Analysis of different color recognition methods for active markers in a motion capture system

Przemysław Kowalski
Kipertech Consulting,
ul. Szwedzka 52,
30-315, Kraków, Poland

przemyslaw.kowalski@kipertech.com

Jan Mrzygłód
Kipertech Consulting,
ul. Szwedzka 52,
30-315, Kraków, Poland

jan.mrzyglod@kipertech.com

ABSTRACT

The article focuses on a method for reliably identify moving colored artificial markers in real-time. The marker was used to determine the 3D position in the space of the user(s).

The goal was to ensure that points were found and identified predictably and reliably by many cameras simultaneously, which, with appropriate calibration, merging, and processing of the data, could provide reliable information about the current 3D position of a given point in real-time. This information was crucial to other components of the broader vision system (VR platform).

The problems encountered and the remedial methods discussed in the presentation concern several aspects that we encountered during research, such as changes in lighting conditions, the quality (and stability) of the generated light and color, the dependence of color recognition on the distance of the light source from the camera matrix, aspects of light reflections, and many others. During our research, we analyzed various RGB/RGBW LED light sources from different manufacturers, which are characterized by different light generation characteristics. We also used a light diffuser. Using different sets of cameras and lighting conditions, we conducted several studies and experiments.

During the research, we managed to find basic colors for our marker-tracking visual system that met the goals. We have proposed an algorithm to deal with the problem and demonstrate the reliability of the visual layout with the algorithm. During our research, we used both conventional and alternative techniques related to ML.

Keywords

Color Space, Color Detection, Marker Detection, Image Processing, Computer Vision, Virtual Reality, Machine Learning, Real Time

1. INTRODUCTION

Our goal was to track users in real-time and visualize their position in VR. For our system Virtual Entertainment Enhanced Platform (VEEP), we proposed a vision-based motion capture system [9]. The system uses cameras to track and identify artificial active markers in 2D (application: tracker) and an additional module to calculate 3D positions from the positions reported by cameras (application: coupler; fig. 1). Systems that combine marker and motion tracking are a popular solution [10, 16]. In our application, we wanted to combine the real position with the virtual one so that the user feels the space naturally at any time.

Markers are placed on the player's head, and the aim of the system is to locate the player so that players can be located in real time. The players are wearing VR head-mounted display (HMD); the system should ensure safe and comfortable movement for the players, where their sensors are responsible for the orientation of the HMD and our vision system is responsible for the location, i.e., rendering real movement. Players thus move freely in space; their position is transmitted from the system to the HMD (Occulus Go or those based on Samsung Android phones) via a wireless network.

There are two reasons for placing the marker on the head: to reduce the obstacles between the camera and the marker, and to easily connect the position with the position of the player from which he or she is observing the world.

Such assumptions lead to requirements related to reliability and efficiency (working in real-time at not less than 60 fps, preferably 90 fps [13] or more). Our test experience confirms these requirements: delivering positions at a frequency of

approx. 30–40 fps induced poor player experience (VR sickness). The unpleasant feeling could be caused by the erratic data delivery itself (we were forced to ensure that the real-time requirements were met by using the Xenomai library on Linux computers (i5-6400T CPU 2.20GHz, 16GB RAM). We replaced the standard v412 Linux interface by communicating directly with the cameras using the LIBUSB library, and we experimentally selected a WiFi router with a 5GHz band to reduce data latency for an Android phone).

Image processing is simplified – pixels exceeding the threshold value are searched for (unless they have been used before), the neighboring area (with a lower threshold and an additional condition of maximum color change) is then filled in as being used, and the pixels are counted to determine the average color (RGB) and center of gravity (thus achieving sub-pixel accuracy of the marker indication). In extreme cases a distant marker may be represented by only 2 pixels.

The coupler calculates the 3D position from the submitted vectors (which originate at the focal point of the camera and indicate the marker in 3D). The procedure uses the method of calculating the nearest point to a given straight line, on a second straight line. For a pair of straights, it is performed twice and the two resulting points are compared. If their distance is within acceptable accuracy (as determined by the calibration quality of the cameras), an average is calculated. We apply the same principle to multiple pairs of cameras – averaging positions as long as they are within an acceptable distance and discarding them when they exceed this value.

