Bayer-Based Vision System: Image Processing in an Incomplete Color-Space

A. Haghshenas[1], M. Ahmad SharBafi[2], D. Esmaeeli[2], and O. Bakhshandeh[2]
[1]Computer Engineering Department, Iran University of Science and Technology, Tehran, Tehran, Iran
[2]Electrical and Computer Engineering, Qazvin I Azad University, Qazvin, Qazvin, Iran

Abstract - *While most imaging devices use the Bayer pattern, virtually all vision methods assume the availability of information for a complete color space. This creates the need for demosaicing step, for which even the fastest naïve interpolation is in some applications a computational burden. In this paper, a complete real-time vision infrastructure is built directly upon the Bayer pattern without interpolation. The specific application area is the vision subsystem for a small-sized soccer robots team, successfully attending world final games. However, the techniques are potentially applicable to other problem areas as well.*

Keywords: Bayer Pattern, Demosaicing, Image Processing

1 Introduction

With the exception of expensive 3ccd devices, all ordinary color cameras in the industry use the Bayer pattern or a similar under-sampled color matrix. In the Bayer pattern, photo sensors are arranged such that each 2x2 square of cells contains two green sensors, one red and one blue. This implies that the sensor only gives one of the three color components for any designated location of the image and the other two are missing. A "demosaicing" process is required to create a complete image. It may appear that a simple interpolation will create good results, at least when the pixel dimensions are much finer than image details. However, close to the border of abrupt changes, false color artifacts will appear by this method, as a result of getting one component from the sensor that falls in "dark" area and averaging the other components that may come from the "bright" area or both areas. Such color artifacts are usually noticeable, even in high resolutions. A good demosaicing algorithm is one that:

- Minimizes the color artifacts near edges
- Minimizes the loss of spatial resolution
- Has low sensitivity to noise
- Has low computational complexity (Especially when hardware implementation is necessary for portable devices)

Several methods have been proposed in the literature, with different trade-offs among these requirements, a good survey of which can be found in [1]. A number of methods are proposed to jointly address the demosaicing problem along with other low level image processing problems, for instance in [2] the super resolution problem and demosaicing are performed in a single processing framework. In [3] a method is proposed to de-blur the image while performing demosaicing. Image tamper detection method is proposed based on verifying artifacts from demosaicing step in [4]. Joint zooming and demosaicing is proposed in [5] as well as other sources.

Most high frame-rate industrial color cameras with Bayer filter, can output only the raw Bayer format when operating at their top frame-rate. Even by sacrificing frame-rate at the expense of internal demosaicing, the output RGB is distorted (from the ground truth or from the output of high-end computationally intensive algorithms) with many types of artifacts. Sometimes algorithms designed and tested with images acquired from conventional digital cameras can break, malfunction or degrade performance when facing real images with such artifacts. An example of such artifacts from output of a real high end camera is displayed in Fig 1.

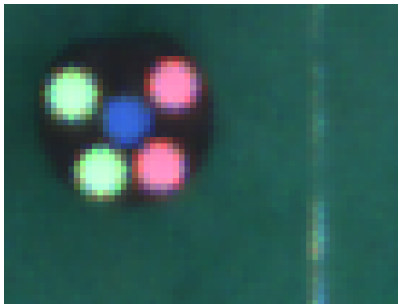

Fig 1:The image acquired from a robot and a white band on a green carpet. The Color artifacts are a result of naïve demosaicing process, not physical phenomena such a surface reflection.

2 A generic computer vision stack

In order to demonstrate how demosaicing step can be avoided (sometimes not altogether), we use a generic

computer vision stack. Though it does not cover all image processing tasks, sufficient insight into the technique can be gained through it. In a typical computer vision algorithm, one first separates the "interesting" parts of the acquired image, a process which is called "target detection", "background separation" and "region of interest selection" in different contexts. The next step is to group the pixels that should be evaluated together, to form blobs. In the evaluation phase, each group is represented with a few quantities, calculated over the entire blob. Finally, the results of the evaluation phase are combined to obtain a high level understanding of the scene. It should be double emphasized that the failure to model a certain algorithm with this stack does not have any implication about the applicability of the proposed methods. The stack is symbolized in Fig 2.

