On Achieving Diversity in the Presence of Outliers in Participatory Camera Sensor Networks

Md Yusuf Sarwar Uddin, Md Tanvir Al Amin, Tarek Abdelzaher, Arun Iyengar, Ramesh Govindan
{mduddin2, maamin2, zaher}@illinois.edu, aruni@us.ibm.com, ramesh@usc.edu

***Abstract*—This paper addresses the problem of collection and delivery of a *representative subset* of pictures, in participatory camera networks, to maximize coverage when a significant portion of the pictures may be redundant or irrelevant. Consider, for example, a rescue mission where volunteers and survivors of a large-scale disaster scout a wide area to capture pictures of damage in distressed neighborhoods, using handheld cameras, and report them to a rescue station. In this participatory camera network, a significant amount of pictures may be redundant (i.e., similar pictures may be reported by many) or irrelevant (i.e., may not document an event of interest). Given this pool of pictures, we aim to build a protocol to store and deliver a smaller subset of pictures, among all those taken, that minimizes redundancy and eliminates irrelevant objects and outliers. While previous work addressed removal of redundancy alone, doing so in the presence of outliers is tricky, because outliers, by their very nature, are different from other objects, causing redundancy-minimizing algorithms to favor their inclusion, which is at odds with the goal of finding a representative subset. To eliminate both outliers and redundancy at the same time, two seemingly opposite objectives must be met together. The contribution of this paper lies in a new prioritization technique (and its in-network implementation) that minimizes redundancy among delivered pictures, while also reducing outliers.**

I. Introduction

This paper addresses a novel problem that arises in the context of participatory camera sensor networks; namely, that of collection and delivery of the most *representative subset* of pictures from a vast pool, where a significant portion of pictures are redundant, irrelevant, or noisy. A representative subset is one that offers roughly the same coverage of the environment, but with fewer pictures.

We define a participatory camera (sensor) network as one where participants contribute pictorial data, either on their own initiative or through participation in a corresponding data collection campaign. For example, in the aftermath of a natural disaster, relief workers and other first responders might survey an area in search of damage that is then pictorially documented and reported. Another application might be to ask residents of a neighborhood to pictorially document issues that require attention in their neighborhood (e.g., graffiti on walls, trash piles, hazardous potholes, or other problems). Yet a third application might be to compile a list of most visited tourist landmarks from pictures contributed by tourists in a given location. Participatory camera sensing applications are made popular by the vast proliferation of cameras and camera phones in the possession of the average individual, not to mention the richness of information contained in pictures compared to other sensing modalities.

Our camera sensing service runs on participants' phones (the clients) and on a destination server (the collection point). When pictures are taken using our application, they are locally stored on the phone. When two participant phones meet, they may gossip by exchanging a portion of their pictures. Similarly, when a phone connects to the destination server it uploads a portion of its pictures. The contribution of the paper lies in prioritizing transmission of pictures both when two phones meet or when a phone meets the server, such that the most *representative subset* is sent (instead of sending all), in order to conserve resources. Resources may need to be conserved for many reasons. For example, participants, who upload pictures from their mobile phones, may have to pay for their data plans. Hence, uploading less data is better. If pictures taken by participants propagate opportunistically, for example over a disruption-tolerant network (DTN), an individual participant may end up collecting too much redundant content and may need to do some data triage to fit the local storage or energy constraints. Such might be the case in disaster recovery scenarios, where infrastructure may be destroyed leaving only DTN-style communication, or in military scenarios, where groups of soldiers in the field may have only a low bandwidth channel to a remote base, making it advantageous to triage the data locally prior to transmission on that channel.

We do not make inherent assumptions regarding the type of network in which our service operates. For example, it could be a star network, where all phones have a direct way of connecting to the server whenever they want. Alternatively, it could be a DTN where the primary data propagation occurs via phone-to-phone gossiping. The type of the underlying network is a routing issue. Either way, the decision we are concerned with is which pictures to send in what order when two nodes meet (either two clients, or a client and the server).

In this paper, resources are conserved by addressing two inefficiencies in participatory camera networks. The first lies in the inherent redundancy among returned pictures. For example, if a community of volunteers are independently uploading pictures of flood damage in their neighborhood after a storm, the server might already know of some of the damage (from earlier uploads) and hence may not need some of the pictures. The second lies in the existence of noise and outliers that are not representative, since the participants may not always be entirely reliable. For example, not all returned pictures will be related to the task at hand. To conserve resources, one would like to minimize redundancy while eliminating the outliers. Note that, the objectives of reducing redundancy and reducing outliers are at odds. Outliers, by definition, are different from

other pictures and hence, not redundant. We show that algorithms that minimize redundancy alone, such as those proposed in previous literature [1], favor outliers as opposed to more representative content. The main contribution of this paper therefore lies in combining redundancy minimization with outlier elimination in participatory camera sensing networks. Towards that end, we propose a technique to identify outliers, propose a new metric for content diversity, and develop a new rule for content prioritization, where outliers receive lower priority, while diversity is maintained.

It is worth noting, at this point, that outlier elimination is not always a goal in a participatory camera network. In some applications, such as anomaly detection, outliers are in fact what carries the relevant information. For example, an in-store security camera might report the same view all night, except when an intruder breaks in. A frame with the intruder in view might be the outlier, but it is also the frame that contains the most interesting information. This paper considers a different type of applications, where a community of users document relatively static conditions in the environment, such as damage or points of interest. In such cases, one is not looking for anomalies in reporting, but rather for *representative* depiction.

To provide context for our service, we shall use a rescue mission in a post disaster scenario as a running example. Rescue workers, volunteers and survivors scout a distressed area, capture pictures of damage and report them to a command station. The regular communication infrastructure is not available in the aftermath of our disaster, due to infrastructure damage or power outage. Instead, a DTN is formed among the mobile devices used by the participants themselves. This DTN is used for ultimate picture delivery to the command station. Note, however, that our prioritization schemes are more general, and apply to other network contexts as well.

The rest of this paper is organized as follows. Section II describes our theoretical construct of outlier-resilient diversity-aware collection service, ORPNet (Outlier Resilient Picture Network). The prioritization scheme for synchronizing content across nodes is described in Section III. Section IV presents performance results and comparison with earlier work, followed by Section V that reviews related literature. Section VI concludes the paper and presents closing remarks.

II. Diversity in Picture Collection

In order to select a representative subset of pictures that maximizes coverage using the fewest pictures, we focus on increasing *diversity* among the selected pictures to minimize overlap. One might be tempted to also favor large panoramic pictures, since they presumably offer more coverage. We do not take that route since often information is contained in the detail (e.g., a close-up of a crack in the wall might indicate a damaged building, but the crack may not show in a wide panoramic view). Since we do not know what the participants' regard at the important information in the picture, we take the more conservative approach of simply removing redundancy as a safer way to offer coverage with fewer pictures. To reduce unrepresentative outliers, we further refrain from selecting pictures that are not corroborated by others.

