Effects of 2-D Preprocessing on Feature Extraction

Accentuating Features by Decimation, Contrast Enhancement, Filtering

Christopher Tsai
{ tsaic@stanford.edu }
Department of Electrical Engineering, Stanford University

***Abstract*—Feature extraction assumes a number of forms in a number of applications. In this paper, we improve feature extraction by not only increasing the number of quality features that one can extract but also ensuring that the features we do extract are, indeed, representative high-quality features instead of false, minute, or noise features. We show that higher frequencies do not, for the purposes of feature extraction, necessarily represent human-salient features and that the combination of contrast enhancement, decimation, and lowpass filtering achieve more robust feature extraction than simple high-frequency boosting. Our ideal feature extractor therefore incorporates a decimator for reduction to an idealized size, contrast enhancement through stretched dynamic range, and frequency-domain filtering with a Gaussian lowpass filter.**

***Index Terms*—image feature identification, feature extraction, object recognition, contrast enhancement, dynamic range, decimation, downsampling, lowpass filtering, Gaussian filtering, two-dimensional Fourier transform, frequency domain filtering, spatial domain convolution**

I. Introduction

Features prove useful in a number of applications in electrical engineering, computer vision, image processing, and image compression. For almost any application that requires robotic motion, autonomous navigation, object identification, human simulation, emotion detection, face recognition, or image classification, features inform the subject of a picture, separating distinguishing key points from more generic background. Similarly, features encapsulate the most crucial information in an image with a much smaller vector, thereby providing quite a reliable compression if the image itself is not the object of an operation; instead of communicating an entire image, pixel value by pixel value, computers and cell phones can instead send feature vectors, minimizing the amount of transmission necessary to communicate key information. For example, identifying the subject of an image, the location of a building, the color of an object, or the artist of a painting or album does not necessarily require

However, scientists and researchers have been trying to improve feature extraction for decades, attempting to find a better way of isolating the most human-salient or visually cognitive features possible without manual selection. In other words, we have not yet optimized unsupervised feature extraction from real images. Algorithms like SIFT and SURF have emerged as preferred feature extractors, but application-specific enhancements continue to surface in new developments.

Despite the particularity of several of these algorithms – the limitations of their use, and the specific nature of their applications – general techniques for preparing an image for feature extraction exist and provide solid foundations on which several different algorithms, regardless of their application, can build. For example, almost any feature extraction routine would want to extract *more* features, as long as they are not extraneous noise features or red herrings. Furthermore, enhancing the most salient features would also be a universal boon, just as finding more cross-checking features would ubiquitously improve robustness, especially when RANSAC requires geometric consistency.

Thus, using the wildly popular Speeded Up Robust Features (SURF) algorithm as our primary feature extractor, we propose several preprocessing techniques in both the spatial and frequency domains that enable any human-perception-based feature extractor to extract more features, ascertain feature quality, and avoid spurious false features. All in all, we seek not only to improve the relevance of the features we do find but also to reduce the possibility of overemphasizing lesser features, such as unrepeatable specks and minute nigh-imperceptible details. The three-step pre-processor hence proposed successfully meets these requirements.

Finally, a summary of our results in two real-time systems – building and CD cover identification – confirms the marked improvement that the three steps individually and collectively precipitate.

II. Feature Extraction: The SURF Algorithm

A. Speeded Up Robust Features (SURF) [1]

Bay, Tuytelaars, and Van Gool developed this scale-invariant, rotation-invariant feature detector as a generally applicable method for emulating human connotative association of image keypoints [1]. An extension of Lowe's Scale Invariant Feature Transform (SIFT), introduced in 1999, SURF works more quickly by essentially reducing the dimensionality of the feature descriptor used in SIFT [2]. Lowe's feature descriptor simplified the Laplacian of Gaussian (LoG) with a Difference of Gaussians (DoG) filter [2]; because the difference operation requires less computation than a trace (the Laplacian), the filter combs the target image more quickly. Like Lowe's SIFT, Bay's SURF approximates a more complex filter to expedite computation. Instead of computing the Hessian matrix with a second derivative

$$H(\boldsymbol{x},\sigma) = \begin{bmatrix} L_{xx}(\boldsymbol{x},\sigma) & L_{xy}(\boldsymbol{x},\sigma) \\ L_{xy}(\boldsymbol{x},\sigma) & L_{yy}(\boldsymbol{x},\sigma) \end{bmatrix}$$

where $L_{xx}(\boldsymbol{x},\sigma)$ represents the convolution of image I and the Gaussian second order derivative $\frac{\partial^2}{\partial x^2} g(\boldsymbol{x},\sigma)$ at each point $\boldsymbol{x}$, SURF instead approximates the Hessian with 9×9 box filters:

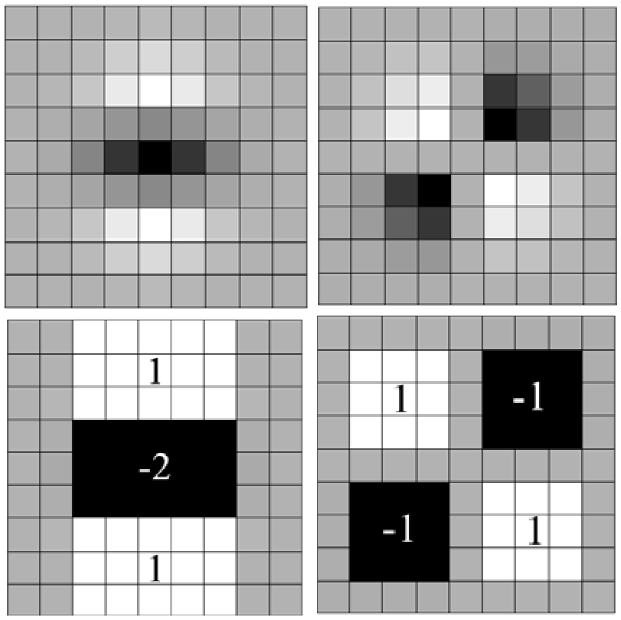

Fig. 1. Box filter approximation (lower) of the Hessian filters (upper) **[1]**

By simplifying the more complex, nuanced second-order Gaussian derivative with ternary box filters, the Hessian determinant that we must compute at each point reduces to

$$\det(H_{approx}) = D_{xx}(\boldsymbol{x},\sigma)D_{xy}(\boldsymbol{x},\sigma) - 0.81 \cdot D_{xy}^2(\boldsymbol{x},\sigma)$$

where the factor of 0.81 balances the relative weights in the Hessian's determinant when converting from the Gaussian filter to the box filter.

