CameramanVis: where the camera should look?
CPSC 547 Project proposal

Jianhui Chen
Computer Science Department
University of British Columbia
jhchen14@cs.ubc.ca

1. Domain, task and dataset

In the soccer broadcasting, human camera operators adjust the pan angle of a pan, tilt and zoom (PTZ) camera to follow player positions and ball position. In this project, we assume the camera only has 1 degree of freedom (DoF) which is the pan angle. And the player positions are the main targets when the cameraman shots the video. The project aims to analyze the relationship between the pan angle and player position.

The dataset is from my research project. It was collected from a semi-professional soccer game. The dataset has 172, 800 frames (48 minutes). Each frame contains a set of player positions and a camera pan angle. The player position is measured on the playing ground and are quantized to a 14 dimension feature.

For simplicity, we define following terms:

- **frame**:one image in a video.
- **frame number**: a non-negative integer. It is quantized time.
- **camera angle**: camera pan angle. It is a scalar value in one frame.
- **player location**: player locations on the playing ground in one frame. It is a set of 2D locations.
- **feature**: quantized player locations in one frame. It is a 14 dimension vector. The quantization method is described in [3].
- **sample**: a pair of a camera angle and a set of player position that are captured (detected) at the same frame.
- **camera angle distribution**: a histogram of a set of camera angles.
- **feature distribution**: mean and standard deviation of features. We assume each dimension is independent.
- **feature distance**: Euclidean distance between two features.

To reduce the redundant information and make the program faster, we uniformly sampled data from the original data. To evaluate the information loss in sampling, we linearly interpolate the whole dataset from sampled data and compared it with the original data. We calculate the average value of difference between the original data and the interpolated data to measure the reconstruction error. Fig. 1 shows camera angle reconstruction error with different sampling steps. Fig. 2 shows player number reconstruction error with different sampling steps. To balance the recon-

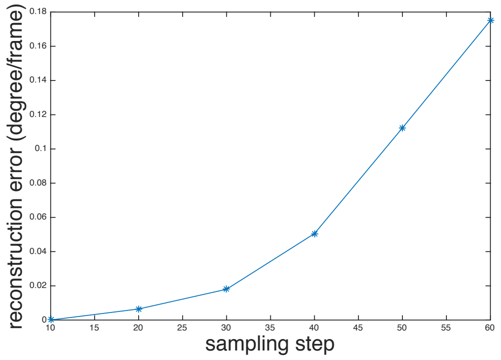


Figure 1. Camera angle reconstruction error from linear interpolation. We choose sample step 30 in which the average reconstruction error is about 0.02^o per frame.

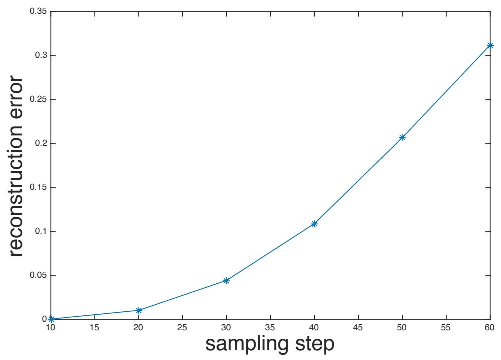


Figure 2. Player number reconstruction error from linear interpolation. We choose sample step 30 in which the average reconstruction error is about 0.05 persons per frame.

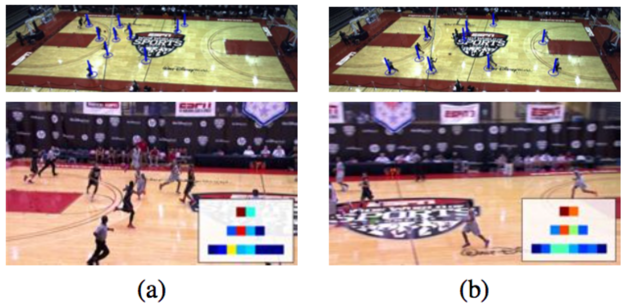


Figure 3. An example of multivalued function. (a) and (b) show two roughly similar distributions of players at different times in the game. The resulting feature (inset) are also quite similar. However, the pan angles of the human operated camera are substantially different. Figure adapted from [3].

struction error and the size of sampled dataset, we chose sampling step as 30. As a result, the dataset size becomes 5,760 after sampling.

