine tuned for a given feature descriptor. We defined a measure of what good feature points are given a feature descriptor using the virtual city dataset proposed in chapter 3. Then we produced large dataset of labeled positive and negative examples of interest points and trained classifier models using a linear support vector machine. Using different evaluation criteria, we compared the performance of the learned models to those of two of the popular detectors - DoG and Harris. We showed that our trained model based on the SIFT descriptor produces a more distinctive and globally separable set of points than do the DoG and Harris detectors.

In chapter 6, we propose a system for extending an image beyond its boundaries under a given camera motion. We study the performance of different image features, either global or quantized densely extracted local features, as a representation of the scene in two specific tasks - scene matching and prediction. The matching task shows how well the features capture the essence of the scene and tests if our image database is large and varied enough to find images of similar scenes. The prediction task tells us if there is enough structure in the images of the given class to be able to make predictions about what lies beyond the boundaries of the image. We focus on a specific class of images - city street scenes. We found that quantized densely extracted HOG features

performed best in both the matching and prediction task.

In the final chapter, we used the transformed image retrieval method proposed in chapter 6 to build a large photorealistic virtual space represented as an image graph based on the predictions for different camera motions. The graph can be used for creating seamless transitions between images in several different applications: building infinite panoramas, exploring the photo collection as a virtual 3D space, image taxi (finding a path between two images in the database). We also provided an extensive analysis of the properties of the image graph.

In the future, we envision extending this work in two concrete ways. Given the large amount of available ground truth data from our virtual city, we can extend the application of learning techniques to the description stage as well, discovering more robust and distinctive image features both in terms of detection and representation. In addition, we can extend the dataset itself to include more scenes, more viewpoints/scales, different atmospheric conditions, people, varied amount of clutter and use it in the design and evaluation of wider variety of computer vision algorithms.

# Bibliography


[1] Adobe Pixel Bender. 2009. http://labs.adobe.com/technologies/pixelbender/. 148

[2] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski. Photographing long scenes with multi-viewpoint panoramas. *ACM Transactions on Graphics*, 2006. 100

[3] A. Agrawal, R. Raskar, and R. Chellappa. What is the range of surface reconstructions from a gradient field? In *Proc. European Conf. Computer Vision*, 2006. 130

[4] F. Attneave. Some informational aspects of visual perception. *Psychological Review*, 61:183–193, 1954. 25

[5] Autodesk. 3ds max. http://usa.autodesk.com/3ds-max, 2012. 56

[6] Francis R. Bach, Romain Thibaux, and Michael I. Jordan. Computing regularization paths for learning multiple kernels. In *Advances in Neural Information Processing Systems*, 2004. 107

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proc. European Conf. Computer Vision*, pages 404–417, 2006. 40

[8] P. R. Beaudet. Rotationally invariant image operators. In *International Conference on Pattern Recognition*, 1978. 39, 77, 84

[9] Jeffrey S. Beis and David G. Lowe. Indexing without invariants in 3d object recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21:1000–1015, 1999. 50

[10] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002. 46

[11] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2005. 26

[12] S.M. Bileschi and L. Wolf. Image representations beyond histograms of gradients: The role of gestalt descriptors. pages 1–8, 2007. 49

[13] A. Bosch, A. Zisserman, and X. Munoz. Scene classification using a hybrid generative/discriminative approach. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(4), 2008. 103, 114

[14] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010. 47

[15] Matthew Brown. Multi-image matching using multi-scale oriented patches. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 510–517, 2005. 50

[16] Matthew Brown and David. G. Lowe. Recognising panoramas. In *Proc. IEEE Int. Conf. Computer Vision*, 2003. 42, 61, 77

[17] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communication*, 31(4):532–540, 1983. 147

[18] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *Proc. European Conf. Computer Vision*, September 2010. 47

[19] Gustavo Carneiro. The automatic design of feature spaces for local image descriptors using an ensemble of non-linear feature extractors. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010. 47

[20] Cheng Chen, Yueting Zhuang, and Jun Xiao. Adaptive and compact shape descriptor by progressive feature combination and selection with boosting. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008. 47