To implement the above selection mechanisms, we define a distance function that measures the level of similarity between pictures based on the degree of match in their visual features and in metadata between them. An important piece of metadata is location. For example, two buildings may appear visually similar, but if they happen to be in different locations, they must be different. Given an appropriate logical distance function to measure redundancy with, the diversity of a picture collection depends on distances between individual pictures in the collection. We attempt to maximize diversity while removing outliers. Some outliers can be detected at the source. For example, a picture that is blurry or otherwise of poor quality may not be useful, and hence can be discarded. Such quality problems can be handled easily by the user or by existing vision techniques applied in an automated fashion at the source, and are not the topic of this paper.

Instead, we attempt to infer relevance based on similarity of the picture to others, considering both geographic attributes and visual image features. The idea is that (versions of) more relevant scenes to the participatory sensing application will generally be photographed by more sources. By "scene", here, we mean a visual observation, such as the observation of a damaged bridge, a collapsed building, a blocked road, a fire, a car accident, a traffic jam, or an interesting person. An explicit goal is to estimate relevance *without having to understand the semantics* of what is in a picture, since this would be very complex, application-specific, and beyond the purview of a general service. Note that, short of truly understanding each picture, and short of understanding the application's mission, there is no error-proof way of assessing relevance of a picture to the mission. Hence, by necessity, we have to settle for an imperfect scheme in exchange for a higher degree of application-independence. A contribution of the paper, therefore, lies in proposing and assessing the performance of one such scheme empirically based on actual photographs and a representative application scenario. Evaluation shows that, despite its limitations, our scheme offers a significant improvement over entirely content-agnostic networks.

Our scheme explicitly ranks (i.e., prioritizes) stored pictures by their contribution to diversity and relevance, estimated in an application-independent manner from the inferred degree of participant interest. Hence, if for some reason there is no opportunity to send all pictures, a greedy algorithm can simply send the best ones by following our *priority order*. In the following, we detail various aspects of our scheme. In the discussion, we use the term picture and object interchangeably.

A. Picture Representation and Similarity Distance

A hallmark of our scheme lies in its separation between application-specific notions of "similarity" between objects, and the generic diversity-maximizing and outlier-eliminating prioritization scheme. The core of that separation lies in the definition of a *distance metric*, $d(x, y)$, between content objects x and y to denote their degree of similarity. Our scheme does not assume any specific distance metric. In other words, it is general in that it does not care how $d(x, y)$ is computed. The definition of the distance metric is, in fact, the

primary way our scheme can be customized for the needs of a particular mission or application scenario. Given a distance metric, the scheme can perform better or worse, depending on how representative this metric is of the amount of information overlap between content objects. The metric should yield a lower distance when there is more overlap.

For the application at hand, we argue that location plays an important role in defining logical distance. When the geographic distance between two pictures is beyond some threshold, say 200m, they are physically far enough apart that they are likely to be of different scenes, regardless of their visual similarity (e.g., if they are both pictures of burning cars, they are likely to involve different cars, even if the cars looked similar). Conversely, for pictures taken from almost the same location, it is the visual features of the respective images that give the best clue on whether they are of the same scene or not. Let us define T as the distance threshold beyond which we can safely assume that the pictures taken are of different scenes. Conversely, when pictures are taken less than distance T apart, we consider them to be originating from (roughly) the same location. As a means of normalization, when pictures are geographically distant, we set their logical distance to a value greater than 1; otherwise, we make it smaller than 1, in which case the logical distance should be dominated by visual difference.

Let $d_l(x, y)$ be the geographic distance between locations of picture x and y, and let $d_v(x, y)$ be their visual distance based on image features. We normalize the visual distance so that $0 < d_v(x, y) \leq 1$. We then combine visual similarity and location into a single uniform logical distance metric using the following expression:

$$d(x,y) = \begin{cases} 1 + \frac{d_l(x,y)}{T} & \text{if } d_l(x,y) > T \\ d_v(x,y) & \text{otherwise} \end{cases} \quad (1)$$

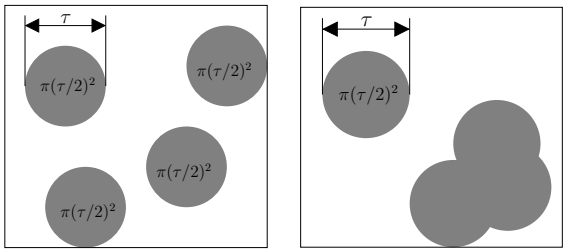
A pair of pictures are said to be *geographically collocated*, if the geographic distance between them is less than T, that is, $d(x, y) \leq 1$. Obviously, T depends on the application. For example, in a city, pictures taken more than a few blocks apart will likely be different so T is of the order of city blocks. For an indoor deployment inside a building, T might correspond to the size of a single room. In an exhibition setup, say in a museum, T can be even smaller (e.g., of the order of the neighborhood of a single exhibit in a room), because users' interest naturally clusters around different objects of the granularity of single exhibits.

B. Measuring Diversity of Picture Collection

The distance metric $d(x, y)$ allows for objects to be represented as points in a multidimensional logical space, where the proximity of points designates the similarity between the corresponding objects. If two points lie very close to each other, they have information overlap, which makes them partially redundant. The purpose of diversity maximization is to reduce overlap among selected objects, subject to resource constraints (e.g., limited storage size). This in turn implies choosing points that are distant in logical space.

We further assume that there exists a certain distance threshold beyond which there is no information overlap. Let this constant be τ. A good estimate of τ in the case of pictures, for example, would be $\tau = T$. Hence, it is useful to imagine that each object logically covers a hyper-sphere with radius $\frac{\tau}{2}$ so that the spheres of two objects overlap when their distance is smaller than τ. Overlapping spheres indicate existence of shared information between the corresponding objects. The volume of a sphere is called the *coverage* of a given object. For an n-dimensional feature space, this volume is proportional to τ^n.

Note that, due to overlap, the total coverage of a set of objects is generally less than the sum of the coverages of the individual objects. The total coverage of all objects in a set can thus be treated as a quantitative estimation of the diversity of the set. The diversity maximization problem is then to chose a subset of objects whose total coverage is maximum, subject to some aggregate resource constraint (e.g., storage capacity) that limits the number of objects chosen. Figure 1 illustrates an example case for a 2-dimensional space.



(a) (b)

Fig. 1. The object collection at (a) is more diverse than the collection at (b), because of greater coverage. In (b), similar objects are overlapped.

In practice, pictures taken by participants would typically fall into groups (each group representing pictures of the same scene at the same place), such that logical distances between pictures within the same group (or cluster) are much smaller than those among different groups. This naturally leads to partitioning objects into a set of *clusters*, so that similar objects are grouped into the same cluster.