The task of this project is analyzing the relationship between the camera angle and player positions on different scenarios:

- Overview of pan angle distribution, player distribution and their relationship.
- Identify the camera angle distribution by given a feature.
- Identify the feature distribution by given a camera angle.
- Identify outliers that camera looks at unexpected angles.

2. Personal expertise

I worked in this area more than one year. Identifying the outliers is one of ways to explain the challenges in the data to reviewers and improve my research. For example, I have to find two frames that have similar player distribution but have very different pan angle (see Fig. 3). I have used Matlab for this task. First, I calculate two distance matrices. The first matrix is the distance between features. The second matrix is the distance between camera angles. Both matrices are visualized by color heat map. After that, I manually look into small value areas in the first matrix (similar features) and the large value areas in the second matrix (different camera angles) to find out outliers. Because I have to switch between two heat maps, it took me about one hour to find satisfied frames. The process is very painful and hurts my eyes. Also, I am not sure the outliers I found is the most significant one or it is representative. Now, the scale of the dataset grows from several minutes to near an hour. A good visualization tool is urgently necessary.

I have many years programming experience using C++ but I only did a small project using JavaScript three years ago. I am a new learner for creating visualization tools using D3. But I believe InforVis will be a very important skill set for my career. I speed about two hours per week to learn D3 by coding the examples in the book [5].

3. Proposed infovis solution

In the solution, the frame number is used as the key to query both camera angles and features. For camera angles, I will use line chart. For player positions, I will use a discretized heat map on top of a court layout. For the feature, I will use color heatmap.

The primary goal of this project is building a tool to help computer vision and machine learning researchers quickly explore the relationship between camera angles and player positions. There are three different views I hope to create: over view, query view and player location view. I recognize there may not be enough time to complete all the views, and so I have assigned a priority ranking to each of them, with the intent to complete those ranked 1, and the others if time permits.

Over view (rank 1) This view provides global view of samples. A line charts shows the camera angles over time. A heat map shows the feature over time.

There will be detailed views below the global camera angle view and feature view. There will be selection operators so that the detailed views only visualize selected data. There will be linked highlighting between the camera angle view and feature view.

Query view (rank 1) This view provides query result by user's input. It will support following queries:

- Query one: feature distribution
  Input: an range of camera angle.
  Input visualization method: sliding bar.
  Output: feature distribution.
  Output visualization method: bar chart and error bar.
  Complexity: O(N).
- Query two: camera angle distribution
  Input: a feature and a distance threshold.
  Input visualization method: mouse click and sliding bar.
  Output: pan angle distribution.
  Output visualization method: bar chart histogram.
  Complexity: O(N).
- Query three: outlier detection
  Input: a sequence of samples and an angle threshold.
  Input visualization method: mouse selection and slider bar.
  Output: two frames that have the smallest feature distance but their angle distance is larger than the threshold.

Output visualization method: text output frame numbers, angles and features.
Complexity: O(dN^2) in which d is dimension of features and N is dataset size.

Play location view (rank 2) In the over view and query view, features are visualized instead of the original player locations. It is easy to calculate distance between features. But it is not very intuitive for where the player locations are. In this view, the original player locations will be visualized by overlaying 2D marks on top of a court layout. There will be linked highlighting between the operators in the query view (query one and query three) and this view.

The focus of our work is to explore the relationship between the camera angles and the player positions. Because this relationship is a time sequence, interactive and quick response is very important. I hope all the operates in the project can have real-time response.

4. Scenario of Use

John is a computer vision scientist working on a robotic camera planning project. He want to analyze how human operators control the camera to improve his automatic system.