Delays or instability also create a lack of confidence in the system among users who do not feel free to move around the room.

Experience gained during the construction of the system has shown that speed (low maximum delay) and certainty of location are decisive.

The system also requires tremendous reliability, assuming 99% correct marker identifications (color detection and recognition) per frame for a camera, At 120FPS and eight cameras (current configuration), this means 9.6 errors per second. We can't allow a single error in about twenty minutes of gameplay. Of course, this is a higher level of reliability than the described algorithm provides; we use numerous methods to verify and correct the result during further processing, but our primary goal is the speed and reliability of marker recognition, which has led to many simplifying assumptions: using active markers, darkening the room, or limiting the distinguishable features of the marker.

We can track markers through time using the distance between consecutive positions, but such a solution is prone to error because of the higher possibility of calculating and tracking errors.

Using the characteristics of markers can reduce calculation time and be less error-prone. Because we chose to work with (various) cameras working in the visible light range, the obvious characteristic of the marker was its color (we also considered the frequency), but it leads to two problems: we need to recognize the color of the marker with the changing light conditions, and we should distinguish the colors. In other words, our system should always be able to identify a color.

In our system, we use a dark room and active markers. We adopted such simplifications due to the requirement of fast (lasting a few milliseconds) and reliable identification of markers in the image. In practical implementations, "darkness" differs depending on the methods used to block external light sources and is practically never complete; it should rather be understood as a significantly limiting amount of light in a room than its complete elimination. The light conditions still differ from acquisition to acquisition, although the differences are reduced.

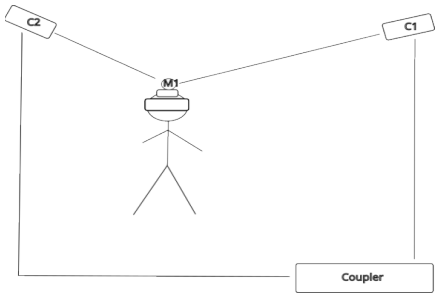


Figure 1. VEEP-system idea. Two (at least) cameras with trackers (C1 & C2) have a localized marker (M1) worn by a user. Vectors (C1→M1; C2→M1) are sent to the coupler. The coupler calculates user position, which is sent to the VR set (HMD) worn by the user.

Theoretically, the task should be simple: the combination of a color invariant and an active marker should result in a constant color.

In practice, the measured color of the marker changes. There are several reasons: the design of the marker itself (slight surface irregularities, quality of the diodes used, power supply), changes in lighting conditions, differences in color recognition by cameras, noise, and the and the small size of the marker expressed in the number of pixels (a large percentage of pixels partially representing the marker, whose color corresponds

to the combination of marker color and background color).

Cameras

During the development our solution, we faced the problem of choosing cheap, fast, and reliable cameras. We have tested different models of cameras, and we chose the PlayStation PS4 cameras, which are significantly cheaper than their counterparts and equipped with a lens that produces very little distortion (although we also tested other solutions, and the system allows us to work with a heterogeneous set of cameras). While working with the cameras, we also faced a choice of operating mode—whether we preferred resolution or speed. In the practice of our issue—providing reliable positioning to a player moving with HMD — operating time proved critical. By choosing to run at 120FPS, we reduced the resolution requirements, which reflected positively on computation time and negatively on image resolution (and color recognition quality).

Active Markers

The active markers (Fig. 2) were developed as a programmed embedded system with color LEDs and a silicone sphere that uniformly diffuses the light. We have used different kinds of LEDs; we focused on two products, one of which is SMD (Surface Mounted Diode) and the other is THT (Through Hole Technology), and we were able to set one of the 360 colors (colors are set by using RGB LED combinations with filling; there is one color per degree of arc on the color circle). For simplicity, the results for both types of diodes will be described using the assembly method (THT, SMD). To control the LED, we use the microcontroller's PWM module, which generates a signal at 490 Hz.