Coverage of a cluster follows two simple properties. First, the coverage is *non-decreasing*, in the sense that as objects are added to a cluster, coverage can only increase (or stay the same). Second, it has a *declining marginal gain* in that the expected additional coverage from adding another object to the cluster declines as the size of the cluster grows (because spheres become more and more overlapped). Since the cluster is ultimately bounded in size, the infinite sum of all such increments is bounded. It is therefore useful to approximate this total cluster coverage, CC_k, for a cluster of k objects, by a geometric series of the form:

$$CC_k = CC_1(1 + \lambda + \lambda^2 + \cdots + \lambda^{k-1}) \quad (2)$$

where $\lambda < 1$. To compute a suitable value for λ in the above equation, it is useful to consider the infinite sum of the series. That is to say, it is useful to compute the coverage achieved in the limit, when the cluster size is very large.

Towards that end, consider a clustering algorithm that ensures that no two objects in the cluster are more than β distance apart. Since all individual object coverage spheres are of radius $\frac{\tau}{2}$ and all objects in a cluster are within distance β, the total volume covered by all objects inside a cluster can never exceed the volume of a sphere of diameter $\tau+\beta$, no matter how many objects we put into the cluster. Figure 2 depicts that in a two dimensional space with $\tau = \beta$. Since coverage grows with volume, which grows with sphere diameter, raised to the power of the number of dimensions, in an n-dimensional space, the cluster can cover a volume that is at most $\left(\frac{\tau+\beta}{\tau}\right)^n$ the volume covered by a single object. In other words:

$$\frac{CC_\infty}{CC_1} = \left(\frac{\tau+\beta}{\tau}\right)^n \tag{3}$$

From Equation (2) and Equation (3), we get:

$$\begin{aligned}\left(\frac{\tau+\beta}{\tau}\right)^n &= 1+\lambda+\lambda^2+\cdots\infty \\ &= \frac{1}{1-\lambda}\end{aligned} \tag{4}$$

From which:

$$\lambda = 1-\left(\frac{\tau}{\tau+\beta}\right)^n \tag{5}$$

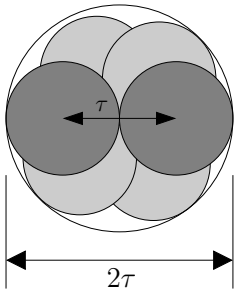


Fig. 2. Maximum possible coverage by a cluster.

Now, we can easily extend the notion of coverage to the entire object collection. Let collection X contain l clusters. Assuming clusters themselves are far enough apart from one another, the total coverage of all clusters is simply the sum of coverage of individual clusters. Let $s(c)$ be the size of cluster c. Therefore, the total coverage, that is, diversity, $\Psi(X)$ of collection X, is estimated by:

$$\Psi(X) = \sum_{c=1}^{l}\sum_{i=0}^{s(c)-1}\lambda^i = \sum_{c=1}^{l}\frac{1-\lambda^{s(c)}}{1-\lambda} \tag{6}$$

What remains is to show how the value of τ and β are chosen. First, since objects more than distance τ apart are considered independent, it is useful to use the same threshold for clustering as well. In other words, we set $\beta = \tau$. Arguably, we want clusters to be formed among similar looking pictures originated from the same geographic area. Pictures from different locations, even if they look similar, should fall into different clusters. According to our definition of distance between pictures (Equation 1), we need to set $\tau < 1$. Now the question is what visual distance makes two pictures look alike. This calls for experiments on our picture dataset. In evaluation, we show that distance less than 0.35 happen to be a good threshold. We therefore choose $\tau = 0.35$.

Each node tries to maximize $\Psi(X)$ as it maintains its picture collection in order to hold as many diverse pictures as possible subject storage constraints. But, not all pictures are equally relevant to the end collection. Some are less representative, hence outliers, which need to be eliminated. Below, we describe how outliers are identified and handled.

C. Outlier Resilient Diversity Maximization

It turns out that clustering offers an elegant way of separating the concern of outlier detection from the concern of diversity maximization. Intuitively, by assigning appropriate *relevance weights* to clusters, we can first get rid of low-ranked clusters (the outliers) to address relevance, then collect objects from the remaining clusters, thereby maximizing diversity, as per Equation (6), for only non-outlier clusters. In that sense, relevance weights are binary; a cluster is either an outlier or not. In the following, we explore the notion of outliers and relevance weights more closely.

1) Outliers versus Rare Items: It is good to remind the reader at this point that an explicit design decision we make (for the sake of efficiency) is to refrain from techniques that rely on understanding picture semantics in order to determine relevance. Short of having such an understanding, we can only approximately *estimate* relevance, which we do from the behavior of data collection agents themselves. Presumably, they are motivated to collect relevant information. Hence, if more sources report an observation, it is more likely that the observation is relevant. With that in mind, outlier detection may seem very easy. For example, singleton clusters (i.e., those that have only one member) can be treated as outliers. This approach, however, is not always appropriate. Sometimes items may be isolated not because they are irrelevant and do not generate interest, but rather because they are in the vicinity of only very few observers. If there were more people in their vicinity, more pictures may have been taken of them. Hence, some consideration to the level of isolation of the *location* of pictures needs to be made in outlier determination. Intuitively, a scene should be considered an outlier not only because it is different but because others who are present at the scene are not taking pictures of it. This motivates our definition of relevance weights.

To define outliers, we borrow a terminology from the data mining community, called *spatial outliers*. Due to Shenkhar *et al.* [2], *a spatial outlier is a spatially referenced object whose non-spatial attribute values are significantly different from those of other spatially referenced objects in its spatial neighborhood*. Correspondingly in our context, a picture is treated as an outlier, if it is geographically collocated with a popular picture set, but is *visually* significantly different from the group. For example, many users took a picture of a damaged house in a certain area, but one of them took a picture of something else which is different than the damaged building, while remaining geographically nearby. This picture

would then be treated as an outlier, since it somehow did not trigger the curiosity of the other individuals in the same area. In contrast, if an isolated picture is reported from a location and no other pictures are taken at the same location, then it is not treated as an outlier because we do not have enough evidence to say it is irrelevant. Instead, we regard it as a rare item that simply has not been found by many observers. With that in mind, we introduce our relevance score that measures the relevance of an item consistently with the above definition.

2) Relevance Weights of Clusters: Relevance weight of a cluster is computed as the fraction of pictures that the cluster represents compared to the total number of pictures that are originated in the *same* geographic area. A cluster represents *all* similar pictures (known to the node) from the same location. The number of all these pictures is called the *estimated size* of the cluster. If the size of a cluster is significantly smaller than the same sizes of other clusters in the same geographic area, then the cluster is likely to be an outlier. It indicates that not many sources were interested in recording that observation compared to others happening in the same location. Consideration of all objects known to the node, rather than only locally stored objects is important, because it allows different nodes, particularly between two communicating nodes, to agree on what they treat as outliers. This information is easy to collect via gossip among nodes. We revisit this issue when we describe our object transfer protocol in the subsequent section.