First, John loads a recently collected data to the CameraManVis. The system gives him the over view (4). He compares camera angles and features in the side-by-side views. He feels like the relationship between these two views is reasonable in general. He thinks it might be helpful to only look a short sequence of the data. He selects a short sequence in the camera angle view. The detailed view (under the over view window) immediately shows the pan angle and the feature map. He feels the all the data so far is reasonable.

Then, John wants to analyze the distribution of pan angle and features to further check the data. He wants to know the variance of the human operators given similar features. So, he randomly selects one frame from the dataset and sets a distance thresholds using the slier bar (Fig. 5 query two). The sub-window on the right of the slider bar immediately shows a histogram of pan angles as John changes the threshold value. John finds the distribution changes from a single Gaussian distribution to multiple Gaussian distribution, and eventually to some arbitrary distribution when the threshold becomes very large. He records two thresholds manually on his notebook: one is from single Gaussian to multiple Gaussian, another is when the distribution becomes arbitrary. He thinks for a while: these threshold values can be used as parameters in his automatic system.

Before John goes to back to coding work, he thinks it is interesting to look up some outliers where the camera looks at unusual angles. To get more intuitive information, he navigates to outlier detection in player location view (Fig. 6). He clicks one frame in the angle view and sets an angle threshold using the slider bar. As he moves the sliding bar, the player locations dynamically change on the right side. For a particular angle threshold, he finds that the player positions are very different in the frame one and the frame two. He looks up the number of players in both frames. The numbers are 20 and 22, which are very similar. He can not figure out why that happens. He records the frame numbers in this notebook so that he can study it future using off-line soccer videos

5. Proposed implementation approach

My solution will be built on D3 [1], with the intention to learn from various examples from the internet. I want to keep the technique difficulty under control and focus on the visualization. The project will be a single web page and I will only implement the front-end part. I plan to implement most of the code using raw D3 framework. If I feel confident using D3 during the project, I will try use some plugs such as Cubism. Meanwhile, I am getting ideas from pre-existing projects such as Buckets.