Figure 2. Active marker connected to VR head-mounted display.

We have prepared a test to see how the 360 programmed colors are visible by the camera; we displayed successive colors in a loop (the beginning is the same as the end). In our example, a PS4 camera was used.

The results show Figs. 3 and 4. We can see that not all changes in LED light are visible for the camera (for both types of markers). It means the real number of colors to find is much smaller than 360.

To propose which colors to use, we have analyzed the RGB (in fact, normalized RGB) distance between neighboring colors with different steps, which suggests basically six colors to use: red, green, blue, cyan, yellow, and magenta. Such colors give us the maximal distances, i.e., they are the easiest to detect.

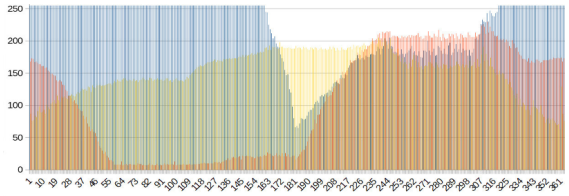


Figure 3. RGB components for SDM LEDs. (The blue bars correspond to the blue component, the yellow bars to the green component and the red bars to the red component.)

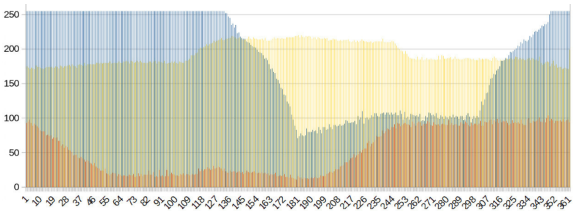


Figure 4. RGB components for THT LEDs. (Color designation – as in fig. 3.)

Search for solution

Basically – cameras return RGB images. The RGB image does not give us color-invariant components. The simplest solution is normalized RGB, where the red, green, and blue values are normalized by the pixel brightness. We have tested the normalized RGB, but the analysis of the output had to contain rules for at least three components and tolerate "gray" pixels (too light or too dark to represent marker color properly). The system worked with such a solution but was sensitive to light condition changes.

The color identification algorithm consists of image transformation into a chosen color space and a color classification method. The main color spaces used are: normalized RGB [2], HSV [3], YCbCr, LAB [4], RGB-L*a*b* [12], MCSS [5]. We have tried to use other color spaces: YUV (with two chrominance components: U and V), YCbCr [14],

LAB, and HSV [1, 6, 7, 8, 11, 14, 15]. The best results (i.e. easiest color detection) were given to us by HSV, and we have focused on this color space.

Tests indicated that practically the best results, taking into account differences in distance and marker reproduction quality, are given by maximizing the spacing between colors in the H (hue) component.

Practical problems

Our active markers were developed to maximize color uniformity on the surface, but in fact, they are not uniform. It may be justified by the limitations shown in Figs. 3, 4, and 5 (and Fig. 5 for a marker seen by a camera): the color is distorted by too-light areas of the marker. We used the average color that reduces the observed distortion.

The other problems are connected with camera noise—the values of the pixel was changing from frame to frame.

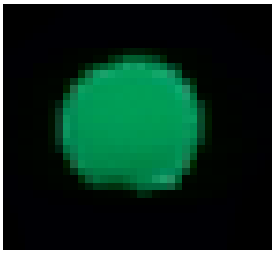


Figure 5. Example image of a marker captured by the camera at medium distance. (The image quality matches that of the camera, and visibility of the marker in the darkened room.)

Additionally, there was a problem with lightning conditions. We assume the system works in a dark room where the light sources are turned off (for artificial) or occluded (for natural). Additional sources of light are markers themselves; if many markers are used in the room, the whole room can be slightly illuminated. We keep the high contrast between the marker and the background, but the brightness of the marker (and its color) changes.

There is a similar problem with the marker's power supply; a discharged battery first changes the brightness of the marker, which may previously have affected the camera's perception or been perceived by a human being.