Instead of iteratively smoothing and subsampling the image with a pyramid of Gaussians, the SURF algorithm iteratively upscales the filter size beyond 9 × 9, into 15 × 15, 21 × 21, 27 × 27, as long as features of the increased scales prove interesting to the application. In other words, the upscaled filter dimensions respond and highlight features of higher dimensions themselves; features that, for example, cover a larger area than a 9 × 9 filter can possibly distinguish. These larger neighborhoods extract differently sized features, allowing the extraction to respond to features of all sizes. The points that yield highest response, after non-maximum suppression in 3 × 3 × 3 neighborhoods, are termed interest points:

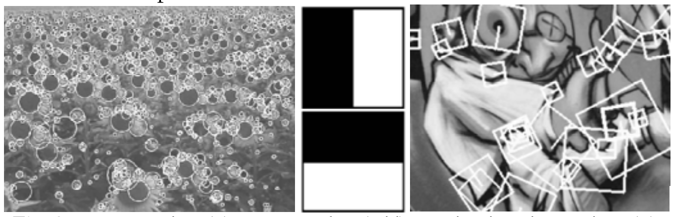

Fig. 2. Interest points (L), Haar wavelets (mid), rotation-invariant regions (R)

The combination of localized information and gradient-related features results in scale-invariance

B. The SURF Descriptor/Feature Vector

Features involve more than simple interest point identification. Additionally, one must also produce rotation-invariant detection, so Haar wavelets of both horizontal and vertical variation enter play. Concentrating on the 4 × 4 region around each interest point, and deeming d_x the response to the horizontal Haar wavelet and d_y the response to the vertical Haar wavelet, SURF quantifies the degree of intensity variation around the interest point as well as the polarity of the intensity changes through $\sum|d_x|$ and $\sum|d_y|$. SURF then compactly represents this information in the feature vector **v** = $\left(d_x, d_y, \sum|d_x|, \sum|d_y|\right)$.

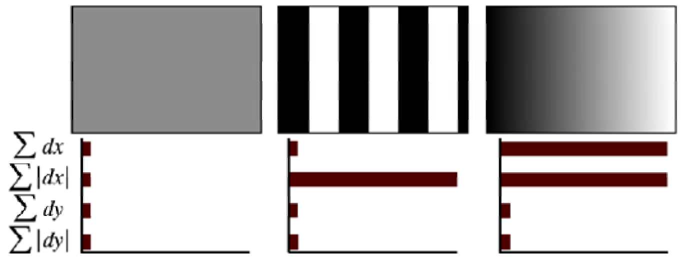

Fig. 3 Descriptor vectors for constant, striped, gradually fading regions **[1]**

Descriptor vectors provide a comparable observation that we can juxtapose between two images. When two corresponding interest points share similar feature vectors, we can hypothesize an affine transformation that will may one into the other. When several pairs of interest points in a pair of images seem to map under the same (or similar) 3-D or 2-D affine transformation, then the model holds, and the two images likely "match."

C. RANdom SAmple Consensus (RANSAC)

Generally not a part of feature extraction until the matching phase, RANSAC is a widely applied algorithm that derives an optimal set of model coefficients from a set of data. While the model is not guaranteed to converge, for all database-query sets that bear reasonable resemblance or similar origin, the model will converge after sufficient iterations. The algorithm proceeds by repeating the following steps to estimating the optimal model:

1.) *n* data points randomly selected to resolve free parameters
2.) Generate 3D affine model on sample points
3.) Test other points against this model
4.) All points with small error are inliers
5.) Compute average error of all inliers
6.) Re-estimate model with inliers included
7.)Repeat steps 3-6 until error is tolerably small, nondecreasing

Eventually, as the model converges for a pair of matching images, computation accelerates to reflect the inclusion of many interest points behaving under the same model. RANSAC will soon distinguish outliers and eliminate them from model computation, allowing the main affine transformation to emerge, if one exists. Even if a significant number of outliers pollute the data, RANSAC generally proves robust in extracting the predominant transformation coefficients after a finite number of iterations. The main disadvantage is the variation of parameters (error threshold, number of iterations) from application to application, but, once we find an optimal set for the current problem, we can repeat RANSAC with the same parameters for other images from the same class or database.

III. Preprocessing System Overview

Our system aims to improve feature extraction and hence matching accuracy by operating primarily on the input image.

