Analysis of Distributed Smart Camera Calibration Accuracy using Minimal Data Transmission

Warwick Stone
Department of Electrical and
Computer Systems Engineering
Monash University, Melbourne, Australia

Y. Ahmet Şekercioğlu
Department of Electrical and
Computer Systems Engineering
Monash University, Melbourne, Australia
Email: ASekerci@ieee.org

***Abstract*—An algorithm for accurately estimating camera overlap in a distributed smart camera network with minimal data transmission was developed and evaluated, as a preliminary step towards three-dimensional calibration. The algorithm incorporates SURF (Speeded-Up Robust Features) to find matching features in live, low-quality images provided by web cameras. To reduce transmitted data, the number of extracted features was reduced, and feature descriptors were compressed using singular value decomposition. Reliable overlap estimation accurate to within 9 pixels was achieved while sending just 30 kilobytes per captured image.**

I. Introduction

As the cost of powerful embedded processors continues to drop, distributed smart camera networks are becoming more and more viable options in applications such as security and monitoring. With the ability to perform many user-defined tasks such as object recognition and tracking, intrusion detection and localization, smart cameras are likely to replace many older systems which rely on human monitoring of all video information.

The benefits of *distributed* smart camera networks over centralized smart monitoring systems are many - no costly central processors, less communication infrastructure and the ability to scale to practically any size. There are some challenges yet to be overcome in the development of smart camera networks, one of which is calibration. This involves determining the location and orientation of all cameras within the network, which is useful in tasks such as 3D reconstruction, and object localization and tracking. Ideally, the calibration process would be completely automatic and performed in minimal time.

A. Camera Calibration and SURF

To perform automatic calibration, a robust method of finding geometric relationships between the cameras' images is needed. A common way to perform stereo camera calibration is by matching common objects in each camera's field of view using *feature detectors and descriptors*. This method can also be used to calibrate multi-camera networks. One recently developed scale and rotation invariant feature detector and descriptor is SURF (short for "Speeded-Up Robust Features") [3]. It has proven to be as effective as similar well-known descriptors such as SIFT [15], while being much more computationally efficient [2], [3].

Computational efficiency is of particular importance in distributed smart camera networks, which typically employ low-power embedded processors. In addition, transmitted data must be kept to a minimum to conserve power and network availability. This means that detected feature data must be reduced and/or compressed before transmitting between cameras, while still maintaining enough information to accurately match image features.

In this paper, the effectiveness of an algorithm incorporating the SURF feature detector and descriptor in determining camera overlap is tested under varying levels of data reduction and compression. Live, low-quality images are provided by web cameras, to test the practicality of the algorithm when using low-cost image sensors. Camera overlap is found as a result of calculating image homographies, which can be used as a preliminary step in the camera calibration process. The algorithm is tested using two cameras, however is designed to work with a many-camera network. Analysis of such a network is left for future research.

This article is presented in the following order. Section II covers a brief review of the current literature on distributed smart camera calibration techniques. Section III provides background theory on camera overlap calculation using homographies, the SURF descriptor and its implementation in software. The complete camera overlap detection algorithm and its implementation is subsequently presented in section IV, with results following in section V. Section VI concludes the report, and includes suggestions of future work.

II. Related Work

Many methods are available to obtain camera calibration. A calibration object such as a flashing light, or an easily distinguishable object, when moved within view of multiple cameras can reveal camera positions via triangulation. This is done in [14], [17], [19].

Other sensor data such as GPS, laser ranging or wireless network signal strength are also used, and would be required in sparse networks of cameras whose fields of view do not overlap. This approach, called *sensor fusion*, is considered in [1], [16], [18], [20]. In this paper, however, a dense network of cameras with at least some overlapping fields of view is hypothesized.

Extra sensors and/or the use of a calibration object moved within the field of view of all cameras within the network aid in obtaining an accurate calibration, however they increase the complexity of the task in terms of equipment and potentially cost. Performing the calibration task using image data alone is a quick and cheap method, however relies on robust image processing techniques.

Devarajan et al. [6], [8], [9] developed a robust calibration method using image data alone. This was done by matching SIFT features of overlapping static images taken with a handheld digital camera. The SURF [3] feature detector used in this paper is able to perform the same task with similar results, using less CPU resources. Also, live webcam images are used instead of high-quality still images, due to their low cost and ease of interface via USB.