We use the standard z-statistic to determine outliers. We compute z-score of a cluster, denoted as $z(c)$, as follows:

$$z(c) = \frac{es(c) - \overline{es}}{S/\sqrt{m}} \tag{7}$$

where $es(c)$ is the estimated size of cluster c, $\overline{es}$ is the average estimated size of m geo-collocated clusters around c and S is the standard deviation of those sizes. A cluster is treated as an outlier if its estimated size, $es(c)$, is very small and $z(c) < \epsilon$, for some threshold $\epsilon < 0$. The value of ϵ affects the accuracy of detecting outliers. Smaller ϵ values can lead to false positives (outliers are not detected) and larger ϵ leads to false negatives (others are detected as outliers), and both are detrimental to the end collection. In evaluation, we show the sensitivity of ϵ on outlier detection.

III. Outlier Resilient Diversity-aware Ranking of Pictures

The main contribution of our scheme lies in implementing diversity maximization and outlier elimination as a content prioritization scheme that decides (i) the order in which objects need to be dropped on a node when storage is exceeded, and (ii) the order in which two nodes exchange content, when a connection between them is established. Objects need to be clustered as they arrive at a node. We describe the clustering process followed by the two prioritization schemes.

A. Online Clustering of Pictures

We use an online agglomerative clustering technique, proposed in [3], which incrementally adds new objects to existing clusters (as well as creates new clusters and splits earlier ones). We know that within a cluster all objects are within distance τ from one another. In that, the distance from the new object to all earlier objects need to be computed, which is somewhat costly to perform per arriving object. Instead, each cluster designates a representative object, called *centroid* object, and the distance to the centroid object is computed. If this distance is smaller than $\tau/2$, the newly object is put to the corresponding cluster. If there are multiple such clusters, it is assigned to the nearest one. If there is none, the object itself becomes a new cluster.

For a cluster c, the centroid object is denoted by $\mu(c)$. For each object x in the cluster, we define $\delta(x)$ as its average distance from other objects in the cluster (i.e., $\delta(x) = \frac{1}{s(c)} \sum_{y \in c} d(x, y)$). The object with the smallest δ is chosen as the centroid object. We also sort objects inside a cluster in the ascending order of their δ's so that the centroid object becomes the highest ranked one followed by others (ties are broken arbitrarily). Let $r(x)$ denote the rank of object x in its cluster. Obviously, $r(\mu(c)) = 0$ and $0 \leq r(x) < s(c)$. Ranking is used when objects from a cluster are chosen one after another.

As objects are added and deleted from clusters, some objects may violate the clustering rule (distance becomes greater than τ). This requires some reshuffling among clusters once in a while. This re-clustering operation is, however, costly in terms of computations. In ORPNet, this operation is executed *offline* when nodes are not in communication with another node.

B. Prioritized Dropping of Pictures

When the storage of a node becomes full, some earlier stored objects need to be dropped. While dropping objects, the dropping policy tries to preserve the diversity of the collection as much as possible, while also being resilient to outliers.

All clusters are divided into outlier and non-outlier clusters. As argued earlier, clusters with smaller $z(c)$ are most likely to be outliers and hence are treated as such. The order for picture dropping is computed as follows. First, the lowest ranked outlier cluster is found and the lowest ranked object is dropped from it. This continues until no outlier clusters remain. After outliers are eliminated, the algorithm switches to improving diversity, which requires maximizing $\psi(X)$ for non-outlier objects. From Equation (6), we have:

$$\begin{aligned} \Psi(X) &= \sum_{c=1}^{l} \sum_{i=0}^{s(c)-1} \lambda^i \\ &= \sum_{c=1}^{l} \sum_{x \in c} \lambda^{r(x)} \\ &= \sum_{x \in X} \lambda^{r(x)} \end{aligned} \tag{8}$$

That means, the object with the least $\lambda^{r(x)}$ value, i.e, the largest $r(x)$, should be dropped first, because it causes the least amount of decrement to $\Psi(X)$. In other words, the lowest ranked object from the largest cluster is dropped. Algorithm 1 gives the pseudo-code of computing the drop order. The worst

case running time is in the order of the number of clusters (finding the largest cluster).

More appropriately, a sorted list of objects can be maintained in the ascending order of their $r(x)$'s, and objects are dropped from the tail of this list whenever required. When nodes drop objects, they drop their object content only (data payload), but store their metadata (feature vectors) for further use, such as outlier detection. Assuming feature vectors are very small compared to actual content, it does not add significant storage overload at each node.

Algorithm 1 get-next-picture-to-drop()

```
Let C be the set of clusters at the node
Let x be the object that would be dropped next
Find cluster c+ with the smallest z(c)
if z(c+) < 0 then
   /* c+ is an outlier */
   x = arg min_{x∈c+} λ^{r(x)}
else
   Find the largest non-outlier cluster, c+, in C
   x = arg min_{x∈c+} λ^{r(x)}
end if
return x
```

C. Prioritized Transfer of Pictures

When two nodes establish a connection, they "sync" their content. In a flooding protocol, this would be achieved by exchanging all pictures that one node has but the other does not, such that both end up with the same set of pictures after the exchange. This, however, would be wasteful in resource consumption. Instead, we aim to exchange only representative content, suppressing both redundancy and outliers.

Each node maintains a list of meta information of all pictures, it ever encountered or stored. At the beginning of a connection, nodes exchange this list and update the estimated sizes and z-scores of their respective clusters, as described earlier. Based on z-scores, only non-outlier objects are considered first for transferring onto the peer node. Each one of the two nodes then determines the order at which it should be transferring objects so that the diversity at the other end is maximized.

Let node A meet B and A be the one who is taking the transfer decision. The same happens at B. At first, the centroid objects from those clusters of A are sent that would create new clusters at B. This is because this would cause the highest increment in B's diversity. After that, B now has all clusters that A has. Next, for each cluster c at A, a corresponding cluster at B, denoted by $g(c)$, is identified to which objects from c will be joining (i.e., distance $< \tau$). If an object from c is sent to B, it would increase B's diversity by $\Delta(c) = \lambda^{s(g(c))}$. But the next successive objects from the same cluster would have declining gain, each time multiplied by λ with the earlier sent object (ordered by their ranks). So, the diversity increment at B due to the transfer of an individual object, x, from A is: $\Delta(x) = \lambda^{s(g(c))+r(x)}$. Once Δ's for all objects are computed, A transfers objects in the descending order of their $\Delta(x)$'s. If pictures have large variation is their sizes, for best utilization of transfer opportunity, the value can be normalized by the size of the picture. Once all non-outlier clusters are considered, outlier clusters are considered, if transfer opportunity still allows sending more.