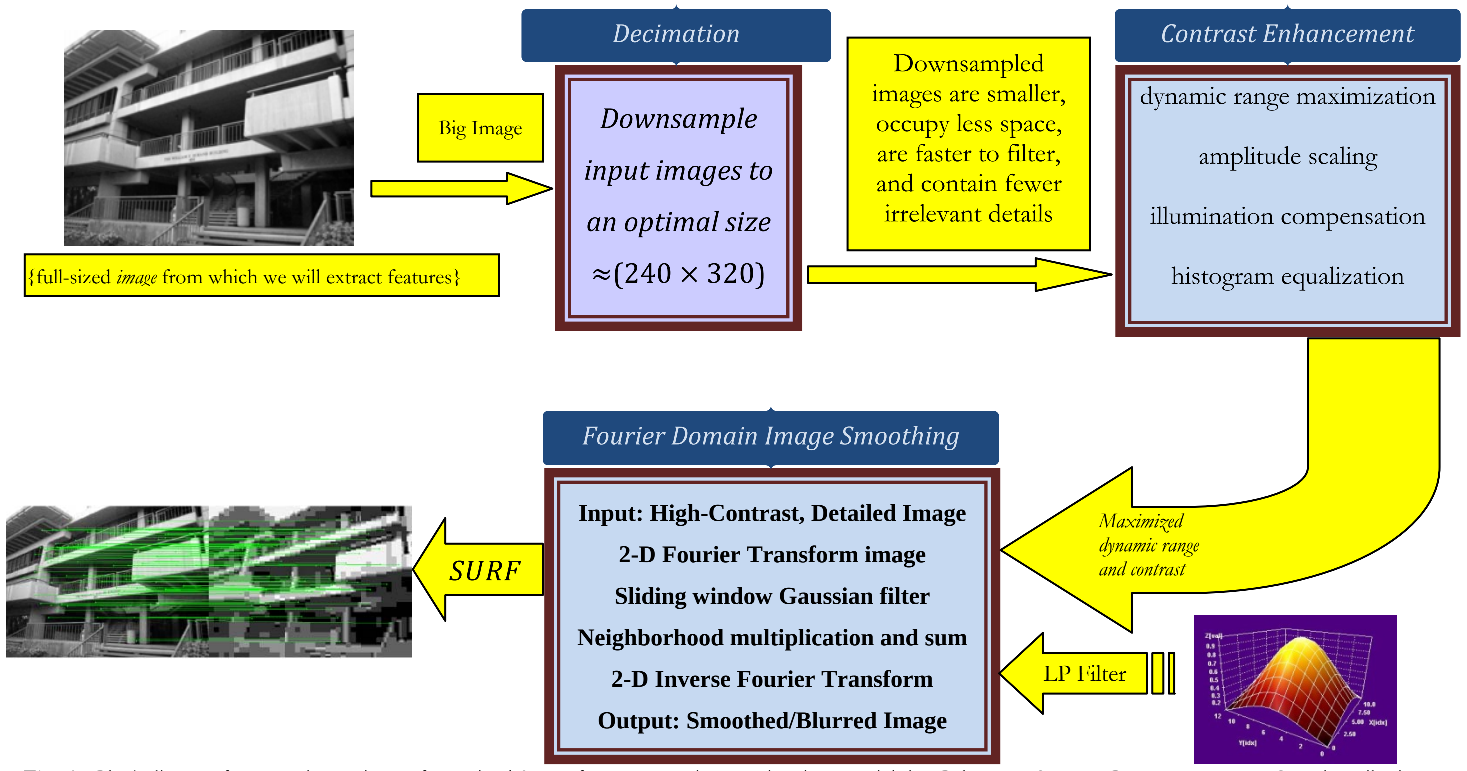
Decimation
Big Image
Downsample input images to an optimal size ≈(240 × 320)
{full-sized *image* from which we will extract features}
Downsampled images are smaller, occupy less space, are faster to filter, and contain fewer irrelevant details
Contrast Enhancement
dynamic range maximization
amplitude scaling
illumination compensation
histogram equalization
Maximized dynamic range and contrast
Fourier Domain Image Smoothing
Input: High-Contrast, Detailed Image
2-D Fourier Transform image
Sliding window Gaussian filter
Neighborhood multiplication and sum
2-D Inverse Fourier Transform
Output: Smoothed/Blurred Image
SURF
LP Filter

Fig. 4. Block diagram for preparing an image for optimal SURF feature extraction: Decimation → Lighting Balance & Contrast Boost → Lowpass Gaussian Filtering

A. Decimation

Before we begin filtering the image with two-dimensional imaging techniques, we downsample the image to a standard small size (we designate 240 × 320 for reasons to be discussed). Decimation serves three primary purposes.

First of all, a smaller image consumes less storage space. Even though modern computers boast gargantuan memory sizes, portable devices such as cellular phones and personal data assistants still require compression. One of the central goals of feature extraction is information compression, from the sender to the receiver. The feature vector – excluding any additional encoding that might occur for a communication channel – is the ultimate compression, but we start earlier by discarding whatever information we do not need.

Secondly, smaller images require fewer computations to process. Since contrast enhancement and lowpass filtering are mathematical operations, we can expedite their execution by giving them smaller inputs.

Thirdly, extraction of reliable, robust, representative features depends heavily on the scale of features we make available. Small, minute details that would be hard-pressed to appear in two slightly different versions of the same image – perhaps taken with slightly different lighting or from a slightly different angle – need not register in SURF. However, if we process the large image, these features remain, so we can eliminate them outright and focus on the more reliable intermediate-sized and region-based features through decimation. Of course, we lose information when we eliminate pixels, but, if a feature occupied only a few pixels in the original image, it was likely either noise or imperceptible, let alone an unlikely keypoint to appear in a similar image.

B. Contrast Enhancement

Oftentimes, real images are taken under imbalanced lighting conditions and suboptimal backgrounds – suboptimal in the sense that the colors and brightness behind the objects do not necessarily accentuate the features of interest as much as SURF might prefer. Mathematically, this suboptimality reflects an incomplete dynamic range. The image values occupy only a portion of the available values – 256 in an eight-bit image – so we can stretch the range. Because color images have three channels – generally Red, Green, and Blue (RGB) – we cannot simply stretch each dynamic range; the differing linear transformations would give the resultant picture a gray, washed-out appearance. Instead, we consider the superposition of all three histograms and preserve the nuances of each individual histogram by respecting the intervals in which one or two – but not all – channels are dominant. In essence, we stretch the dynamic range while preserving ratios between color channels.

In addition to color balancing, we also compensate for uneven illumination. Some settings – such as museums, art galleries, and poorly-lit store shelves – bias the picture strongly due to overhead or unidirectional illumination. As a result, some of the most salient features may not protrude desirably. By redistributing the areas of intense brightness over the entire image, features that might normally be clouded can appear and protrude as we expect of features.

C. Fourier Domain Image Smoothing

Finally, we experiment with various lowpass filters to smoothen our image, juxtaposing the results of feature-matching a smoothened image with those obtained from the original image.

Smoothening the image achieves two major goals.

Most noticeably, it denoises the image (when the noise is speckle or impulsive). Secondly, and more importantly for our purpose, lowpass filtering blurs out minute features that might prove spurious or outlandish in SURF+RANSAC. By reducing the effect of single-pixel extremities, this step allows SURF to identify more consistent features, ignoring ones that might be singular and therefore false. As counterintuitive as it may first appear, we do not want SURF to focus heavily on edges. Even though this sudden variation from one brightness value to another represents an important indicator of objects and features, the more revealing and indicative variations occur over a larger region; there may be sharp gradients, but they are sustained across an entire window (or two) instead of one or two pixels. Smoothening does dampen the sharpness of important edges, but the sharpest and most sustained edges will still remain noticeably abrupt to flag features, while "noise" features that mean little from a matching or representation standpoint will fade into the woodwork.