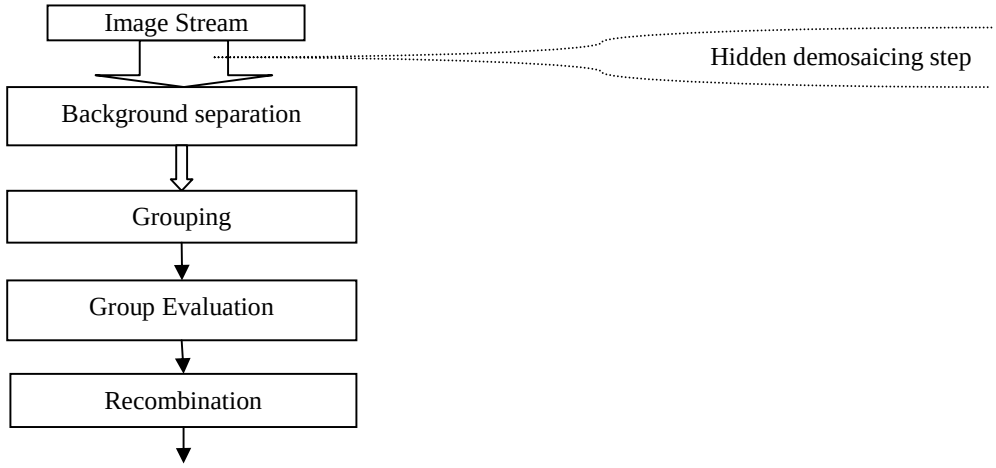

Fig 2: An illustration of a generic vision stack. The widths of arrows symbolize the data bandwidth between two successive steps.

3 Our computer vision task

The proposed method was first applied in the following vision context: In Robocup competitions, SSL league, two teams of up to five robots are to play a soccer match. Vision cameras are mounted on top of the field and interesting moving objects are an orange golf ball and the robots which use color markers for identification purposes. Frame rates of up to 60 are common and given the current fast-paced nature of the competition, high frame rate is an advantage or even a necessity. Vision delay is also of critical importance and several methods are employed to compensate for it [6].

A traditional approach to the vision subsystem can be summarized as follows: amongst the incoming pixel data, use color thresholding to find interesting colors and use neighborhood data to create blobs. After finding the centers of blobs, close blobs are merged together into robots position and head direction is estimated based on knowledge from color patters.

In our vision implementation for the 2009 games, we planned to use overlapping cameras, for a better solution to 3d vision than previous tracking approaches. This meant that more cameras had to be processed with a single computer. Other problems such as problem with false color artifacts emphasized the need to remove the demosaicing step.

4 Removing or postponing Demosaicing

As can be observed from Fig 2, a demosaicing step exists (usually hidden) just before the first step of the image processing, over the largest data bandwidth. If this step can be eliminated, or at least postponed to a higher processing level, some or all of the following benefits can be expected:

- Better performance in the first phase: the first phase just throws away uninteresting parts of the image and in many systems is the only module that handles the whole picture area. If the demosaicing step is shifted after this phase, the size of the data to be processed will decrease by a factor of 3. This can imply other meanings such as better cash-memory usage and elimination of (usually floating point) color space conversion tasks.

- Better performance in the demosaicing step: If the demosaicing step follows the background separation instead of preceding it, the interpolation algorithm has to be executed only over the selected pixels. This can mean huge performance gains if only small percents of the image area pass the first step. Another possible implication is that this decrease in data size can make high-quality demosaicing methods feasible.

In our problem of interest, the camera could be set to output lower intensity images such that almost all parts of the background become very dark in all three color components.

A simple thresholding on the Bayer data was therefore sufficient to remove “background” pixels and obtain only the pixels belonging to color markers or the ball.

The grouping phase can also be performed before the demosaicing step. This is possible if the membership condition on each group can be determined from a single color component (plus the trivial location and neighborhood information). We defined the membership condition as follows: pixels from similar color component, falling above a certain threshold and connected to each other through a modified neighborhood. The simple 4 or 8 connectivity neighborhoods cannot be used in this context as not all adjacent pixels to any pixel are of the same color component. The neighborhood connectivity for each color component is displayed in Fig 3.

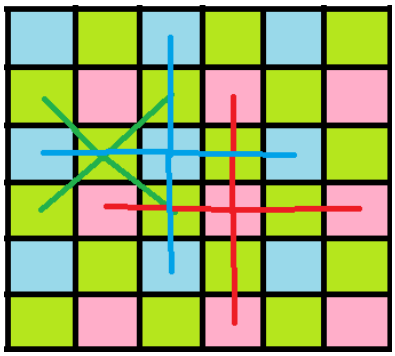

Fig 3: Neighborhood connectivity is shown for each color component. Please observe how red and blue components should use a different neighborhood from green.

This condition on blob creation has the following problem: Composite color markers like yellow, will create two or three overlapping blobs, one for each color component above the threshold. This problem is easily addressed by merging all blobs that are closer to each other than a certain threshold (like 1.5 pixels). For the position of each blob we use the weighted average of x and y components of its pixels, weighted with how much above the threshold each pixel was.

Sometimes, a more subtle problem arises that is illustrated in Fig 4A. The two closer blobs in the back of the robot both have green values above the threshold. Also, the green values are connected and a large blob is formed. The center of the large blob does not coincide with any other blob and is not merged.

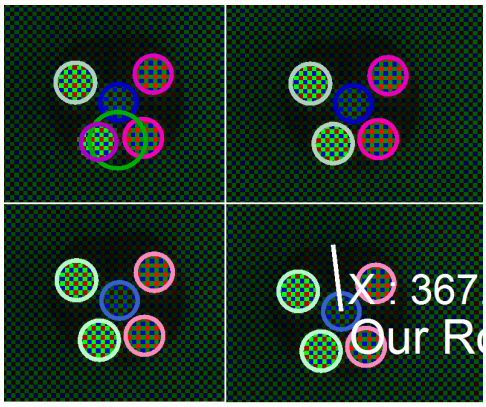

Fig 4: The large blobs, formed by falsely connecting green components of two very close blobs

To solve this problem we had to “split” the large blob into its two contributors. This we did using the famous K-means algorithm, again weighted with how much each pixel is above the threshold. The corrected result is shown in Fig 4B.