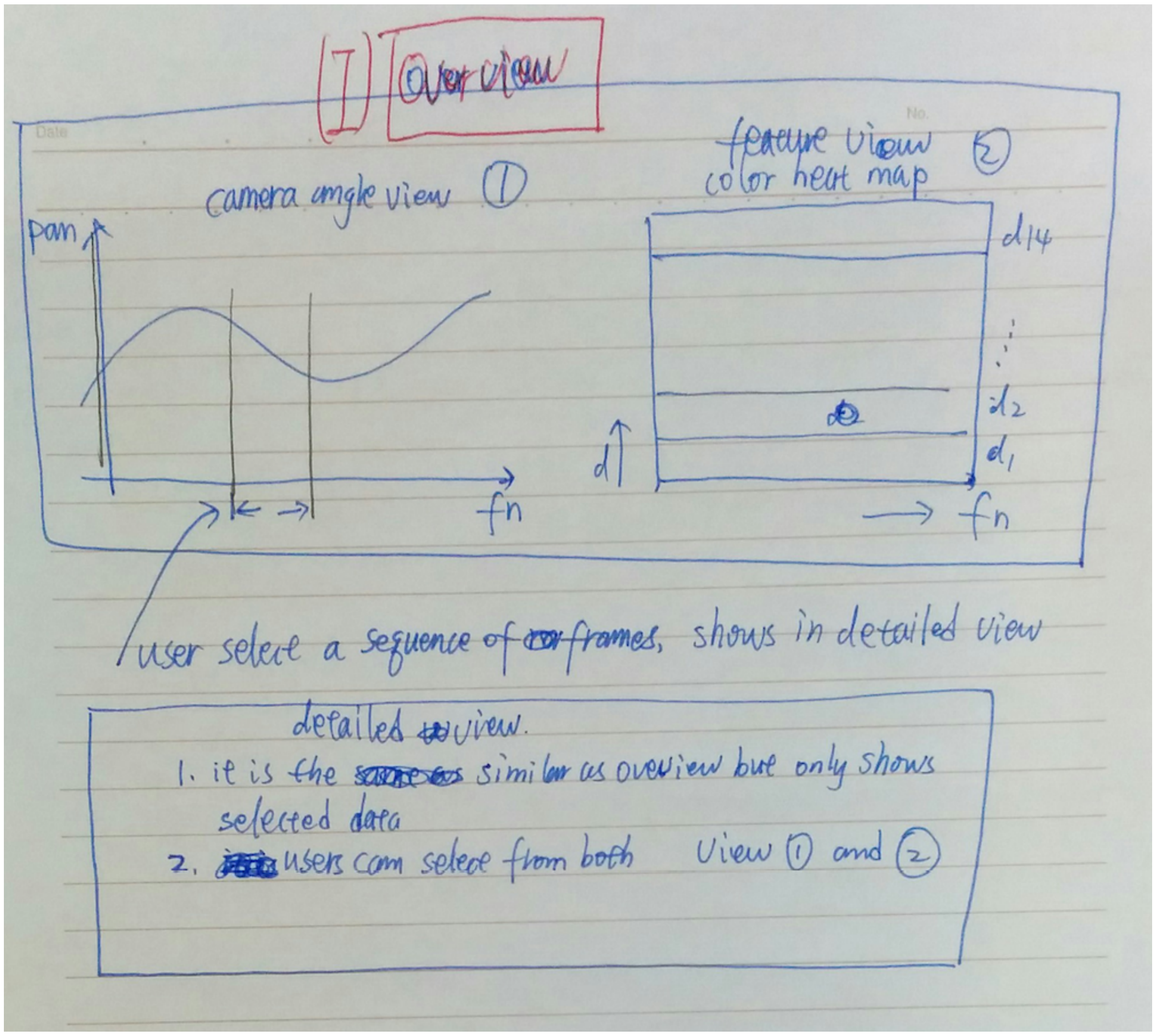
6. Milestones

- **Nov 10** Start camera angle (line chart) and feature (heat map)views.
- **Nov 17** Finish linked highlighting between line chart and heat map.
- **Nov 23** *Status updates due*. Finish query one and query two.
- **Nov 30** Finish query three.
- **Dev 7** Complete the over view and query view. Start prepare presentation.
- **Dev 15** *Presentation deadline*. Start player location view. Draft the final paper.
- **Dev 18** *Paper due*. Finish the paper and record the demo video.

7. Previous work

Visualization of camera angle and player positions have been studied separately in the literature. Camera calibration toolbox [2] is a general purpose camera pose visualization tool. As we only visualize the pan angle of the camera, we does not require the 3D visualization. We regard pan angle as a regular time sequence.

Player position/trajectory visualization have been intensively studied. For example, Perin *et al.* [6] designed a system that provides an overview+detail interface of soccer game phases. In order to improve story telling and save times for experts such as online journalists, they designed



user select a sequence of ~~car~~ frames, shows in detailed view

detailed ~~to~~ view.

1. it is ~~the~~ similar as overview but only shows selected data
2. users can select from both View ① and ②

Figure 4. Over view. This view supports selection operator. Users can select and visualize the data from the whole dataset.

a series of connected visualizations. Hamid *et al*. [4] visualized the player position under multiple static cameras by transforming player locations from multiple camera views to a common ground plane.

References

[1] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.