IV. Decimation

Decimating the image is a straightforward operation, since dropping samples only removes information; with no additions necessary to supplement the smaller images, we need not interpolate as we would when upsampling. The only art to downsampling is our choice of size. Thus, we compare the effects of downsampling on a variety of factors, starting with total feature count, the rawest measurement of success:

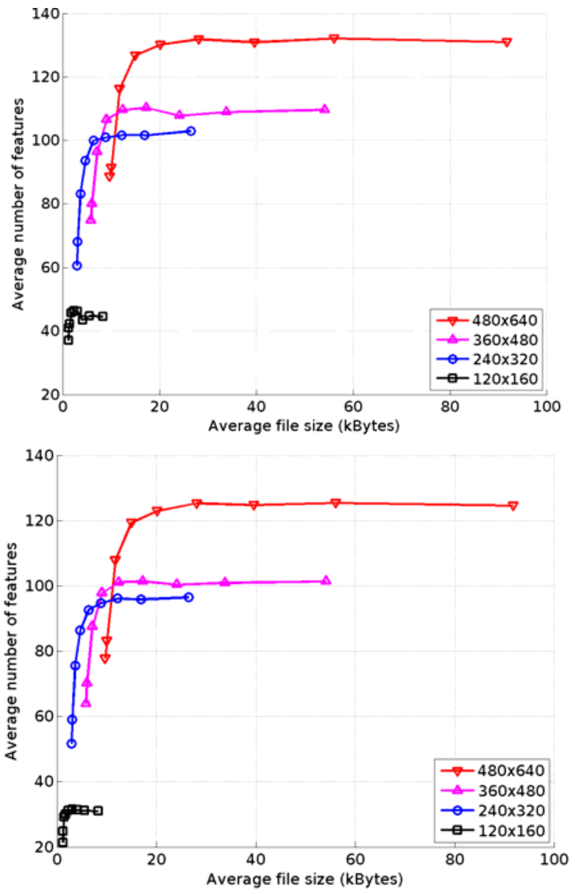

Fig. 5. Average number of features extracted as a function of rate and image size

The two plots reveal, unsurprisingly, that larger images yield more features. Two interesting points remain, however. The stability in the curve after the bend reveals that differently sized images respond differently to quantization. Because almost all computerized images receive some form of compression – typically some variation of JPEG – behavior with size is a good indicator of feature robustness. We notice that the largest images lose their features at the largest sizes; their bends occur at larger sizes than the bends of smaller images.

However, merely counting the number of features reveals nothing about the quality of features extracted. For that case, we need to consider also the number of false features, which we define as features that fail to correspond to features in the original, non-downsized image. In other words, when we compare features extracted from the original image to features extracted from the downsampled image, true features that appear in the downsampled version should correspond one-to-one to features in the original image; though the converse is not necessary true due to information loss, new features should not emerge *because* of downsampling. We define these questionably introduced features as *false* or *spurious* features.

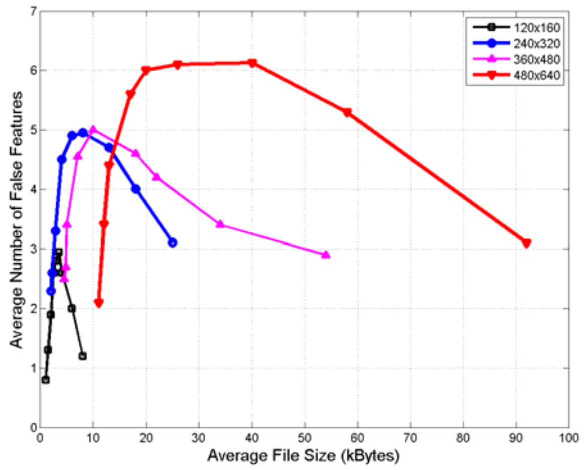

Fig. 6. Average number of false features as a function of rate and resolution

The larger images also yield more false features than their smaller counterparts; this behavior does not surprise us because more features overall remain when we have more pixels. However, the proportion of extracted features that are *real* (not false) does contain new information:

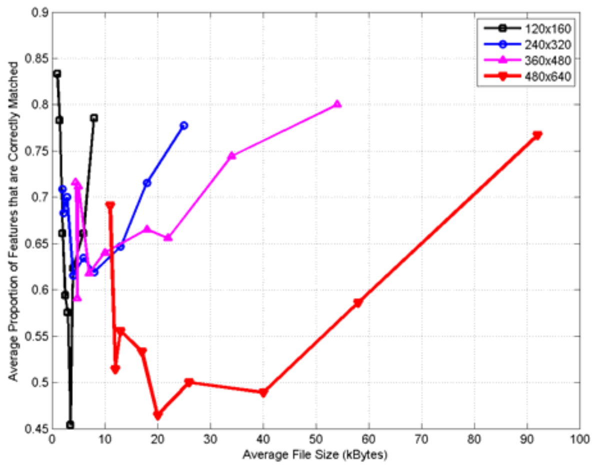

Fig. 7. Proportion of features that correspond correctly, vs. rate and resolution

The largest image actually yields the lowest percentage of true, genuine features. Thus, higher resolutions are not necessarily ideal for robust feature extraction. Despite the fact that larger images contain more information and naturally more SURF features, smaller images actually provide a more reliable subset of representative features that correspond to features in the original, normal-size image.

This seemingly strange behavior occurs for two reasons. For one, the larger images contain more "noise" features. These features do not necessarily masquerade as noise in the communications sense, although some of these features are undoubtedly the result of camera noise, since our image set originated from a suboptimal mobile phone camera. More likely, these noise features are minute details or features within features that really do not correspond to points that a human observer would logically label as a feature. For example, the shine on a rail, the stain on a tooth, the speck in the background, the dead pixel in the camera all represent minute details that SURF may capture in one resolution but not the other; these features may have intuitive meaning, but they certainly are not reliable sources of information. Smaller images do not suffer these inconsistencies because the removal of many pixels has wiped most minutiae from the image before it can raise a flag for the feature detector.