Chandrasekhar et al. [5] examined the effect of compressing SURF and SIFT descriptors using transform coding. They achieved near-perfect image matching performance at $16x$ reduction in SURF data size. Their focus was on image retrieval rather than camera calibration, however their approach would be ideal for a wireless camera network calibration task in which minimal data was to be transmitted across the network.

III. Background Theory

A. Calculating Image Overlap with Homographies

Camera overlap can be calculated by determining the homography between images of a planar surface viewed by two cameras.

A homography, otherwise known as a *planar projective transformation* [12], describes the relationship between two-dimensional image points viewed from different perspectives. The relationship between points on one plane P and a second plane P' is defined by a non-singular 3×3 matrix $\mathbf{H}$ (the homography matrix) such that:

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \tag{1}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are the point coordinates on the first and second plane.

Equation 1 was used in this paper to project the outline of one camera's field of view onto the other camera's image, thus providing an easy visual indicator of camera overlap. An obvious limitation of using homographies is that enough features on a planar surface must be detected for the homography calculation to work. This limits the number of scenes that this algorithm can be used on.

B. SURF

This section provides an outline of the SURF descriptor, which is useful in understanding the compression stage of the algorithm. For a full description of SURF, see [3]. The SURF feature descriptor is based on the proven method of mapping local contrast gradient orientation. The gradient in each grid segment is represented by the sums of first order Haar wavelet responses in the x and y directions.

A number of grid sizes can be used; larger grids increase computation time while increasing the distinctiveness of the descriptor. Bay et al. [3] found the best tradeoff in the 4×4 grid. Each grid element contains a four element vector describing the intensity pattern as shown in Figure 1:

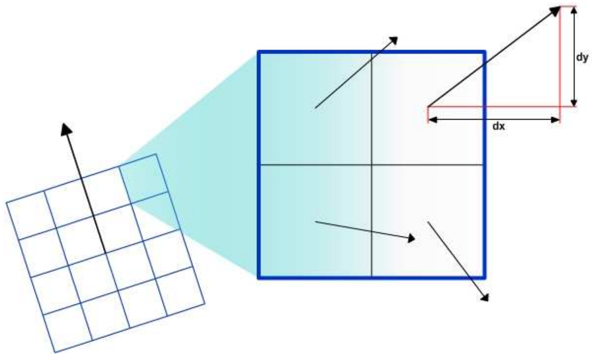


Fig. 1. The standard SURF-64 descriptor. The 4×4 grid is placed over each detected feature. Each grid element contains four sub-elements in which the contrast gradient vector is calculated. These vectors are then summed to produce the average gradient information for the grid element: $\sum dx$, $\sum|dx|$, $\sum dy$ and $\sum|dy|$

$$\sum dx, \sum|dx|, \sum dy, \sum|dy| = \sum\nolimits_{\mathbf{x}} \tag{2}$$

These vectors are stored sequentially in a larger vector describing the entire grid, resulting in a 64 dimensional descriptor vector for each feature (referred to as SURF-64):

$$\left[\sum\nolimits_{\mathbf{1}} \quad \sum\nolimits_{\mathbf{2}} \quad \cdots \quad \sum\nolimits_{\mathbf{16}}\right] = \mathbf{d_x} \tag{3}$$

To increase distinctiveness of the descriptors, an "extended" descriptor is available, which computes the sums of d_x and $|d_x|$ separately for $d_y < 0$ and $d_y \geq 0$, and similarly separates d_y and $|d_y|$. This doubles the size of the descriptor and takes only slightly longer to compute, but increases matching time significantly due to its larger size. The extended descriptor for the 4×4 grid, called SURF-128, was found in [3] to have the highest accuracy amongst all SURF descriptors.

IV. Description of Algorithm

The algorithm was developed for two cameras feeding images to a single computer, with the future intention of developing a many-camera network. The image processing functions used were provided in the Open Source Computer Vision (OpenCV) library.

The process followed for each frame captured by the first camera is described below.

A. Feature Extraction and Compression

Features and descriptors from camera one's images are extracted using the SURF feature extractor function provided in the OpenCV library.

Reducing the number of features allows faster computation in the descriptor extraction, compression, decompression and feature matching stages of the algorithm.

Spacing of feature points was not dealt with. Cheng et al. [6] used k-d trees to spread detected features, however this was overlooked for this paper due to time constraints.