Algorithm 2 shows the transfer routine. Finding $g(c)$ for each cluster requires $O(l_B)$ computations (B has l_B clusters), so a total of $O(l_A l_B)$ computations. Computing $\Delta(x)$ per object then takes an iteration over the entire collection. So, the total running time of Algorithm 2 is $O(n_A + l_A l_B) \approx O(n + l^2)$ for a collection of n pictures with l clusters.

One last concern is the storage of metadata. Recall that each node attempts to store metadata of all pictures that it comes to know from other nodes, even though it may not store them all (it stores only a representative subset). When the number of pictures generated in the network becomes high, the volume of these metadata also rises. This may lead to an extra overhead of exchanging them. In order to reduce the metadata volume, we partition all these metadata into smaller clusters based on their distances, just like stored objects. While exchanging metadata, nodes then send only one representative item per cluster, called *pivot*, with the associated object IDs in that cluster. Pivots efficiently summarize the metadata of all objects known to a node. When pivots are exchanged between two nodes, both of the nodes check whether they have the same set of pivots (by measuring distances between them). If not, they update their current metadata clusters and pivots accordingly. Recent results, such as [4] that used bloom filters, can also be investigated in this regard.

Algorithm 2 compute-transfer-order(Contact c)

```
Let c be a meeting between node A and B
Let A and B be the set of clusters at A and B

for all c ∈ A and c is not an outlier do
   Compute g(c) = arg min_{b∈B} d(c, b), where d(c, b) < τ
   Set s(g(c)) = 0, if g(c) does not exist
   for all x ∈ c do
      Δ(x) = λ^{s(g(c))+r(x)}
   end for
end for
Transfer objects in the descending order of Δ(x)
```

IV. Evaluation

We simulate ORPNet in the ONE [5] simulator for a post-disaster rescue mission. In this setting, the underlying network is DTN, where the simulated nodes (mainly rescue workers and volunteers) carry cameras, visit places, shoot pictures of interest, and exchange these pictures, when they meet, ultimately to pass them to a central command station. Most of the simulation setting is inspired from PhotoNet [1]. PhotoNet uses a scheme that tries to minimize redundant delivery of pictures by choosing pictures that are most diverse. This leads to serious vulnerabilities to outliers. We compare our major results with PhotoNet to demonstrate that the elimination of outliers is important for any diversity-aware content delivery service. We also implemented ORPNet on Android phones. Since we cannot produce a large scale physical instance of

a network envisioned by our rescue mission, we limit our experiments with phones only to show various timing results performed on devices. We defer building a fully deployed service and conducting experiments involving real humans as future work.

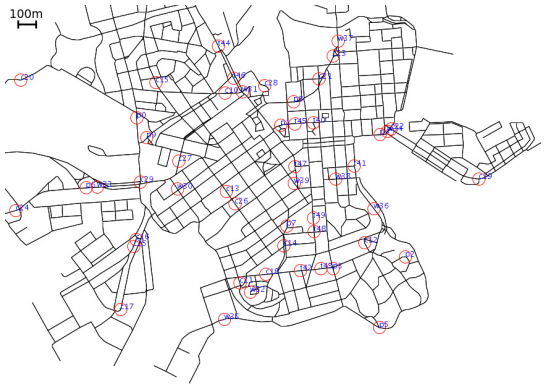


Fig. 3. The city map used in the ONE simulator.

A. Simulation Environment

We use the Post-Disaster Mobility (PDM) model [6] to simulate a participatory sensing mission in a hypothetical town (Figure 3). PDM uses a map file to generate streets in the simulated area, such that mobile agents use streets for moving between destination points. PDM randomly locates a couple of neighborhoods with houses and puts service stations, such as rescue centers, relief camps, and police stations (specified in a configuration file) in the map. It also places a central command station located far away from the neighborhoods. Four types of mobile agents are deployed: (i) vehicles that move back and forth between service stations, (ii) rescue workers and volunteers (mainly responsible for taking pictures) who roam around inside a given neighborhood and occasionally report to the nearby service stations, (iii) regular police patrols that visit neighborhoods, and (iv) a few data mules that commute between the command station and the different service stations in distant neighborhoods. We create 5 neighborhoods, 10-15 service stations, nearly 100 volunteers and 5 data mules.

1) Generating Scenes and Pictures: We generate 25-50 scenes or events and associate each with a pool of pre-taken similar-looking *real* pictures, a total of nearly 1000 pictures. These pictures are actually of different landmarks/scenes in our campus taken at different angles and zoom levels. Different scenes have different observation popularity resulting in varying number of similar pictures per event (following a Zipf law distribution). In simulation, nodes visit event locations and take one of the of pictures at random from the pre-assigned pool. The picture is then tagged with the location of the node. Each node is equipped with a limited storage of 5-10 MB. This storage is obviously smaller than what a device could really have (in GBs). As argued in [1], this is to match the scaled-down size of our network, compared to the real size of tens of thousands of nodes (participants in a large city) and hundred of neighborhoods. Furthermore, we consider a network setting with a very poor delivery ratio (only 20-30% pictures are delivered) so that the diversity of picture collection really matters. We use a popular DTN routing protocol, Prophet [7], as the base packet forwarding protocol for the network and override Prophet's default dropping and transfer ordering as suggested in our scheme to implement ORPNet on top of Prophet.

2) Injecting Outliers: In our simulation, outliers are generated that are pictures of random scenes other than those mentioned above. They are geographically collocated with other pictures, but visually different from the rest of the pool. In each event pool, we artificially inject some non-relevant pictures. Figure 4 demonstrates the scene "shrubs" with an outlier. The total number of outliers is controlled by a parameter, called *outlier ratio*, which specifies what fraction of pictures could be outliers. Unless otherwise stated, we use 15-20% outliers.

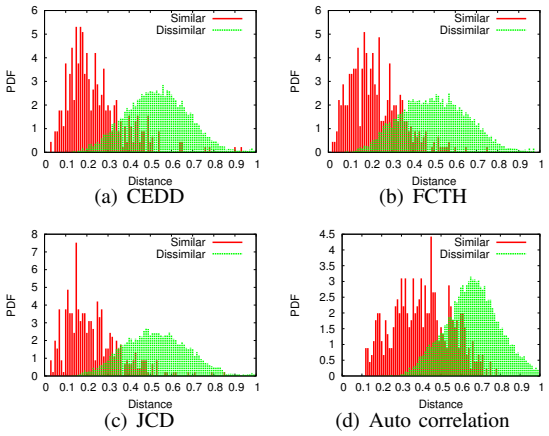
Fig. 4. Pictures pertaining to a scene "shrubs"; the rightmost one is an outlier in this pool.