The other reason for the rapid increasing in false features (shown as a rapid decrease in good features) is the effect of quantization. When JPEG compresses an image, inevitably the procedure introduces some artifacts, most of them in the form of blocks:

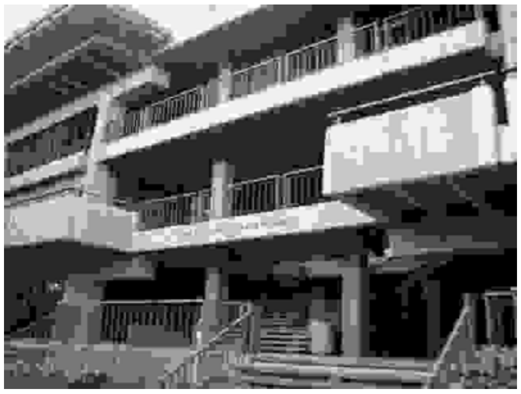
Fig. 8. Example of blocking artifacts following JPEG compression.

When an image is large, and its features numerous and minute, the slightest strength of a block could raise a flag and draw the attention of a sensitive feature extractor. Of course, the original, non-decimated image also suffers from the same blocking artifacts, but the even scarier consequence of this universal degradation is that block features could end up matching or corresponding to other block features in RANSAC. Even a generally sound algorithm like SURF will, if enough of them surface, detect and match blocking artifacts between two equally blocky images. Thus, with a larger image's plethora of pixels and features to latch onto, the emergence of blocking artifacts is nightmarish from a matching standpoint. The likelihood of a false match – or, more importantly, a completely meaningless match – is too high. Again, the smaller resolutions and more aggressive downsampling factors slightly allay this concern – although not completely, as the depressions in the curve indicate – simply by supplying fewer minute features onto which blocking artifacts can erroneously latch and match. Based on these curves, 240 × 320 and 360 × 480 seem like ideal image sizes for extracting features on these images using SURF. We use 240 × 320 for the remainder of the system, but the face images, painting images, and album cover images also respond most favorably to these downsampled sizes. Other feature extractors may beg to differ, but, for SURF, downsampling provides more than mere storage savings – it also removes unwanted features, increasing the proportion of features that represent authentic keypoints and hence improving the overall matching accuracy:

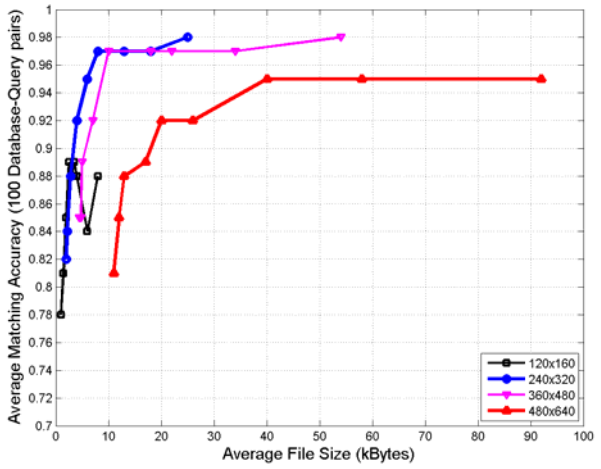

Fig. 9. Matching accuracy in 50-image test set as a function of rate & image size

V. Contrast Enhancement

We begin by balancing the color histograms and compensating for uneven illumination.

Illumination compensation for a color image can be a tricky affair, but we begin with the histogram of each color channel, and attempt to stretch it to its full dynamic range. Expanding the range requires caution, and we try two approaches: an equal treatment and transformation of each channel, termed the Gray-World approach, and a channel ratio-preserving approach.

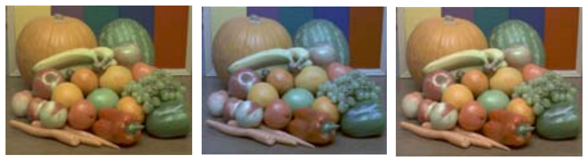
Fig. 10a. Illumination compensation for a color image: original (left), stretching to full dynamic range in each channel (middle), ratio-preserving stretch (right)

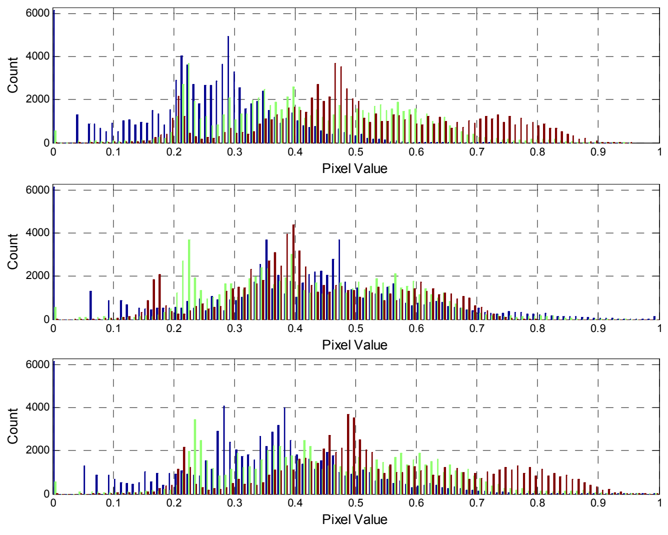

Fig. 10b. Tri-color channel histograms for original (left), stretching to full dynamic range in each channel (middle), ratio-preserving stretch (right)

The leftmost image corresponds to the uppermost histogram: an image whose individual color channels do not fill their entire dynamic ranges. The central version and central histogram represents transforming each channel individually to fill the range, resulting in a washed-out image. The channels individually, when gray-scaled, appear optimal, but the combination is suboptimal, causing color features that previously existed to change nature. Even if we eventually gray-scale the image before applying SURF, we do not want to sacrifice color-facilitated features this early in the process, so we opt for the rightmost image and lowermost histogram, which scales each channel *not* to its full range but rather to the maximum while preserving ratios present in the original histogram. Mathematically, this means computing the ratio of color channels at each bin and ensuring that the ratio is preserved – within a certain percentage – in the final transformation. Stretching occurs, as the extension of the blue channel reveals, but the stretching does not extend beyond the point at which new ratios emerge. With the ratios intact, the rightmost image looks balanced, with illumination permeating the entire arrangement of fruit used as our default test image.