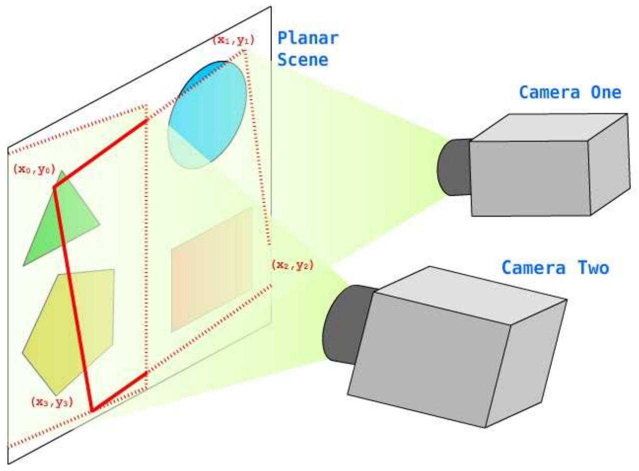


Fig. 2. Each camera's field of view is outlined as a dotted line. The algorithm determines the portion of camera one's field of view that can be seen by camera two, represented by the solid line. Camera one's field of view is described by four corner coordinates in camera two's image plane - (x_0, y_0) - (x_3, y_3)

To compress the calculated feature descriptors, descriptors $\mathbf{d_x}$ are arranged into a matrix $\mathbf{D}$:

$$\begin{bmatrix}\mathbf{d_1} & \mathbf{d_2} & \cdots & \mathbf{d_N}\end{bmatrix}^T = \mathbf{D} \tag{4}$$

which results in an $N \times 64$ (for SURF-64) or $N \times 128$ (for SURF-128) matrix, where N is the number of features. Singular value decomposition (SVD) is performed on $\mathbf{D}$ to find its principal components, of which an adjustable amount can be sent over a network connection. The approximated descriptor matrix $\mathbf{D'}$ at the receiving end is then constructed by back-substituting the SVD matrices. The level of compression is defined as the inverse of the number of principle components sent:

$$\text{compression}_{\text{(percent)}} = 100 \times (q - N_{\text{pca}})/q \tag{5}$$

where q is the descriptor size and N_{pca} is the number of principle components.

B. Feature Matching, Homography Calculation and Camera Overlap Estimation

Matching features are found using a simple naive nearest neighbors algorithm, which compares every descriptor against one another.

Calculation of the homography is done using Random Sample Consensus (RANSAC) [10] on the matching features, then using a non-linear algorithm to minimize reprojection error, as described in Chapter 4 of [12].

Figure 2 shows an example setup of the two cameras used for testing. The calculated homography is used to project the corners of camera one's field of view onto camera two's images, using equation (1).

Due to camera noise and erroneous feature matches, corner estimates vary from frame to frame. The magnitude of this variation increases as less features are extracted and descriptors are compressed. To suppress these errors, the corner estimates can be averaged over a number of frames. However, the homography calculation becomes less stable as the supplied data is reduced, and occasionally produces very large errors that dominate the average. To mitigate these effects, RANSAC is used on the corner estimates to reject such outliers.

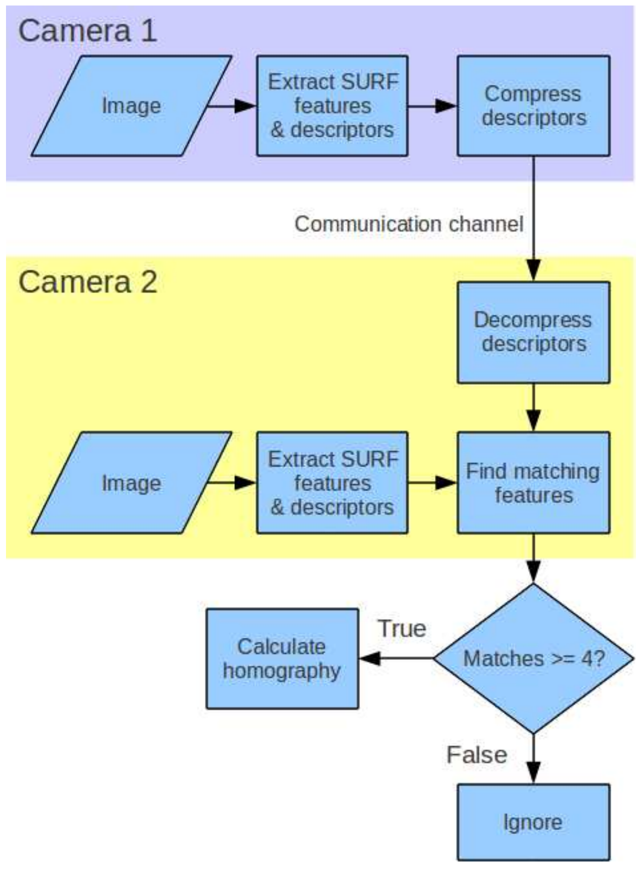


Fig. 3. Process followed for each frame captured by the cameras. Note that only the transmitted descriptors need to be compressed, in this case camera one's descriptors. Matching is performed at camera two. For testing of the algorithm, the compression and matching processes were performed by a single computer.