3) Similarity Distance between Pictures: For computing distances between pictures based on visual similarity, we used existing CBIR (Content Based Image Retrieval) techniques. We used an open source lightweight library LIRe [8] (http://www.semanticmetadata.net/) with four visual features, namely CEDD [9] (Color and Edge Directivity Descriptor) , FCTH [10] (Fuzzy Color and Texture Histogram), Auto Color Correlogram [11], JCD [12] (Joint Composite Descriptor). In all cases, the feature is represented as global image descriptor vectors, which are mainly histograms of one or more particular interest in a very compact representation (CEDD and FCTH vectors are of 54 and 72 bytes per image respectively). Given two vectors, the distance is computed as Tanimoto coefficient [13] defined as $\frac{x^T y}{x^T x + y^T y - x^T y}$. Figure 5 shows the probability density of similarity distance values computed between pairs of similar-looking and pairs of dissimilar pictures. We see that the distribution is multi-modal that enables us to separate similar pictures from dissimilar ones. We use JCD features for our experiments and choose the clustering threshold, τ=0.35. Due to the Java based implementation, we easily ported the library to Android phones.

B. Simulation Results

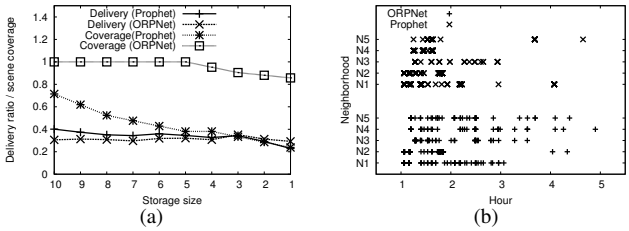
As a performance metric, we are interested in measuring what fraction of relevant scenes got delivered to the command station. We refer to this metric as *scene coverage* or simply coverage in this section. We treat a scene as delivered or *covered*, if at least one non-outlier picture of that scene is reported to the command station. Delivery of outliers does not contribute to valid coverage. ORPNet aims at preventing outliers from being propagated through the network and attempts to drop them before they reach the command station.

To appreciate the significance of diversity in picture collection, we begin with showing results for content-agnostic



(a) CEDD (b) FCTH (c) JCD (d) Auto correlation

Fig. 5. Probability density of visual distances for between pictures.



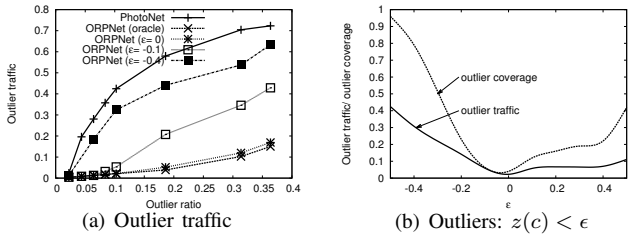
(a) (b)

Fig. 6. (a) Delivery ratio and coverage of Prophet and ORPNet, (b) Delivery across neighborhoods.

forwarding scheme (Prophet) versus our diversity-aware forwarding scheme (ORPNet). Figure 6(a) shows the delivery ratio of pictures (fraction of total pictures delivered irrespective of scenes/outliers) as well as coverage of scenes in both protocols. We see that both schemes produce nearly the same delivery ratio (below 40%), but the coverage is very poor for Prophet. The reason is obvious. Prophet being unaware of similarity among pictures stores different pictures merely by chance, whereas diversity-aware ORPNet suppresses similar stuffs and results in higher coverage even at very scarce resources. Figure 6(b) shows the delivery of pictures (each point is a picture) from different neighborhoods against time. We observe that ORPNet does a better job of distributing pictures across neighborhoods than Prophet. The standard deviation of the number of pictures per neighborhood is around 30 and 65 for ORPNet and Prophet respectively.

Figure 7 shows the coverage of ORPNet and PhotoNet when transfer bandwidth and storage capacity at each node are varied. We use separate coverage for valid scenes and outliers. Outlier coverage means the fraction of delivered outliers out of what generated. We see that ORPNet covers almost all scenes, while delivering only a smaller fraction (around 20%) of outliers. On the contrary, PhotoNet results in almost the opposite: it delivers almost all outliers leaving legitimate scenes behind (valid coverage falls below 70%). This is because PhotoNet favors different pictures which weighs outliers highly. This deprives a whole bunch of legitimate valid scenes from being reported to the base. Figure 7(c) shows the delivery of outliers as outlier ratio increases. The delivery of a few outliers in ORPNet is mainly due to the error in identifying outlier clusters and the inclusion of outliers in valid clusters.

It would be interesting to see how robust ORPNet is in detecting outliers. We trace all transfers of pictures across



(a) Outlier traffic (b) Outliers: $z(c) < \epsilon$

Fig. 8. (a) Ratio of outlier traffic with varying outlier ratio, (b) Effect of ϵ on robustness to outlier detection.

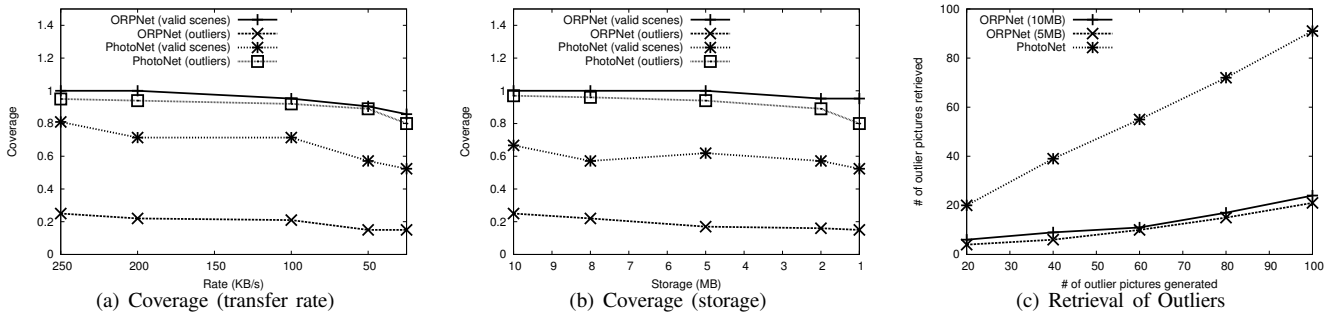
the nodes in the simulation and find the fraction of transfers attributed to outliers. We term this as the *ratio of outlier traffic*. ORPNet intends to suppress outliers while transferring pictures across nodes. Figure 8(a) shows the ratio of outlier traffic for ORPNet and PhotoNet. In PhotoNet, outlier traffic is very high, even dominates valid traffic at some point (beyond 20% outlier ratio). On the contrary, ORPNet is robust to outliers and keeps outlier traffic low. This robustness, however, depends on the accuracy of detecting outliers. Recall that outliers are detected based on estimated sizes and z-scores of clusters, namely $z(c) < \epsilon$ implies an outlier. We run experiments on several values of ϵ and observe that outlier detection is sensitive to ϵ. We also show results for an "oracle" run that allow nodes to know *truly* which objects are outliers. We see that $\epsilon = 0$ produces the lowest outlier traffic. Figure 8(b) shows outlier traffic at varying values of ϵ. It depicts that as ϵ deviates around 0, outlier detection becomes weak and both outlier traffic and outlier coverage rise. This is because smaller ϵ leads to false positives and larger ϵ leads to false negatives. Again, when outliers are misclassified, they are given higher priority than others at the time of exchanging pictures (generally, because of their smaller cluster size). That's why false positives lead to higher outlier coverage than false negatives. In all of our experiments, we use $\epsilon = 0$.