The basic research was done in the office, with limited distances between markers and cameras. We have overlooked the scale problem caused by the color of small marker images, i.e., a situation with the distant markers when we increase the size of the scene. The problem was even bigger because we have reduced image resolution (from 1280×800 to 640×400), which was caused by an increase in camera speed (from 60 fps to 120 fps), which is essential for VR immersion. The marker surface becomes less than 20 pixels at a distance of about 5 meters. With the decrease in marker size, the quality of color detection using hue (from HSV) decreases, and often, the hue changes for the same markers.

The problems identified above means that the color of the marker differs during the system life-cycle. We can increase the quality of the analyzed images, e.g., by equipping our system with higher-quality cameras (frequency greater or equal to 120FPS at higher resolution), thus reducing the problem of unevenness of color perception. However, such a decision affects the cost of the system; we considered the adopted solution of Sony PS4 cameras as a reasonable compromise between quality and price.

2. EXPERIMENTS

We have tested six markers, each in one color. The markers were in the same position, which facilitated the acquisition and limited the impact of changes in brightness. The hue was represented by one byte, i.e., the possible range for hue is 0..255. We have tested markers twice: first, as static (fixed position), and as moving marker (marker moved by the user) in a small office room.

	Static			Moving		
	Min	Max	Diff	Min	Max	Diff
MG	211	222	11	202	244	42
GN	82	85	3	76	85	9
RD	174	176	2	171	178	7
BL	0	10	10	0	39	39
CY	33	48	15	29	55	26
YL	127	131	4	113	154	41

Table. 1. Hue range for marker tests at short distances (less than 5m).

For the static marker – the changes in the hue value were small; for such conditions, we can define much more color range and precisely identify markers. For the moving markers, the differences are bigger, and we can see that the fixed ranges for camera types cause the marker to misinterpret the color.

We have proposed an algorithm that uses hue and marker tracking. If the marker can be tracked, we

smooth the marker position and compare the hue with the previous color. If the previous color can be tolerated (i.e., the distance between hue and color hue limits is acceptable), the previous color is set; otherwise – we use hue ranges for colors.

The change of space made it necessary to work at distances of up to seven meters. Tests quickly showed that the method that works well in office conditions is effective up to approximately 5 meters. An additional two meters forced a change in the acceptable size of the marker. For the dark markers (displaying the primary color (red, blue, green) means one component of the RGB LED is lit, while the other three colors (magenta, yellow, and cyan) mean two components are lit and the brightness is higher), we have to reduce the minimal brightness of the acceptable marker pixel, which increases the influence of the camera noise. The color identification algorithm has become unreliable.

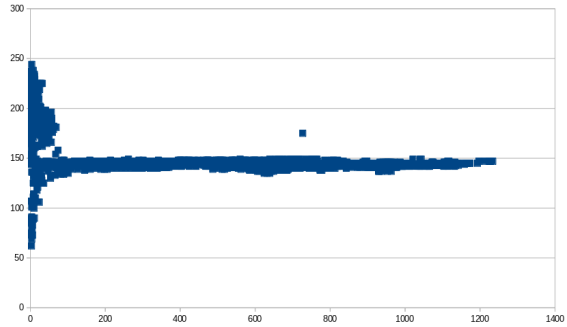


Figure 6. Relation between hue (y) and the size (x) of the marker in pixels (for yellow marker).

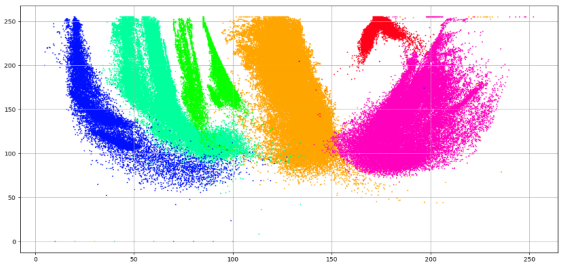


Figure 7. Pairs hue saturation for all six used colors. Hue: horizontal; saturation: vertical. The colors used correspond approximately to the colors of the markers.

As an example of the problem, we can see (Fig. 6) the hue values for one yellow marker in relation to the marker size.