C. Accuracy Measurements

The accuracy of the projected field of view was measured by obtaining an estimate of the correct corners using large numbers of features, no compression, and RANSAC of corner estimates over many frames. These corners can then be compared with those calculated from smaller data sets. The initial estimate of the correct corners must be checked manually to ensure that they are reasonable. Although these "correct" corners will not be exactly correct, the aim was to compare the performance as data is limited, not to test the general accuracy of SURF.

The same method was used by Shafique et al. in [21], however they do not mention how they obtain the "true" homography. They call the error between the estimate and true homography *transfer error* e:

$$e = \frac{1}{n}\sum_{i}^{n} \|\mathbf{H}^{*}\mathbf{p_i} - \mathbf{H}\mathbf{p_i}\| \tag{6}$$

where $\mathbf{H}^*$ is the estimated homography, $\mathbf{H}$ the true homography and $\mathbf{p_i}$ are the points (corners in this case) in camera one's field of view.

D. Implementation

Testing of the algorithm was done using using a laptop computer with a 1.6 GHz Intel Pentium M processor, and two attached Logitech Quickcam E3500 web cameras. A linux OS (Ubuntu) was used, and the computer vision functions were provided in the OpenCV library.

The scenes used for testing comprised mainly of walls with distinctive patterns (posters/signs). Fourteen scenes in total were used for testing. For each scene, ten different compression levels as well as ten different extracted feature counts were used for each descriptor size.

The number of features was varied with a fixed compression level of 21.88%, since it was realized early on that slight compression of the descriptors seemed to produce more reliable results. The number of extracted features was set at 600 for the compression tests, to allow a reasonable chance of there being matching features present.

V. Results

The data shown in these results includes the corner estimates using simple averaging separate to the corner estimates using RANSAC.

The graphs of averaged corner estimates provide an insight into the scale of errors produced by SURF and the homography calculation as the features are reduced and descriptors compressed. As discussed in IV-B simple averaging produces much less accurate results than RANSAC. The graphs of errors using RANSAC show the results of the entire algorithm.

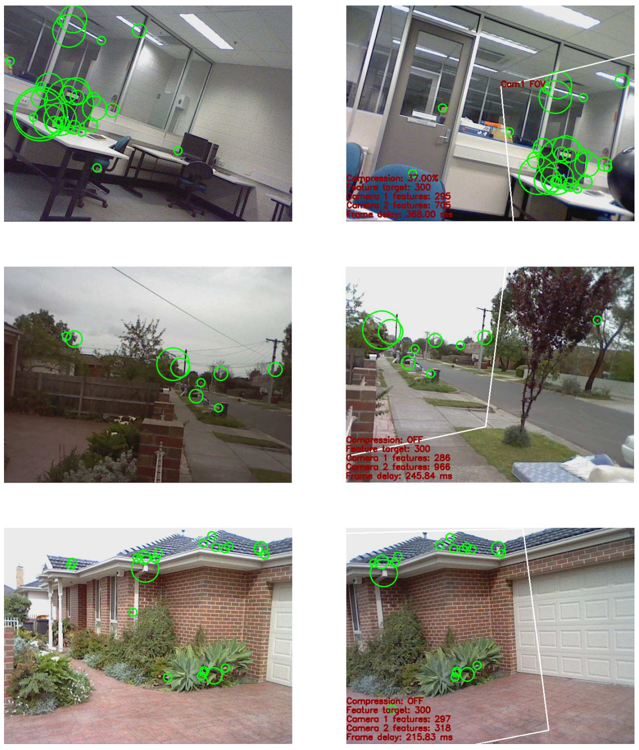
A. Effect of Descriptor Compression

As can be seen in Figure 5, average error is generally low until compression levels rise above 50%. Very large errors can occur even at low compression levels as discussed in IV-B, as shown by the large peak at 30% compression of the SURF-128 descriptor.