C. Phone-based Implementation

We implemented ORPNet on Android phones (Google Nexus S) and evaluated results that are crucial for running the service on phones. These are mostly timing values for various computations invoked by ORPNet. We also compare the results with PhotoNet.

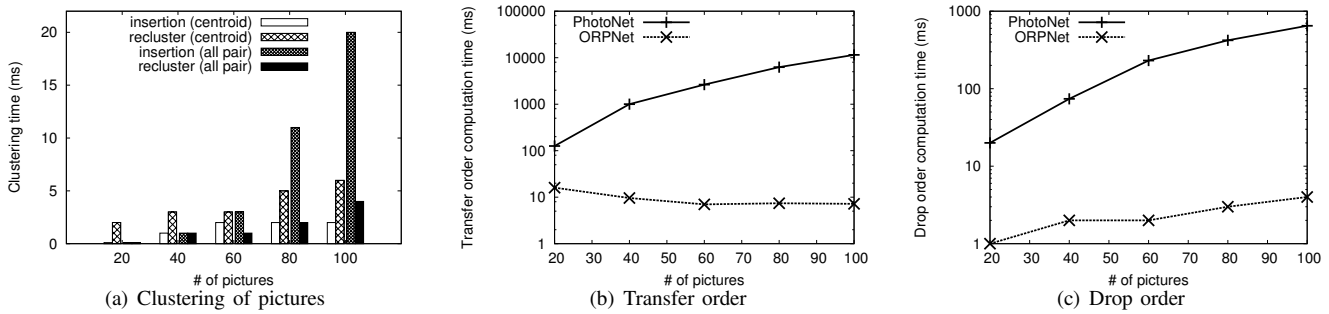
We generate timing results for several computational cases. The first timing value we measure is the clustering time, the time taken by a new picture when it is inserted into one of the clusters of the collection. Recall that in our scheme the clustering operation is online, that is, the insertion is executed immediately upon the arrival of the picture. Also, clusters are reorganized once in a while that requires another computation. Moreover, recall that, in our scheme, when an object is added to an existing cluster, the distance is measured only to the centroid object of the cluster. We refer to this as the centroid scheme. In ideal approach, distances to all objects in a cluster need to be measured (all-pair scheme). We produce the timing results for these two cases. In all cases, we produce the median values over 100 runs.

Figure 9(a) shows delays for clustering and re-clustering. We see that for inserting an item, centroid scheme results in smaller delay (less than 2ms per 100 pictures), whereas



(a) Coverage (transfer rate) (b) Coverage (storage) (c) Retrieval of Outliers

Fig. 7. Scene coverage (a), (b) at varying storage size and transfer rate. (c) Retrieval of outliers at varying storage.



(a) Clustering of pictures (b) Transfer order (c) Drop order

Fig. 9. Time for (a) adding a new item and re-clustering, (b) computing transfer order, (c) determining the next picture to drop.

all-pair scheme produces larger delay (nearly 20ms). On the contrary, re-clustering time is longer for centroid scheme compared to the all-pair scheme. This is because centroid distance introduces more distortions into clusters, which in turn requires more shuffles during re-clustering. In all-pair scheme, however, clusters are more accurate and they need less shuffle afterward, but the cost for each insertion is high to begin with. Since ORPNet applies online clustering, it uses centroid scheme and defers the costly reshuffle operation to offline, when nodes are not in communication with others (DTN-style communication allows that).

Next, we observe delays for computing transfer order and dropping order of pictures. Unlike OPRNet, which effectively ranks clusters ($O(n+l^2)$ computations), PhotoNet does the same for each picture by measuring pair-wise distances ($O(n^2)$), where l and n is the number of pictures and clusters respectively. It turns out that checking all pictures one by one is very expensive. Figure 9(b) shows delays of computing transfer order of pictures. We see that the time for computing transfer order in ORPNet does not change much as the number of pictures grows, whereas for PhotoNet it grows constantly. This is because the number of clusters changes far slowly than the number of objects. Figure 9(c) presents the same results for dropping pictures. ORPNet takes magnitude order of smaller time to determine the next picture to drop compared to PhotoNet, again due to clustering.

V. Related Work

Camera sensor networks have received great attention over recent years. This is due to rapid advances in camera technologies enabling camera embedded sensor platforms, and, more specifically, due to the inevitable availability of cameras in almost all mobile phones these days. Camera based sensor platforms focused image recognition and activity recognition ([14]) that have applications to surveillance [15], habitat monitoring [16], security systems [17], and assisted living [18]. As cameras are becoming more ubiquitous in recent years, a set of participatory applications, in the form of "urban image sensing", are also emerged, such as microblogging ([19]) and telemedicine (e.g. documenting diets [20]). Soro *et al.* [21] make a comprehensive survey of recent camera sensor networks and their applications.

There have been works on mobile phone-based data collection and retrieval system. Works proposed in [22] and [23] use 3G networks on mobile phones, vehicle based DTNs, and available nearby WiFi access points to transfer HTML pages against user queries. Cartel project [24] develops a mobile sensing system, in the form of a Web portal service, where vehicles sense, record and submit data to a central database. A distributed image retrieval service on a sensor platform is proposed in [25].

Information retrieval (IR) community has worked at length on information retrieval system that considers redundancy and novelty of retrieved information ([26], [27], [28], [29]). In most cases, these works are for text documents. Content-based image retrieval systems ([30], [31]) work for querying images from a set of given pool of images. These services mainly index an initial collection of pictures at the server and then return images that are visually very close to the queried image. Although we borrow techniques from literature for computing image similarity, our contribution lies in in-network implementation of the service. In that, most challenges arise due to redundancy injected by similar pictures residing in different nodes and the existence of outliers scattered over the network.

For evaluation purpose, we run the service over a DTN. DTN routing protocols usually replicate messages onto other nodes to increase message delivery. A few notable routing protocols are, Prophet [7], MaxProp [32], Delegation routing [33], RAPID [34], Spray and Wait [35], EBR [36] and IC-Routing [37]. Our scheme of content synchronization is, however, routing protocol agnostic, in that, it can be used in

association with any routing protocol underneath.