Eventually, we choose to gray-scale our image in order to simplify the input to SURF. Multi-channel implementations of feature extraction exist, but we focus our attention on a gray-scale feature extractor that nevertheless derives edges from the original full-color image. Contrast enhancement for a gray-scale image – easily obtained from a color image – can assume a variety of forms. For one, we can treat the problem as a linear mapping:

If we seek to transform the mean to $\mu_{desired}$ (128 – the center of the range – is a reasonable choice) and the standard deviation to $\sigma_{desired}$ (80 – one third the full range – is a practical choice), then we can perform the linear transformation:

$$\begin{cases} \mu_{desired} = \alpha\,\mu + \beta \\ \sigma_{desired} = \alpha\,\sigma \end{cases}$$

Multiplying each pixel by α increases the dynamic range of the pixels, stretching both the value of the mean and the separation represented by standard deviation. To achieve desired centrality, we can further shift the image value distribution by adding a constant; however, shifting the distribution does not alter its variance, so the constant shift β does not disturb $\sigma_{desired}$. To solve this linear system, we can simply solve a linear equation:

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \begin{bmatrix}\mu & 1\\ \sigma & 0\end{bmatrix}^{-1} \begin{bmatrix}\mu_{desired}\\ \sigma_{desired}\end{bmatrix}$$

Therefore, in order to transform the image mean and standard deviation, we perform the following steps:

1.) Multiply each image pixel value by α,

2.) Add the constant β.

3.) Round negative values to zero and values above 255 to 255.

Another method for dynamic range maximization that parallels the gamma characteristic of image displays is the exponential transformation, as opposed to the linear one considered above. We use a pointwise operation for our exponential transform, which exponentiates each input image pixel value:

$$f_{out}(x,y) = f_{in}^{p}(x,y)$$

We heuristically substitute different values of *p* to scale the image most pleasingly. After scaling each pixel in the input image by *p*, we display it and examine the image quality, varying *p* before appraising the image again. Without knowing the specifics of our particular display, this heuristic substitution – as random as it may seem – is our best way to gradually converge to our ideal parameter *p*. For a black-and-white image, the exponential transform performs adequately, as we ascertained with a test on CD cover images (first color-balanced as detailed previously):

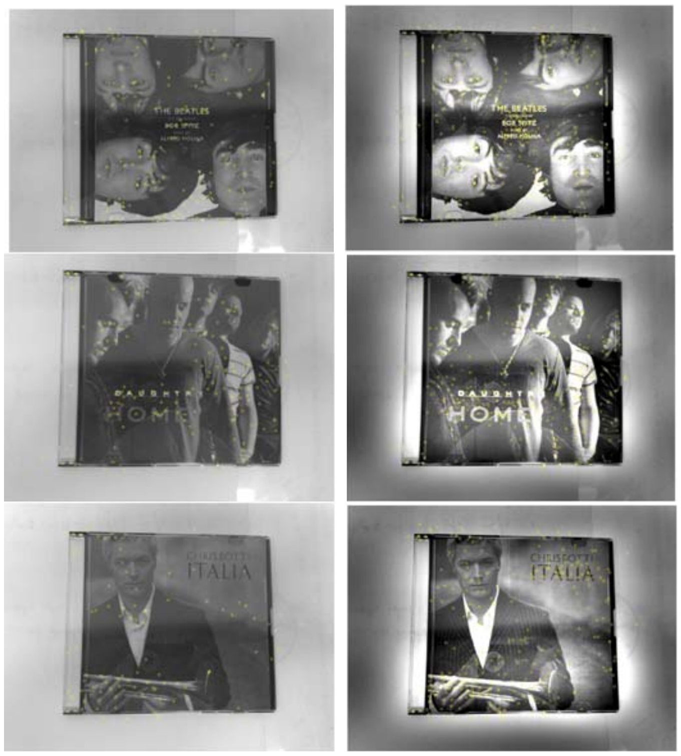

Fig. 11. Dull images (L) offer more features when dynamic range is stretched (R)

The green circles represent SURF-extracted features. Clearly, images that began preprocessing as gray and relatively washed-out renditions improved perhaps not visually but certainly practically following color balancing and exponential dynamic range stretching. In each case, the number of features increased; more importantly, the number of meaningful features proliferated, especially around text such as the title words.

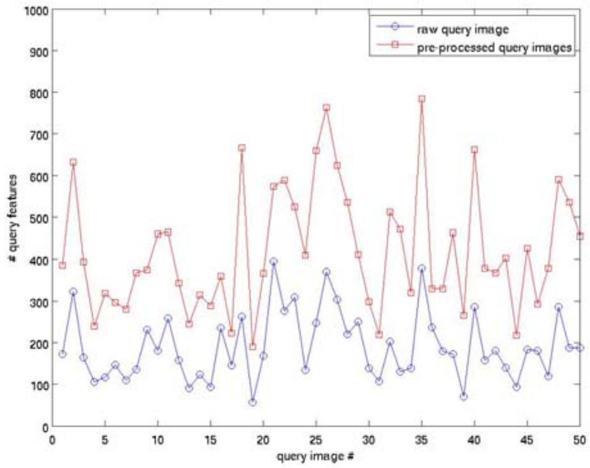

Fig. 12. More features appear, in clusters and new locations, after contrast boost

This type of feature multiplication occurred ubiquitously, across all images in the album data set. The plot above reveals that some more text-laden covers respond better than their more pictorial counterparts to illumination balancing and subsequent contrast enhancement.

Furthermore, the legitimacy of these features speaks for itself; the clustering of several features around the same keypoints indicate high importance and repeatability. For example, the text generates several interest point flags; even though the tip of a single letter is essentially one human feature, the plethora of green circles surrounding each piece of text speaks to the robustness and authenticity of the feature. This cross-confirmation of feature legitimacy proves one of the more unexpected results of contrast enhancement, but it allows us to rest assured that the additional features that this operation introduces are not false; some may be redundant, but, besides its utility in cross-checking feature validity, the redundancy also reduces the influence of the few outliers that inevitably appear.