Ignoring the large peak of the SURF-128 descriptor, average error is slightly larger at zero compression compared the range of 10-50%. This is likely due to the use of SVD to compress the descriptors. SVD is useful in noise reduction, as shown in [11], and is also widely used in computer vision for tasks such as object recognition and feature reduction. Yan et al. show that the SIFT descriptor is enhanced via PCA [13]. Looking at figure 7, it can be seen that zero compression results in the most accurate estimates. It seems that SVD may reduce the probability of erroneous feature matches which reduce homography accuracy.

The corners estimated using RANSAC are reliable until compression levels of around 65% for SURF-64, and 75% for SURF-128, as shown in Figure 6. In some cases, accurate corners could be calculated at 85% compression, however such high levels of compression were not reliable.

SURF-128 was on average 23% more accurate than SURF-64 at reliable compression levels. This is at a cost of higher computation time, however the results show that SURF-128 can be more accurate than SURF-64 when compressed to a comparable data size (8). SURF-64 performs much better than SURF-128 as the number of extracted features is reduced (9). To reduce network traffic while still attaining maximum accuracy, SURF-128 could be used on a smaller number of features once an initial estimate has been calculated using



```
Compression: 37.00%
Feature target: 300
Camera 1 features: 295
Camera 2 features: 705
Frame delay: 368.00 ms
```

```
Compression: OFF
Feature target: 300
Camera 1 features: 286
Camera 2 features: 966
Frame delay: 245.84 ms
```

```
Compression: OFF
Feature target: 300
Camera 1 features: 297
Camera 2 features: 318
Frame delay: 215.83 ms
```

Fig. 4. Screenshot of the tested software. The field of view (FOV) of camera one is plotted as a white outline in camera two. Matching features are shown as green circles. Note that erroneous feature matches, such as those outside camera one's FOV, are detected and disregarded during the homography calculation.

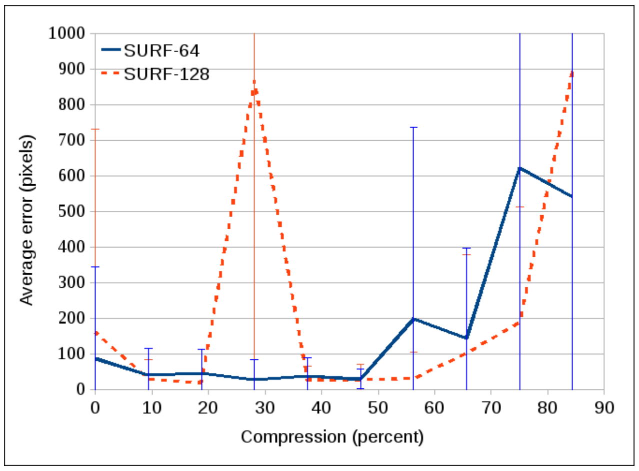


Fig. 5. FOV corner errors using simple averaging versus descriptor compression. The large peak at 30% compression of SURF-128 is due to a large error caused by the homography algorithm converging on an erroneous solution, which is possible even with no compression.

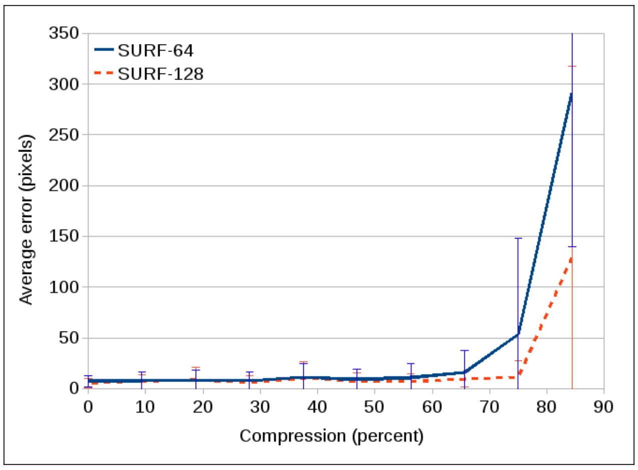


Fig. 6. FOV corner errors using RANSAC versus descriptor compression. Large inconsistent values produced by the homography calculation are rejected, resulting in much more accurate corner estimates.