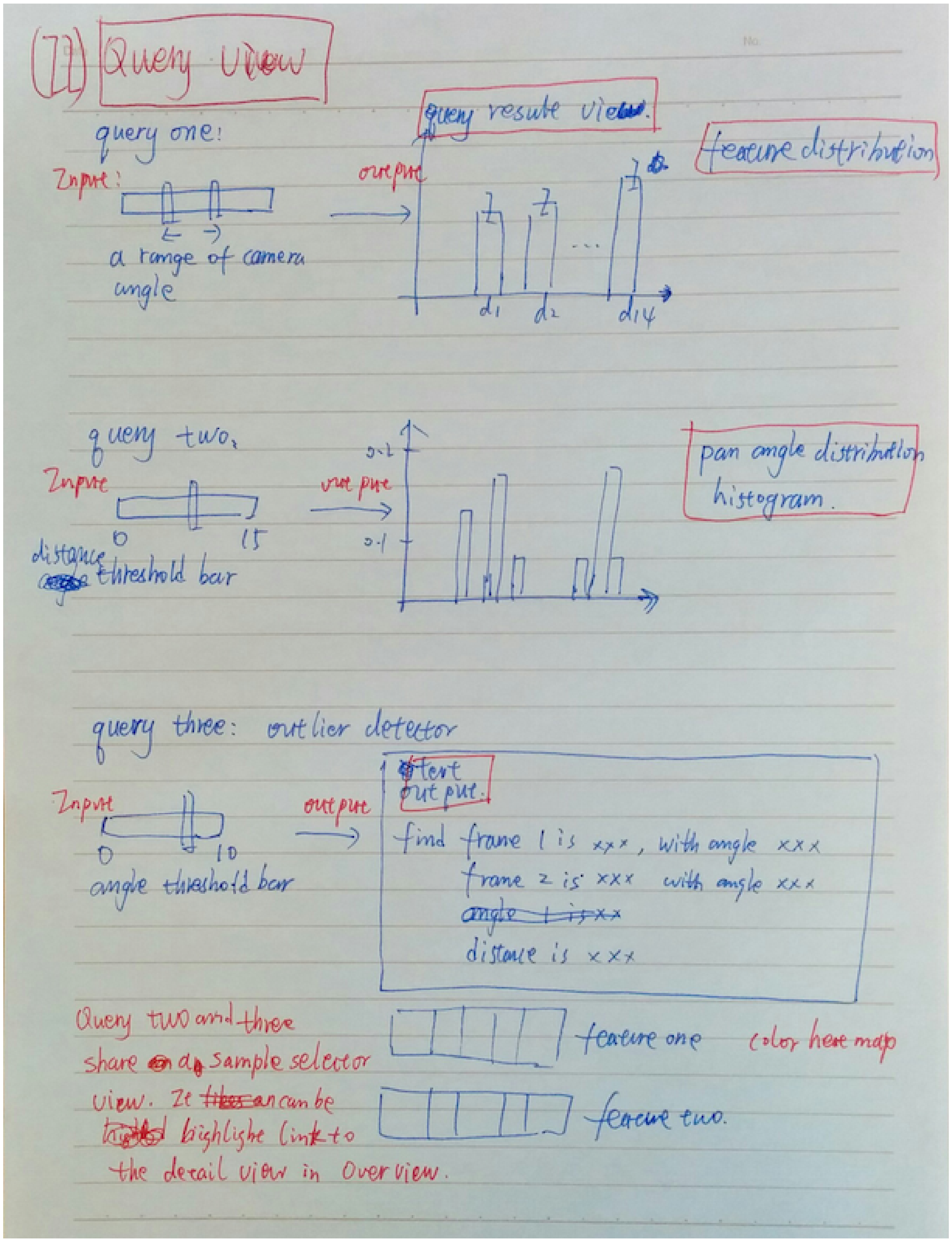
[2] J.-Y. Bouguet. Camera calibration toolbox for matlab. 2004.

[3] J. Chen and P. Carr. Mimicking human camera operators. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 215–222. IEEE, 2015.

[4] R. Hamid, R. Kumar, J. Hodgins, and I. Essa. A visualization framework for team sports captured using multiple static cameras. *Computer Vision and Image Understanding*, 118:171–183, 2014.

[5] S. Murray. *Interactive data visualization for the Web*. O'Reilly Media, Inc., 2013.

[6] C. Perin, R. Vuillemot, and J.-D. Fekete. Soccerstories: A kick-off for visual soccer analysis. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2506–2515, 2013.



Query two and three share a sample selector view. It ~~then~~ can be ~~[illegible]~~ highlight link to the detail view in Overview.

Figure 5. Query view. This view supports three query operators. The query has real-time response.