VI. Fourier Domain Filtering

The final – and possibly most conventional – step to our three-step preparation is frequency-domain filtering. Downsampling reduces the effect of impulse noise and point features, but we seek a further simplification in order to accentuate slower variations and smoothen the blocking artifacts that result from compression (and decompression). Regardless of the type of lowpass filter we apply, the justification is identical: the human eye works much like a lowpass filter, blurring images – convolving them in the spatial domain. **[4]**

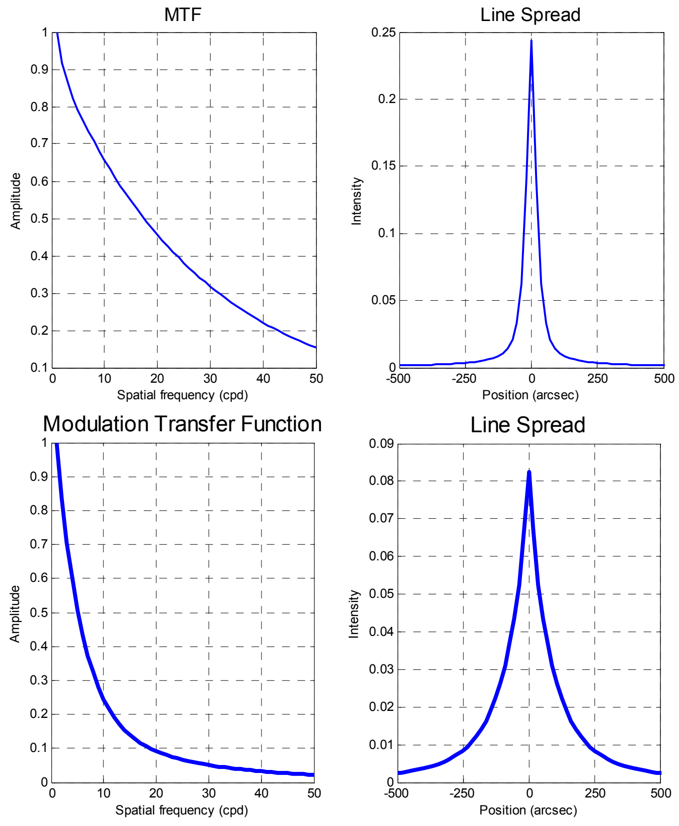

Fig. 13. The human eye, squinted (top) or not (bottom), is like a lowpass filter **[4]**

The pupil of the human eye is like a camera aperture; its impulse response and transfer function change as we widen or squint, but, however small we make our eyes, the impulse response is wider than an impulse, and the transfer function is non-constant, resulting in attenuation of higher frequencies. The upper plot represents a squinted eye (1.5 mm diameter pupil) relative to the lower plot, which represents an 8-mm opening. Because the impulse response is not an impulse, convolution with the image on our lens results in slight blurring, not unlike a lowpass filter. Hence, to simulate human vision prior to identifying SURF features, we lowpass filter our images as well.

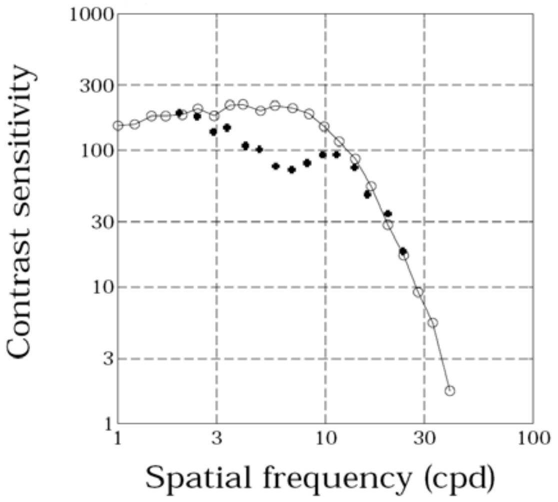

Fig. 14. Contrast sensitivity of human eye is higher for lower frequencies **[4]**

Note that the eye is also more sensitive to slower variations in contrast than to higher variations; in fact, several optical illusions like colored stripes arise from the decreased contrast sensitivity at higher frequencies. **[4]**

While traditional filtering with a Gaussian lowpass filter is a viable option, we can obtain a more visually representative result with the intricate soft-coring method:

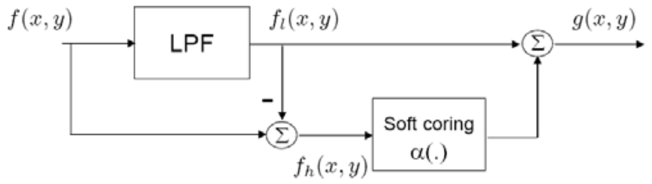

Fig. 15. Soft coring system to preserve essential high frequencies during lowpass

The lowpass filter designated as "LPF" in the system has the following kernel:

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & [1] & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Finally, we tap the soft coring function:

$$\alpha\left(f_h(x,y)\right) = m \times f_h(x,y)\left[1 - e^{-\left(abs\left(\frac{f_h(x,y)}{\tau}\right)\right)^{\gamma}}\right]$$

Because the soft coring function operates on the high-passed portion $f_h(x,y)$ of our original image – edges, lines, and noise – we can adjust the parameters *m*, γ, and τ to control how much we accentuate the high frequencies relative to the passed low frequencies in $f_l(x,y)$. This preserves some of high frequencies that might outline features and separate them from background.