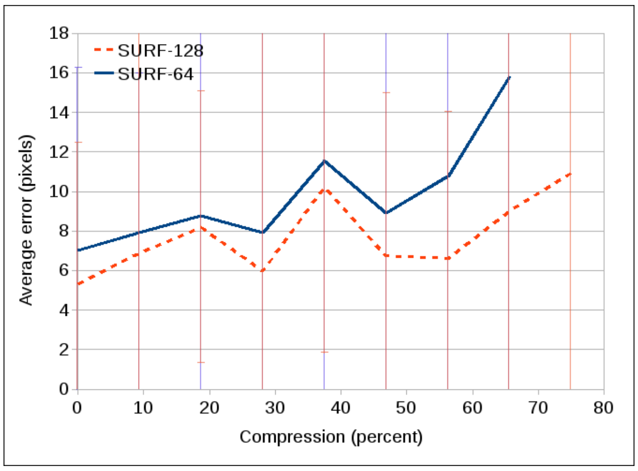


Fig. 7. A closer look at figure 6. SURF-128 proves to be 25 - 40% more accurate than SURF-64 at reliable compression levels.

SURF-64.

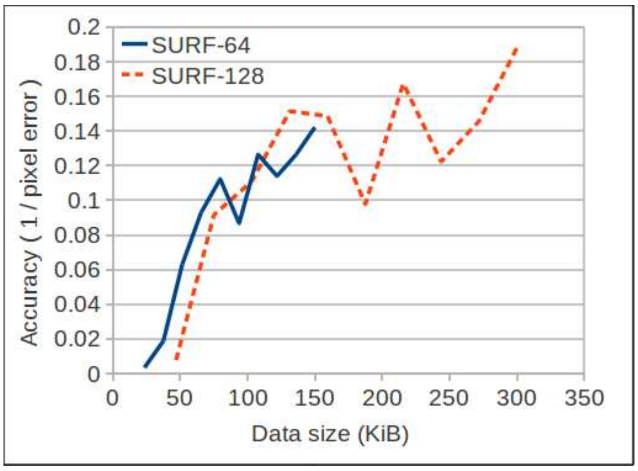


Fig. 8. Plot of accuracy (1/pixel error) versus data size. Data size is varied by compression.

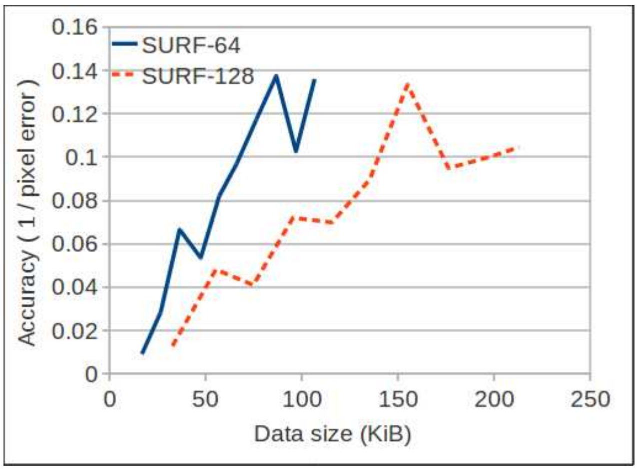


Fig. 9. Plot of accuracy (1/pixel error) versus data size. Data size is varied by adjusting the number of features used.

Compression level had no effect on computation time, due to the simplicity of the compression process. SVD is computationally expensive, especially for large numbers of features. Smarter compression processes may increase the computation efficiency of the overall algorithm, however this is left for future research. Chandrasekhar et al. [5] used PCA to reduce the dimensionality of SURF descriptors in a similar way, however their focus was not on computational efficiency during the compression stage.

B. *Effect of Feature Reduction*

As the number of extracted features was reduced, the accuracy of the algorithm varied greatly from scene to scene, as can be seen by the size of the error bars in Figures 10 and 11. It is likely that this is due to the lack of spacing of extracted features, causing clustering around distinct objects. The scenes used varied in the number of distinct objects on planar surfaces, meaning the homography calculation benefited from more planar point matches in some scenes and not others. Also, the reduction of feature points was achieved by increasing the contrast gradient threshold. Although features with higher contrast gradients may be easier to detect from different viewpoints, their descriptors may not necessarily match. This would result in fewer matching features, making the homography calculation less accurate.

Using the tested scenes, errors of less than ten pixels required 350 or more extracted features. In their use of k-d trees to separate features, Cheng et al. [6] found that as few as 78 features was enough to perform reliable and accurate camera calibration.

Computation time increased exponentially with increasing extracted features as expected, shown in Figure 12. This is due to SVD used in compression, since it has $O(m \times n^2)$ complexity.