[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001. 138, 139, 144

[22] J. Coughlan and A. L. Yuille. Manhattan world: Orientation and outlier detection by bayesian inference. *Neural Computation*, 15:1063–1088, 2003. 128

[23] Franklin C. Crow. Summed-area tables for texture mapping. In *Proc. ACM SIGGRAPH*, pages 207–212, 1984. 40

[24] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005. 42, 45, 61, 63, 77, 105, 114

[25] K. Dale, M. K. Johnson, K. Sunkavalli, W. Matusik, and H. Pfister. Image restoration using online photo collections. In *IEEE International Conference on Computer Vision (ICCV)*, 2009. 26

[26] J. Deutscher, M. Isard, and J. MacCormick. Automatic camera calibration from a single manhattan image. In *Proc. European Conf. Computer Vision*, pages 175–188, 2002. 128

[27] S. Divvala, D. Hoiem, J. Hays, A. Efros, and M. Hebert. An empirical study of context in object detection. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2009. 99

[28] Gy. Dorko and C. Schmid. Selection of scale-invariant parts for object class recognition. In *Proc. IEEE Int. Conf. Computer Vision*, pages 634–640, 2003. 25

[29] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proc. ACM SIGGRAPH*, 2001. 130

[30] P. Erdos and A. Renyi. Random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Science*, 5:17–61, 1960. 137

[31] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008. 85

[32] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008. 63, 86, 105

[33] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 264–271, 2003. 25, 42, 61

[34] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Simultaneous object recognition and segmentation by image exploration. In *Proc. European Conf. Computer Vision*, 2004. 25, 42, 61

[35] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The qbic system. *Computer*, 28(9):23–32, 1995. 124

[36] Per-Erik Forssén. Maximally stable colour regions for recognition and matching. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Minneapolis, USA, June 2007. 41

[37] W Forstner and E Gulch. *A fast operator for detection and precise location of distinct points, corners and centres of circular features*, pages 281–305. 1987. 38, 84

[38] W. T. Freeman, D. Anderson, P. Beardsley, C. Dodge, H. Kage, K. Kyuma, Y. Miyake, M. Roth, K. Tanaka, C. Weissman, and W. Yerazunis. Computer vision for interactive computer graphics. *IEEE Computer Graphics and Applications*, 18:42–53, 1998. 45, 61

[39] N. Gershenfeld. *The nature of mathematical modeling*. Cambridge University, 1998. 129

[40] Google. Google Street View Pittsburgh data set. In *http://maps.google.com/help/maps/streetview/*, 2008. 100, 111

[41] S. J. Gortler, R. Grzeszczuk, Szeliski R., and M. F. Cohen. The lumigraph. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA, 1996. ACM. 123

[42] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Efficient learning with sets of features. *J. Mach. Learn. Res.*, 8:725–760, May 2007. 49

[43] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988. 37, 64, 77, 84, 87

[44] J. Hays and A. A Efros. Scene completion using millions of photographs. *Proc. ACM SIGGRAPH*, 26(3):4:1–4:7, 2007. 26, 49, 99, 100, 107, 118

[45] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics*, 26(3):4:1–4:7, 2007. 127

[46] D. Hoiem, A. Efros, and M. Hebert. Geometric context from a single image. In *Proc. IEEE Int. Conf. Computer Vision*, 2005. 106

[47] D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006. 106

[48] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4), 1987. 46

[49] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 506–513, 2004. 43

[50] Les Kitchen and A Rosenfeld. Gray-level corner detection. *Pattern Recognition Letters*, 1(2):95–102, 1982. 77, 84

[51] J. Kopf, M. Uyttendaele, O. Deussen, and M. Cohen. Capturing and viewing gigapixel images. *ACM Transactions on Graphics*, 26(3), 2007. 124

[52] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Proc. IEEE Int. Conf. Computer Vision*, 2009. 50

[53] J. F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi. Photo clip art. *ACM Transactions on Graphics*, 26(3):3:1–3:10, 2007. 128

[54] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006. 26, 42, 48, 49, 61, 103, 106, 114

[55] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Affine-invariant local descriptors and neighborhood statistics for texture recognition. In *Proc. IEEE Int. Conf. Computer Vision*, pages 649–655, 2003. 46, 61