A part of the wrong H values for the marker comes from the unnecessarily recognized marker reflection, but it is a potential source of error, and we should be ready to handle it.

The tests indicate that markers bigger than 20–25 pixels can be properly recognized (the ranges for colors do not overlap), but for smaller markers, overlapping ranges are possible. The tests also suggest that while hue ranges are often overlapped, a pair hue-saturation, or hue-value, is rarely overlapped, and triplet hue-value-saturation is even more rarely overlapped.

The data obtained for the calculation of the hue-saturation (fig. 7) and hue-value (fig. 8) dependence come from eight PS4 cameras for individual colors: blue was calculated using 21033 measurements of moving marker's color (worn and moved by the player, simulating gameplay behavior and using all available room space), cyan using 112769 measurements, green using 105709 measurements, magenta using 68132 measurements, red using 83070 measurements, and yellow using 124841 measurements. Totally, 515554 measurements were used with the final setting of the system (camera parameters, etc.).

On figs. 7 and 8, we can see the overlapping areas for hue ranges: blue, cyan, and green have seamless transitions; a similar situation can be seen for green and yellow, and red and magenta (the magenta range covers the whole red range).

For practical purposes, we have reduced our usage of blue (we have some problems with this color for marker detection: the marker was poorly detected and color was sometimes confused with cyan) and proposed a new algorithm for color detection.

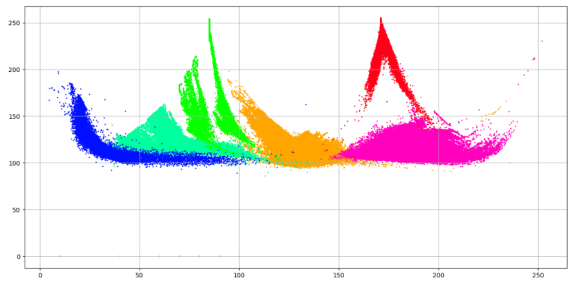


Figure 8. Pairs hue value for all six used colors. Hue: horizontal; value: vertical.

3. PROPOSED ALGORITHMS

There are two main procedures: color recognition and marker tracking/recognition.

The first procedure is used for the base recognition of the color: we have developed tables (fig. 9, 10) for this application that are based on the data from fig. 7, 8; the space has been transformed into a table (one table field represents space 1x1).

In the table we store:

- certain colors: where hue-saturation and hue-value gives the same color, or where certain color was matched with uncertain;
- uncertain colors; for pairs where data comes from at least two color markers, the more likely color is pointed;
- unknown color – there are no data and no guesses what color the pair represents.

The color of the found marker is represented by two parts: the main color and the complementary color. At the same time, both colors can be described as 'uncertain'.

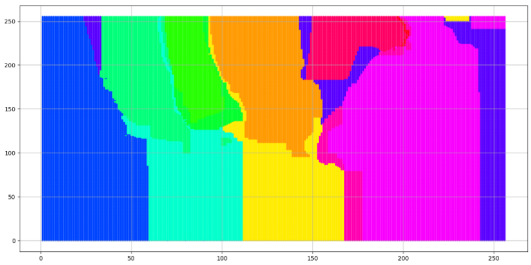


Figure 9. Hue-saturation table for color detection (man-made based on measurement data).

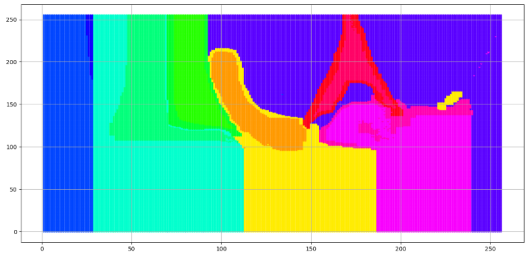


Figure 10. Hue-value table for color detection (man-made based on measurement data).