SURF-64 and SURF-128 produced similar error responses to a reduction in extracted features, as can be seen in Figures 10 and 11. Therefore, SURF-64 should be used for faster computation. Errors of less than ten pixels were produced by

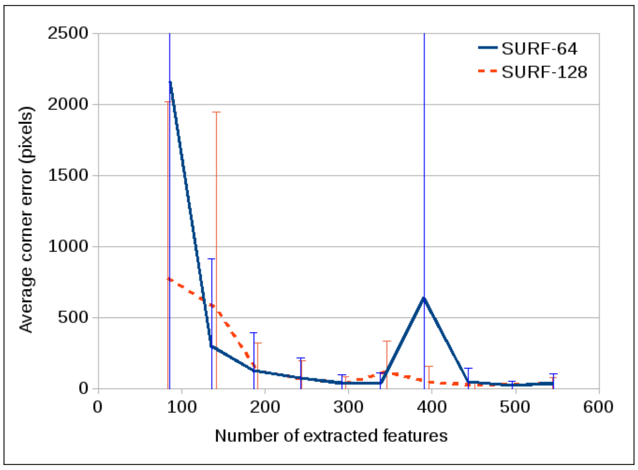


Fig. 10. FOV corner error using simple averaging versus the number of extracted features.

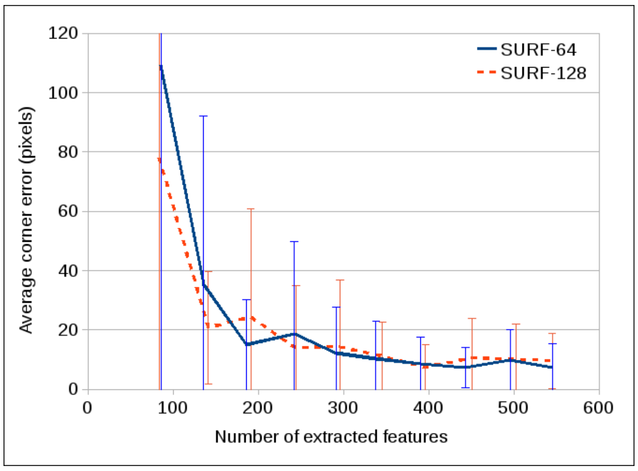


Fig. 11. FOV corner error using RANSAC versus the number of extracted features. Note the large reduction in error due to the use of RANSAC.

using 350 or more features, however the accuracy is heavily scene-dependent as discussed earlier.

C. *Minimum Reliable Data Size*

Although constraints on network traffic will vary between networks, it is desirable to send as little data as possible to calculate image overlap. The results of this paper show that SURF-64 can achieve reliable overlap calculation using less transmitted data than SURF-128, however SURF-128 can achieve higher overall accuracy.

The OpenCV implementation of SURF stores feature information such as orientation and scale in a 24-byte structure for each feature. Descriptor vector elements are of 4-byte floating point type, resulting in 256 byte descriptors for SURF-64 and 512 bytes for SURF-128.

Using SURF-64, for each set of N extracted feature points, the uncompressed data for network transmission is:

- Number of features, sent as integer - 4 bytes
- Feature information - $N \times 24$ bytes
- Descriptors - $N \times 256$ bytes

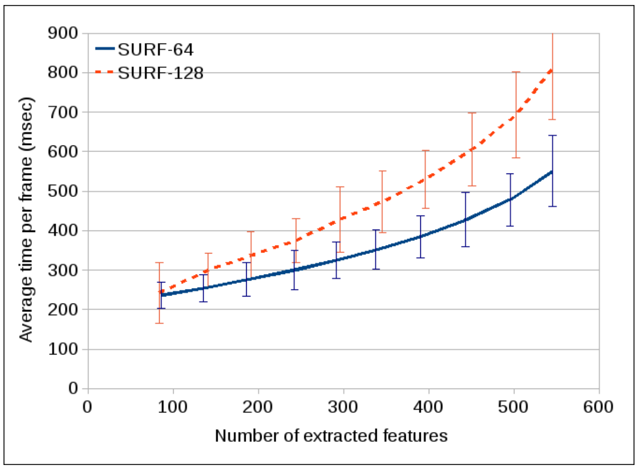


Fig. 12. Number of extracted features versus computation time per frame. Variability in computation time is due to time measurement being taken over the complete frame cycle, thus including operating system overhead, camera frame capture time variability etc.

Using SURF-64 descriptors, a compression level of 65% was deemed the maximum reliable compression level, after which errors in excess of 50 pixels were generated. Using 200-300 features and 65% compression produces a data size of 22 - 34kB per frame, with expected errors of around 16 pixels. Accuracy was nearly doubled by decreasing compression to 50% (31 - 49kB per frame).