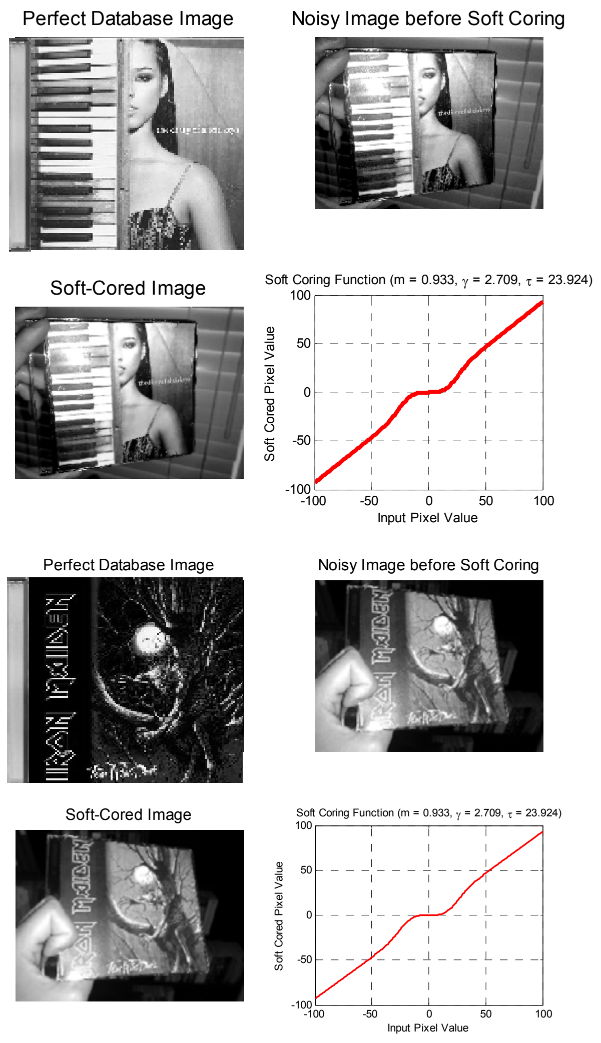

Fig. 16. Results of soft coring noisy images: sharper text, smoother regions

While the results of soft coring look more visually pleasing, it is a predominantly spatial operation; only the lowpass filtering portion can be performed in the frequency domain. To expedite processing when lowpass filtering is sufficient, we are faster performing processing as pointwise multiplication in the Fourier domain rather than two-dimensional convolution with a sliding window in the time domain, especially as our Gaussian filter begins increasing in size. For example, using the 9 × 9 Gaussian filter – a size commensurate with that of the smallest SURF window – frequency domain computation saves orders of magnitude. More intricate filters like the 21 × 21 Gaussian are simply impossibly computationally intensive in the time domain:

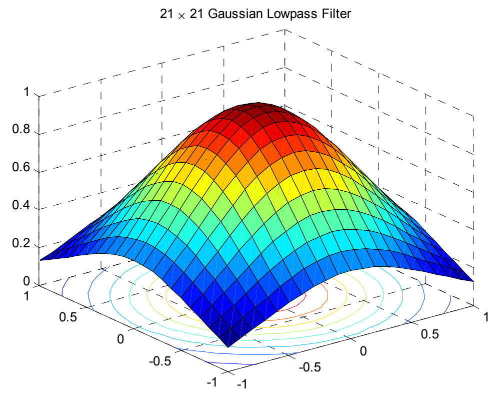

Fig. 17. 21 × 21 Gaussian filter used as sliding window in frequency domain

We witness an improvement in feature extraction after filtering:

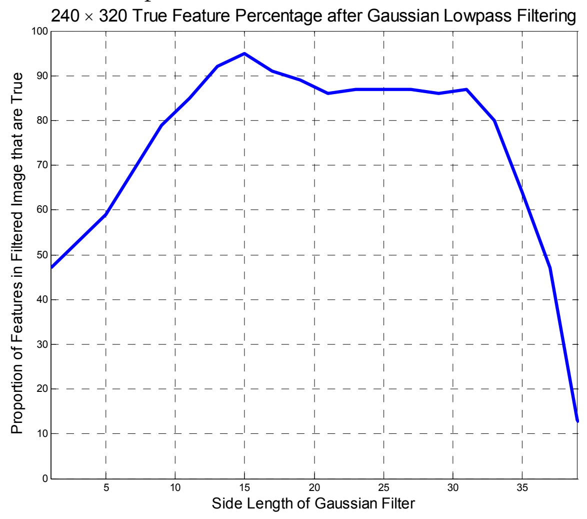

Fig. 18. Proportion of features that assist SURF, as a function of filter width

Generally, lowpass filtering with a small 15 × 15 filter seems to yield the highest proportion of usable, representative features, and the size falls sensibly within the filter sizes used to determine interest points in SURF. Smaller filters represent lower degree of smoothing; when the side length is unity, no filtering is performed; we can see the proportion of features that prove informative increase from about 50% to nearly 100%. Many of the corrupting spurious features vanish as we smooth them into the background, and we eliminate progressively more of these are we increase the smoothing window. At some point around 31 × 31, the window becomes too large, and key features lose distinction. As a result, the total number of features plummets, causing the percentage of true features to drop as well. However, all in all, the system improves the quality of extracted features and ultimately the matching accuracy of the album cover matching and building identification systems.

VII. Discussion and Conclusion

Each portion of our proposed three-step preprocessing system contributes some noticeable effect to the quality of extracted features. The initial decimation eliminates single-pixel noise features while reducing processing time. Illumination compensation and contrast enhancement accentuate text and reduce the influence of unevenly shaded background on otherwise salient features. Finally, lowpass filtering blurs away any remaining point features, which we seek to eliminate with the more regionally interested SURF extraction algorithm.

After a full execution of the algorithm on 50 building and 50 album cover images, a noticeable improvement in matching accuracy ensues. When performed on full-size, uncompensated, unfiltered images, the building matching accuracy is approximately 74%, while the album covers begin at 62%, a lower number due to disruptive flash. However, with the aggressive preprocessing applied (240 × 320 resizing, lighting balance, contrast enhancement, and 15 × 15 Gaussian lowpass filtering), the building matching accuracy soars to 98%, while 90% of the album covers in the 50-image test set correctly match despite the tilted angles and the suboptimal lighting.

We conclude that preprocessing, at least for the SURF extraction algorithm, significantly boosts our chances at locating reliable, robust features. Trials with other feature extraction algorithms would be a worthwhile next step.

REFERENCES

[1] Herbert Bay, Tinne Tuytelaars, Luc Van Gool. "SURF: Speeded Up Robust Features." *Proceedings of the Ninth European Conference on Computer Vision*, May 2006.

[2] David Lowe. "Object Recognition from Local Scale-Invariant Features." *Proceedings of the International Conference on Computer Vision 2*, pp. 1150 – 1157, 1999.

[3] David Lowe. "Distinctive Image Features from Scale-Invariant Keypoints." *International Journal of Computer Vision*, 60, 2, pp. 91-110, 2004.

[4] Brian Wandell. *Foundations of Vision*. Sinauer Associates, May 1995.

[5] R.C. Gonzalez, and R.E. Woods. *Digital Image Processing, Third Edition*. Upper Saddle River, NJ: Prentice Hall, 2007.

[6] R.C. Gonzalez, R.E. Woods, and S.L. Eddins. *Digital Image Processing using Matlab*. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[7] Ronald Bracewell. *Fourier Analysis and Imaging*. Springer, 2004.