The algorithm calculates HSV for the found marker and looks for the proper color in both tables (hue-saturation, Fig. 9, and hue-value, Fig. 10).

```
//color:
// mainCol, complementaryCol, certain

colorUsingHSV(h, s, v) → color
if hsCol[h,s].val==hvCol[h,v].val
  return color(hsCol[h,s].val, null, true)
else
  if (hsCol[h,s].certain) and
           (hvCol[h,v].certain)
     return color(hsCol[h,s].val,
           hvCol[h,v].val, false)

  else if (hsCol[h,s].certain)
     return color(hsCol[h,s].val, null, true)
  else if (hvCol[h,v].certain)
     return color(hvCol[h,v].val, null, true)
  else
     return color(hvCol[h,v].val, null, true)
```

The second procedure is used for markers tracking:

```
pos2d = findCenterOfGravity(marker)
rgb = findRGB(marker)
hsv = rgb2hsv(rgb)
col = colorUsingHSV(hsv)

for old in allMarkersPrevFrame:
  spatDist = dist(odl.pos2d, pos2d)
  colDist = colDist(old.hsv, hsv)
  if (spatDist<maxSpatial) and
        (colDist<maxCol)
     new2d = (a*pos2d+b*old.pos2d)/(a+b)
     if (old.col == col)
       newMarkers.add(marker(new2d, old.col,
           hsv, true))
     else if (old.col.certain)
       newMarkers.add(marker(new2d, old.col,
            hsv, true))
     else if (col.certain)
       newMarkers.add(marker(new2d, col,
           hsv, true))
     else
       newColor = color(old.col, col, false)
       newMarkers.add(marker(new2d,
           newColor, hsv, true))
  else
    newMarkers.add(marker(pos2d,col,hsv,false))
```

The marker "is visible" (sent to the further elements of the system) if it was found in a series of markers in consecutive frames (with a given threshold). If the marker was not found in the analyzed frame but was found in the series of previous frames, it is still "visible" (if the visibility break was not too long). This increases the stability of our system.

4. EXPERIMENTAL RESULTS

Using a system of 8 cameras (PS4) and 4 computers, we collected data on calculated HSV triplets for markers. The tests were carried out in conditions imitating the target ones, i.e., the user walked with a marker at different speeds throughout the entire system operation area.

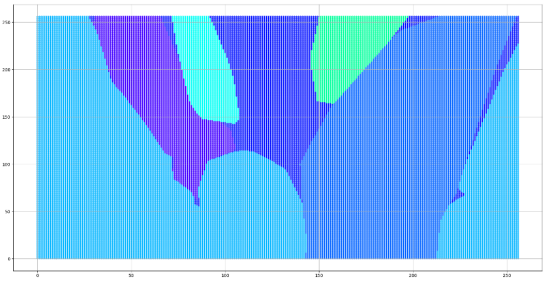
Tests were conducted separately for subsequent colors to ensure that the collected HSV triplets corresponded exactly to a given color. For the final tests we collected a new of 262281 samples divided into six colors. In the tests, we divided the dataset into a training and a test subset.

When tested for a sample of 268281 measurements, there were 15497 false matches using the above algorithm; that is 94.2% correct matches. In practice, the results obtained are corrected in the system based on the measurement history, and then the change in position is analyzed, practically ensuring that the room can be navigated safely and realistically.

We have tested some machine learning methods to compare results. We used saved test data to determine quality (ML techniques were not tested with a real-time system). SVC classificators (with

RBF kernels—independent classifiers for different colors) gave 1276 wrong outputs for 53657 HSV positions (size of test set; the remainder of the 268281 measurements was used as a learning set) —which means proper results where achieved for 97.6% (for polynominal kernels: 95.2%). Some colors were classified better (e.g., for yellow, red, and magenta, the proper results were above 99%), while others were classified worse—for blue, it was only 86.8%, and for cyan, it was 96.7%.

For decision trees, the results were similar: 97.5% properly classified triplets h, s, and v; the best results (better than 99%) for yellow, green, magenta, and red; the worst for blue (87.7%). We have also tested neural networks (simple perceptrons) from SciKit-Learn. Simple perceptron (for three parameters) gives us 97.7% proper results. The worst results were given by a blue marker – 91.5%. We have used the same method (but models were built for pairs: h, s, and h, v) to create new tables, and our simple algorithm yielded 97.1% correct matches. Perceptron build for h-s pairs, gave 64.5% proper answers, and for h-v pairs, gave 97.5% proper answers.