[56] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, New York, NY, USA, 1996. ACM. 123

[57] Tony Lindeberg. Feature detection with automatic scale selection. *Int. Journal of Computer Vision*, 30:79–116, November 1998. 84

[58] C. Liu, J. Yuen, and A. Torralba. Dense scene alignment using SIFT flow for object recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009. 26

[59] D. Lowe. Object recognition from local scale-invariant features. In *Proc. IEEE Int. Conf. Computer Vision*, September 1999. 25, 39, 42, 61, 62, 63, 64, 77, 83, 84, 87, 105, 114

[60] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60:91–110, 2004. 39

[61] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the 7th Int. Joint Conf. on Artificial Intelligence*, pages 674–679, 1981. 129

[62] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1–8, 2008. 26

[63] Jitendra Malik, Serge Belongie, Jianbo Shi, and Thomas Leung. Textons, contours and regions: Cue integration in image segmentation. In *Proc. IEEE Int. Conf. Computer Vision*, pages 918–925, 1999. 106

[64] Javier Marin, David Vazquez, David Geronimo, and Antonio M. Lopez. Learning appearance in virtual scenarios for pedestrian detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007. 53

[65] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. British Machine Vision Conference*, pages 384–393, 2002. 41

[66] Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. *Int. Journal of Computer Vision*, 60(1):63–86, 2004. 78, 87

[67] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27:1615–1630, 2005. 26, 28, 37, 39, 43, 51, 57, 62, 63, 77

[68] Pierre Moreels and Pietro Perona. Evaluation of features detectors and descriptors based on 3d objects. In *Int. Journal of Computer Vision*, pages 800–807, 2005. 26, 53, 77, 78

[69] Greg Mori, Xiaofeng Ren, Alexei A. Efros, and Jitendra Malik. Recovering human body configurations: Combining segmentation and recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 326–333, 2004. 40

[70] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340, 2009. 50

[71] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int. Journal of Computer Vision*, 42(3):145–175, 2001. 26, 42, 48, 61, 104, 126

[72] D. Parikh and C. L. Zitnick. The role of features, algorithms and data in visual recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010. 25

[73] P. Perez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, 2003. 130

[74] Isard M. Sivic J. Philbin, J. and A. Zisserman. Descriptor learning for efficient retrieval. In *Proc. European Conf. Computer Vision*. 47

[75] J. Ponce, M. Hebert, C. Schmid, and A. Zisserman. *Towards category-level object recognition*, volume 4170. Springer, 2006. 123

[76] P. Pritchett and A. Zisserman. Wide baseline stereo matching. In *Proc. IEEE Int. Conf. Computer Vision*, pages 754–760, January 1998. 25, 42, 61

[77] P. Rademacher and G. Bishop. Multiple-center-of-projection images. *ACM Transactions on Graphics*, 1998. 100

[78] Maxim Raginsky and Svetlana Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Advances in Neural Information Processing Systems*, 2009. 50

[79] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *Proc. IEEE Int. Conf. Computer Vision*, pages 10–17, 2003. 40

[80] Eleanor Rosch, Carolyn B. Mervis, Wayne D. Gray, David M, and Penny Boyesbraem. Basic objects in natural categories. *Cognitive Psychology*, 1976. 46

[81] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *In European Conference on Computer Vision*, pages 430–443, 2006. 78

[82] C. Rother, L. Bordeaux, Y. Hamadi, and A. Blake. Autocollage. In *Proc. ACM SIGGRAPH*, 2006. 124

[83] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efiňĄcient alternative to SIFT or SURF. In *Proc. IEEE Int. Conf. Computer Vision*, November 2011. 47

[84] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. Adaptive color-image embeddings for database navigation. In *'Asian Conference on Computer Vision*, pages 104–111, 1998. 124

[85] Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng. Robotic grasping of novel objects using vision. *International Journal of Robotics Research*, 27:157–173, February 2008. 53, 54

[86] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *Int. Journal of Computer Vision*, 37:151–172, 2000. 26, 37, 51, 77, 87, 88

[87] B. Scholkopf and A. Smola. *Learning with Kernels*. 2002. 103

[88] Gregory Shakhnarovich, Paul Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proc. IEEE Int. Conf. Computer Vision*, page 750. 50