VI. Conclusion and Future Work

In this paper, we developed a scheme for delivering a representative subset of pictures from a larger pool in a participatory camera sensor network, where many pictures may be redundant and some may not be relevant. Heuristics were developed that balance outlier elimination and diversity maximization to achieve better coverage with the lowest number of pictures. The service was shown to offer a much higher coverage compared to previous work that focused on diversity maximization alone without outlier elimination. This is because outliers, by their very nature, are diverse, and hence (incorrectly) favored by diversity-maximizing algorithms. Future work on this topic will consider integration of our mechanisms with network caching in applications where content is requested by multiple sinks. Another direction is to extend the scheme with an estimation of source reliability, such that content prioritization is affected by reliability estimates. For example, pictures sent from unique locations by unreliable sources need not be considered. An experimental evaluation of the service deployed on Android phones is another direction of future work.

Acknowledgment

Research reported in this paper was sponsored by ONR grant N00014-10-1-0172, DTRA grant HDTRA-1-10-1-0120, NSF grants CNS 06-26825 and 1040380, and the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] M. Y. Uddin, H. Wang, F. Saremi, G.-J. Qi, T. Abdelzaher, and T. Huang, "Photonet: a similarity-aware picture delivery service for situation awareness," in *Proc. of RTSS*, 2011.

[2] S. Shenkhar, C.-T. Lu, and P. Zhang, "Detecting graph-based spatial outliers: algorithms and applications (a summary results)," in *Proc. of SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[3] I. D. Guedalia, M. London, and M. Werman, "An on-line agglomerative clustering method for nonstationary data," *Neural Comput.*, vol. 11, no. 2, pp. 521–540, 1999.

[4] D. Eppstein, M. T. Goodrich, F. Uyeda, and G. Varghese, "What's the difference? efficient set reconciliation without prior context," in *Proc. of SIGCOMM*, 2011.

[5] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE simulator for DTN protocol evaluation," in *Proc. of SIMUTools*, 2009.

[6] M. Y. S. Uddin, D. Nicol, T. Abdelzaher, and R. Kravets, "A post-disaster mobility model for delay-tolerant networking," in *Proc. of WSC*, December 2009.

[7] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, 2003.

[8] L. Mathias and S. A. Chatzichristofis, "Lire: Lucene image retrieval an extensible java cbir library," in *Proc. of MM*, 2008.

[9] S. A. Chatzichristofis and Y. S. Boutalis, "CEDD: Color and edge directivity descriptor. a compact descriptor for image indexing and retrieval," in *Proc. of ICVS*, 2008.

[10] ——, "FCTH: Fuzzy color and texture histogram a low level feature for accurate image retrieval," in *Proc. of WIAMIS*, 2008.

[11] J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih, "Image indexing using color correlograms," in *Proc. of CVPR*, 1997.

[12] S. Chatzichristofis, M. Lux, and Y. Boutalis, "Selection of the proper compact composite descriptor for improving content based image retrieval," in *Proc. of SPPRA*, 2009.

[13] Z. Chi, H. Yan, and T. Pham, "Fuzzy algorithms: With applications to image processing and pattern recognition," *Advance in fuzzy systems Applications and theory*, vol. 10, 1996.

[14] D. Lymberopoulos, A. S. Ogale, A. Savvides, and Y. Aloimonos, "A sensory grammar for inferring behaviors in sensor networks," in *Proc. of IPSN*, 2006.

[15] R. Goshorn, J. Goshorn, D. Goshorn, and H. Aghajan, "Architecture for cluster-based automated surveillance network for detecting and tracking multiple persons," in *Proc. of ICDSC*, 2007.

[16] M. Rahimi, R. Baer, O. I. Iroezi, J. Garcia, J. Warrior, D. Estrin, and M. Srivastava., "Cyclops: In situ image sensing and interpretation in wireless sensor networks," in *Proc. of Sensys*, 2005.

[17] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan, "Mesheye: A hybrid resolution smart camera mote for applications in distributed intelligent surveillance," in *Proc of IPSN-SPOTS*, 2007.

[18] H. Aghajan, J. Augusto, C. Wu, P. McCullagh, and J. Walkden, "Distributed vision-based accident management for assisted living," in *Proc. of ICOST*, 2007.

[19] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt, "Micro-blog: Sharing and querying content through mobile phones and social participation," in *Proc. of MobiSys*, 2008.

[20] S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, and M. Hansen, "Image browsing, processing, and clustering for participatory sensing: Lessons from a dietsense prototype," in *Proc. of EmNets*, 2007.

[21] S. Soro and W. Heinzelman, "A survey of visual sensor networks," *Advances in Multimedia*, vol. 21, 2009.

[22] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "Enhancing interactive web applications in hybrid networks," in *Proc. of MobiCom*, 2008, pp. 70–80.

[23] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3G using WiFi," in *Proc of MobiSys*, 2010, pp. 209–222.

[24] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "Cartel: a distributed mobile sensor computing system," in *Proc of SenSys*, 2006.

[25] T. Yan, D. Ganesan, and R. Manmatha, "Distributed image search in camera sensor networks," in *Proc. of SenSys*, 2008, pp. 155–168.

[26] Y. Zhang, J. Callan, and T. Minka, "Novelty and redundancy detection in adaptive filtering," in *Proc of SIGIR*, 2002, pp. 81–88.

[27] C. Zhai and J. Lafferty, "A risk minimization framework for information retrieval," *Information Processing and Management*, vol. 42, pp. 31–55, 2006.

[28] H. Chen and D. R. Karger, "Less is more: Probabilistic models for retrieving fewer relevant documents," in *Proc of SIGIR*, 2006.

[29] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon, "Novelty and diversity in information retrieval evaluation," in *Proc. of SIGIR*, 2008, pp. 659–666.

[30] R. Datta, J. Li, and J. Z. Wang, "Content-based image retrieval: approaches and trends of the new age," in *Proc of MIR*, 2005, pp. 253–262.

[31] R. H. van Leuken, L. Garcia, X. Olivares, and R. van Zwol, "Visual diversification of image search results," in *Proceedings of the 18th international conference on World wide web*, 2009, pp. 341–350.

[32] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for vehicle-based disruption-tolerant networks," in *Proc. of INFOCOM)*, 2006.

[33] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot, "Delegation forwarding," in *Proc. of MobiHoc*, 2007.

[34] A. Balasubramanian, B. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem," in *Proc. of SIGCOMM*, 2007, pp. 373–384.

[35] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proc. of WDTN*, 2005, pp. 252–259.

[36] S. Nelson, M. Bakht, and R. Kravets, "Encounter-based routing in DTNs," in *Proc. of INFOCOM*, 2009.

[37] M. S. Uddin, H. Ahmadi, T. Abdelzaher, and R. Kravets, "A low-energy multicopy inter-contact routing protocol for disaster response networks," in *Proc. of SECON*, 2009.