Figiue 11. Color maps h-s created by multilayer perceptron. Note: The colors do not correspond to the actual marker colors or the colors in the earlier illustrations.

The worst results where given by the blue marker; after eliminating it (it means using blue and cyan as one color), the neural network gave 99.4% proper results for triplets h, s, and v, while the neural networks for the h-s pair and the h-v pair gave respectively 99.2% and 99.4%.

We have used perceptron to create new color maps for h-s (fig. 11) and h-v (fig. 12). Tests suggest that new color maps give slightly better results with our algorithm.

We performed the above tests using Python (with the Anaconda environment and the SciKit Learn library), which we also used to build tables describing the relationships of h-s and h-v pairs with colors.

When testing the algorithms above, we were interested in quality—to identify candidates for system improvement. So we didn't test the execution time.

Otherwise, in the real system, we put great importance on the calculation time. There, we examined the time needed to calculate position information, and we built both programs (detecting markers and their colors, calculating 3D positions for markers) using the Xenomai library, which ensures real-time implementation. We assumed an 8-ms loop for both programs.

In the tested configuration our system consists of four computers to which 8 cameras are connected to (2 per computer), and each camera corresponds to one running marker detector application. One of the computers runs a “coupler" app, a program that calculates the 3D position from the positions sent by the marker detectors; and on another of the 4 computers, there is a server prepared in Unity, which is used to coordinate game events between client applications (running on stand-alone VR HMD).

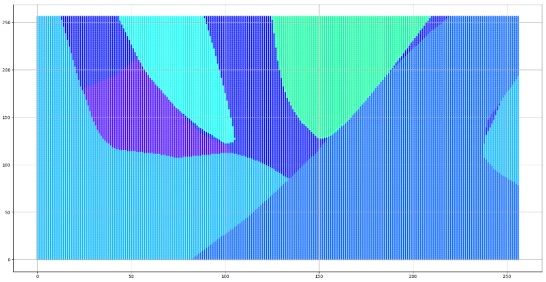


Figure 12. Color maps h-v created by multilayer perceptron. Note: the colors do not correspond to the actual marker colors or the colors in the earlier illustrations.

Average calculation time for 1003 test frames in a typical configuration (2 PS4 cameras connected via USB ports to a computer with Linux, 16GB of RAM, and an i5 processor, 2.20GHz; so two marker detection programs were running in parallel on one computer), total time for calculating the position and color of the marker (from the image acquisition to sending output data to the next program) was on average 1.51ms. The calculation in the second program, which determines the player's position from the sent markers and transmits it to the players, takes an average of 1.08 ms.

5. SUMMARY

To summarize, we developed an algorithm for identifying the marker color. We demonstrated that the changes in the observed marker are significant and require the development of an appropriate algorithm. The algorithm is simple, highly reliable, and fits our problem. We have also tested ML

algorithms, and simple perceptrons give better results than our algorithm. While this is better solution, transferring it to a real-time tracker could present difficulties, so we only used this solution to improve the recording of the color tables for our algorithm. The transfer of neural network solutions to the real-time tracker can be done as part of the work to improve the system in the future.

Acknowledgments

This work was supported by the NCBiR (National Centre for Research and Development), project VEPP (Virtual Entertainment Enhanced Platform), no. POIR.01.02.00-00-0155/17-00.