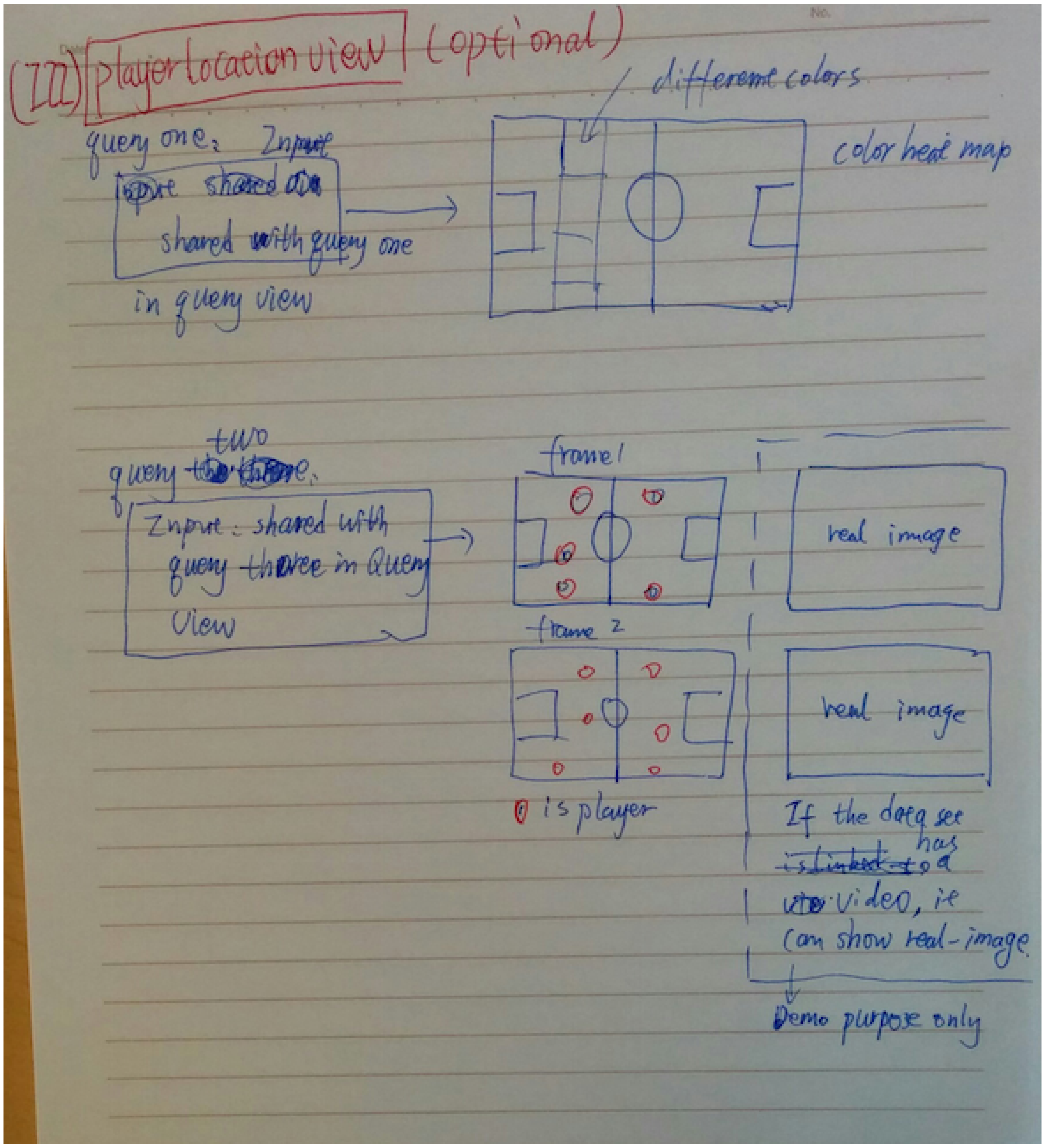


Figure 6. Player location view. This view is complementary to the query view in 5. The query result is player locations on the playing ground so it gives more intuitive information.