For each blob we keep its size, which the sum of weights of its pixels, its X and Y center, and its color component(s). Before the blob merging step, each blob belongs only to one color component. When merging two blobs of different color component, a new blob is formed, whose size is equal to the sum of its contributing blobs and X coordinate is calculated from X=(X_1 W_1+X_2 W_2)/(W_1+W_2). The new blob is marked as having both color components.

The next step will be to assign an aggregate color to each blob. For this purpose we used the color of all pixels that fall strictly inside the blob area. Pixels from each color are summed up separately and averaged. Using this method, a very noise tolerant color estimation is produced for each blob. The reason is that the color is calculated using many pixels instead of for each separate pixel. The precise color estimate for blobs is displayed in Fig 4C.

One fine point is in calculating the blob radius: the radius not only depends on how many pixels belong to each blob, but also on how many and what color components contribute to the blob. Consider a blob of only blue pixels. If the blob radius is R, its area is πR^2 and overlaps πR^2 weighted pixels; but only one out of each 4 pixels are blue, so the number of weighted pixels belonging to this blobs is (πR^2)/4. So before calculating the blob radius, we need to multiply blob size with ¼ for pure blue and red blobs, ½ for pure green blobs and for red-blue blobs, ¾ for blobs with two colors one of which is green and 1 if all color components contribute to the blob.

From here on, we have all the information one would require to identify object on the field. This way, the process of “interpolating” as completer color value for each pixel is reduced to “aggregating” a complete color average over each

blob which is considerably faster (as the number of blobs is much less than the number of pixels) and at the same time very precise.

Other than the two previously mentioned benefits, by moving the demosaicing step to after the grouping or totally removing demosaicing, we expect the following advantages:

- High stability and noise tolerance in quantifying each region.

- Avoiding unintended usage of wrong information. For instance, when a decision is made based of the green color component of a pixel, the actual green value may be a bad estimate from neighboring pixels. This can reposition the borders of a region. In our example, it can be shown theoretically that using the proposed method the expected error in blob center calculation is less than the traditional interpolated method. Actually, measuring this quantity is very difficult and we have no empirical results to back it.

- Avoiding the loss of spatial resolution as a result of blurring inherent to many fast demosaicing methods. Actually, in some application areas a very low cost solution to false color artifacts is sometimes to keep the camera lens slightly out of focus. This will blur sharp edges and decrease many artifacts. The same resolution loss is observed with low-end algorithms.

Using the proposed method, we were able to process 4 cameras with a single computer at 74 frames per second for each camera which was the native frame rate of the cameras. On the same computer a complete RGB implementation reached 15 frames for each camera, slightly less than what we expected from a factor 3 increase in data size plus demosaicing time. The whole vision framework was reliable enough to play successful games in the Robocup 2009 world finals in Austria.

5 Conclusions

In certain scenarios, it may be possible to eliminate the demosaicing process, or at least postpone it to a higher processing level, to engage it with much less data. The possible benefits range from improved performance to more reliable processing results. While the proposed method is not applicable to wide range of general purpose vision problems, in some cases, a smart implementation can incur performance, reliability and/or quality improvements.

6 References

[1] Xin Lia, Bahadir Gunturkb and Lei Zhang. “Image Demosaicing: A Systematic Survey”; Proc. IS&T/SPIE Conf. on Visual Communication and Image Processing, Vol. 6822, (68221J-68221J-15), (Jan 2008).

[2] Sina Farsiua, Michael Eladb and Peyman Milanfara. “Multi-Frame Demosaicing and Super-Resolution from Under-Sampled Color Images”; Proc. IS&T/SPIE Symposium on Electronic Imaging, Vol. 5299, (222-233), (Jan 2004).

[3] Dmytro Paliy, Alessandro Foi, Radu Bilcu, Vladimir Katkovnik, and Karen Egiazarian. “Joint Deblurring and Demosaicing of Poissonian Bayer-Data Based on Local Adaptivity”; Proc. European Signal Processing Conference EUSIPCO, (Aug 2008)

[4] Ahmet Emir Dirik and Nasir Memon. “Image Tamper Detection Based on Demosaicing Artifacts”; IEEE International Conference on Image Processing ICIP, (1497-1500), (Nov 2009).

[5] Kuo-Liang Chung, Wei-Jen Yang, Pang-Yen Chen, Wen-Ming Yan, Chiou-Shann Fuh. “New Joint Demosaicing and Zooming Algorithm for Color Filter Array”; IEEE Transactions on Consumer Electronics, Vol. 55 Issue 3 (1477 – 1486) (Aug 2009).

[6] Laue, Tim, et al. B-Smart, Extended Team Description for RoboCup 2009. s.l. : Robocup, 2009.