BIBLIOGRAPHY

[1] Muhammet Fatih Aslan, Akif Durdu, Kadir Sabanci, *Shopping Robot That Make Real Time Color Tracking Using Image Processing Techniqes*, Journal of Applied Mathematics, Electronics and Computers, 5(3):62-66, 2017, DOI: {10.18100/ijamec.2017331881}

[2] Homayoun Bagherinia, Roberto Manduchi, *Robust real-time detection of multi-color markers on a cell phone*, Journal of Real-time Image Processing, 8:1-17, 2013, DOI: {10.1007/s11554-011-0206-9}

[3] Fabrizio Cutolo, Cinzia Freschi, Stefano Mascioli, Paolo Parchi, Mauro Ferrari, Vincenzo Ferrari, *Robust and Accurate Algorithm for Wearable Stereoscopic Augmented Reality with Three Indistinguishable Markers*, Electronics, 5, 2016, DOI: {10.3390/electronics5030059}

[4] Joseph DeGol, Timothy Bretl, Derek Hoiem, *ChromaTag: A Colored Marker and Fast Detection Algorithm*, Proceedings ofInternational Conference on Computer Vision, Venice, 2017, DOI: {10.1109/ICCV.2017.164}

[5] Hesam Eskandari, *Detection and tracking of sphere markers*, Master Thesis, École de technologie supérieure, Montreal, 2009

[6] Rabah Hamdini, Nacira Diffellah, Abderrahmane Namane, Abderrahmane, *Robust Local Descriptor for Color Object Recognition*, Traiment du Signa, 36(6):471-482, 2019, DOI: {10.18280/ts.360601}

[7] Allan Hanbury, *Constructing cylindrical coordinate colour spaces*, Pattern Recognition Letters, 29(4): 494-500, 2008, DOI: {10.1016/j.patrec.2007.11.002}

[8] Priyanto Hidayatullah, Miftahuddin Zuhdi, *Color-Texture Based Object Tracking HSV Color Space and Local Binary Pattern*, International Journal on Electrical Engineering and Informatics, 7(2):161-176, 2015, DOI: {10.15676/ijeei.2015.7.2.1}

[9] Przemysław Kowalski, Krzysztof Skabek, Jan Mrzygłód, Jan, *VEEP – The System for Motion Tracking in Virtual Reality*, Proceedings of 6th International Conference on Man-Machine, Cracow, 1:12-22, 2019, DOI: {10.1007/978-3-030-31964-9_2}

[10] Liu Jiamin, Chen Shuo, Sun Hongxing, Qin Yongxu, Wang Xibo, *Real Time Tracking Method by Using Color Markers*, Proceedings of \em International Conference on Virtual Reality and Visualization, X'ian, 1:106-111, 2013, DOI: 10.1109/ICVRV.2013.25}

[11] Martin Loesdau, Sébastien Chabrier, Alban Gabillon, *Hue and Saturation in the RGB Color Space*, Proceedings of International Conference on Image and Signal Processing, Cherbourg, 1:203-212, 2014, DOI: {10.1007/978-3-319-07998-1_23}

[12] Chanh-Nghiem Nguyen, Van-Thoai V, Nguyen Cong Ha, Developing a computer vision system for real-time color measurement – A case study with color characterization of roasted rice, Journal of Food Engineering, vol. 316, 2022, DOI: {10.1016/j.foodeng.2021.110821}

[13] Erin Pangilinan, Steven Lukas, Vasanth Mohan, *Creating Augmented & Virtual Realities: Theory and Practice for Next-Generation Spatial Computing*, O'Reilly, 2019

[14] Patrick Sebastian, Vooi Yap, Ross Comley, *The effect of colour space on tracking robustness*, Proceedings of IEEE Conference on Industrial Eletronics and Applications, Singapore, 1:2512-2516, 2008, DOI: {10.1109/ICIEA.2008.4582971}

[15] Minjie Wan, Guohua Gu, Weixian Qian, Kan Renm, Qian Chen, *Hue preservation based color space transformation for brightness-robust tracking*, Optik - International Journal for Light and Electron Optics, 144, 2017, DOI: {10.1016/j.ijleo.2017.06.073}

[16] Yueting Zhuang, Yunhe Pan, Jun Xiao, *Human Motion Capture Using Color Markers*, in A Modern Approach to Intelligent Animation: Theory and Practice, Springer-Verlag, Berlin Heidelberg New York, 59-75, 2008, DOI: {10.1007/978-3-540-73760-5}