SURF-128 at a compression level of 28% reduced pixel error to 6, while producing 78 - 118kB per frame. An adaptive method could improve accuracy while maintaining lower transmitted data by obtaining an initial estimate of the homography using SURF-64, then reducing error by transmitting small numbers of features with SURF-128 descriptors at lower compression.

VI. Conclusions and Future Work

In this paper, an algorithm to calculate image overlap in a distributed smart camera network using minimal data transmission and low quality image sensors was described and evaluated. For this purpose, it was found that the SURF-128 descriptor achieved higher accuracy than SURF-64, at the cost of computational efficiency and larger data sizes. SURF-64 produced reliable estimates with the minimum transmitted data size, estimating overlap to within 9 pixels of the true overlap while transmitting 30kB per captured image.

Results of the algorithm testing suggested that moderate levels of compression using SVD reduced the likelihood of erroneous matches of SURF features, however this was not confirmed.

We are currently focusing on the following areas for further algorithm improvements:

- More efficient means of compression
- Feature distribution to increase the probability of detecting overlap. Cheng et al. [6] achieved this through the use of k-d trees, while Brown et al. [4] developed a technique called “adaptive non-maximal suppression”.

- Accuracy of the homography. An iterative technique to improve the homography in the presence of image noise is presented in [7].
- Implementation of the algorithm in a many-camera network

References

[1] A. Barton-Sweeney, D. Lymberopoulos, and A. Sawides. Sensor localization and camera calibration in distributed camera sensor networks. In *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on*, pages 1–10, 2006.

[2] J Bauer, N Sunderhauf, and P Protzel. Comparing several implementations of two recently published feature detectors. *Proceedings of the International Conference on Intelligent and Autonomous Systems, IAV, Toulouse, France*, 2007.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. SURF: speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

[4] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 510–517 vol. 1, 2005.

[5] Vijay Chandrasekhar, Gabriel Takacs, David Chen, Sam S. Tsai, Jatinder Singh, and Bernd Girod. Transform coding of image feature descriptors. volume 7257, page 725710. SPIE, 2009.

[6] Zhaolin Cheng, Dhanya Devarajan, and Richard J. Radke. Determining vision graphs for distributed camera networks using feature digests. *EURASIP Journal on Advances in Signal Processing*, 2007:11, 2007. Article ID 57034.

[7] Zhou Chuan, Tan Da Long, Zhu Feng, and Dong Zai Li. A planar homography estimation method for camera calibration. In *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium on*, volume 1, pages 424–429 vol.1, 2003.

[8] D. Devarajan, Zhaolin Cheng, and R.J. Radke. Calibrating distributed camera networks. *Proceedings of the IEEE*, 96(10):1625–1639, 2008.

[9] Dhanya Devarajan and R.J. Radke. Calibrating distributed camera networks using belief propagation. *EURASIP Journal on Advances in Signal Processing*, 2007.

[10] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[11] P.C. Hansen and S.H. Jensen. FIR filter representations of reduced-rank noise reduction. *Signal Processing, IEEE Transactions on*, 46(6):1737–1741, 1998.

[12] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[13] Yan Ke and Sukthankar R. Pca-sift: A more distinctive representation for local image descriptors. volume 2, pages 506 – 513, 2004.

[14] G. Kurillo, Zeyu Li, and R. Bajcsy. Wide-area external multi-camera calibration using vision graphs and virtual calibration object. In *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pages 1–9, 2008.

[15] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, pages 1150–1157, 1999.

[16] L. Marchesotti, S. Piva, and C. Regazzoni. An agent-based approach for tracking people in indoor complex environments. In *Image Analysis and Processing, 2003.Proceedings. 12th International Conference on*, pages 99–102, 2003.

[17] V. Mehta and Weihua Sheng. Distributed calibration of a camera sensor network. In *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pages 1974–1979, 2009.

[18] T. Miyaki, T. Yamasaki, and K. Aizawa. Multi-Sensor fusion tracking using visual information and WI-Fl location estimation. In *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, pages 275–282, 2007.

[19] L. Lo Presti and M. La Cascia. Real-time estimation of geometrical transformation between views in distributed smart-cameras systems. In *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pages 1–8, 2008.

[20] F. Schweiger, I. Bauermann, and E. Steinbach. Joint calibration of a camera triplet and a laser rangefinder. In *Multimedia and Expo, 2008 IEEE International Conference on*, pages 1201–1204, 2008.

[21] K. Shafique, A. Hakeem, O. Javed, and N. Haering. Self calibrating visual sensor networks. In *Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on*, pages 1–6, 2008.