[89] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007. 64, 105

[90] Eli Shechtman and Michal Irani. Matching local self-similarities across images and videos. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007. 47

[91] Jianbo Shi and Carlo Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994. 37, 84

[92] J. Sivic, B. Kaneva, A. Torralba, S. Avidan, and W. T. Freeman. Creating and exploring a large photorealistic virtual space. In *First IEEE Workshop on Internet Vision*, 2008. 49, 99

[93] J. Sivic and A. Zisserman. Efficient visual search for objects in videos. *Proceedings of the IEEE*, 96(4):548–566, 2008. 124

[94] S. M. Smith and J. M. Brady. Susan - a new approach to low level image processing. *Int. Journal of Computer Vision*, 23:45–78, 1995. 38

[95] N. Snavely, R. Garg, S. Seitz, and R. Szeliski. Finding paths through the world's photos. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008)*, 27(3):11–21, 2008. 125

[96] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *Proc. ACM SIGGRAPH*, 2006. 51, 100

[97] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *Proc. ACM SIGGRAPH*, 2006. 124, 126

[98] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Computer Vision*, 2006. 109, 111, 129

[99] Geoffrey R. Taylor, Andrew J. Chosak, and Paul C. Brewer. OVVV: Using virtual worlds to design and evaluate surveillance systems. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007. 53

[100] Matthew Toews and William M. Wells III. Sift-rank: Ordinal description for invariant feature correspondence. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 172–177, 2009. 44

[101] Engin Tola, Vincent Lepetit, and Pascal Fua. A fast local descriptor for dense matching. 2008. 42, 44, 61, 63, 77

[102] A. Torralba. Contextual priming for object detection. *Int. Journal of Computer Vision*, 53:169–191, 2003. 99

[103] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008. 50, 106, 107, 118

[104] A. Torralba and P. Sinha. Statistical context priming for object detection. In *Proc. IEEE Int. Conf. Computer Vision*, pages 763–770, 2001. 128, 129

[105] Leonardo Trujillo and Gustavo Olague. Automated design of image operators that detect interest points. *Evolutionary Computation*, 16:483–507, December 2008. 78

[106] TurboSquid. Library of 3D products. In *http://www.turbosquid.com/*, 2010. 55

[107] Tinne Tuytelaars and Luc Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *Proc. British Machine Vision Conference*, pages 412–425, 2000. 25, 42, 61, 77

[108] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2007. 26

[109] N. Vasconcelos. From pixels to semantic spaces: Advances in content-based image retrieval. *IEEE Computer*, 40(7):20–26, 2007. 124

[110] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008. 64

[111] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008. 87

[112] P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004. 123

[113] Paul Viola and Michael Jones. Robust real-time object detection. *Int. Journal of Computer Vision*, 57(2):137–154, 2002. 40

[114] C. Wallraven, B. Caputo, and A. Graf. Recognition with local features: the kernel recipe. In *Proc. IEEE Int. Conf. Computer Vision*, 2003. 49

[115] H. Wang and J. Brady. Corner detection for 3d vision using array processors. *Proceedings from BARNAIMAGE*, 1991. 84

[116] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, 2009. 50, 107

[117] S. Winder, G. Hua, and M. Brown. Picking the best daisy. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009. 26, 28, 45, 52, 58, 59, 63, 66, 67, 68

[118] Simon A. J. Winder and Matthew Brown. Learning local image descriptors. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1–8, 2007. 26, 47, 52, 62, 63, 65, 77, 78, 89

[119] D. N. Wood, A. Finkelstein, J. F. Hughes, C. E. Thayer, and D. Salesin. Multi-perspective panoramas for cel animation. *ACM Transactions on Graphics*, 1997. 100

[120] Changchang Wu, Brian Clipp, Xiaowei Li, Jan-Michael Frahm, and Marc Pollefeys. 3D model matching with viewpoint-invariant patches (VIP). In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008. 45, 61

[121] J. Xiao, J. Hays, K. A. Ehinger, A. Torralba, and A. Oliva. SUN database: Large scale scene recognition from Abbey to Zoo. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010. 42, 45, 61, 104, 107