25th IEEE International Symposium on
Robot and Human Interactive Communication (RO-MAN)
August 26-31, 2016. Columbia University, NY, USA

Look Where You're Going: Visual Interfaces for Robot Teleoperation

Jim Vaughan[1], Sven Kratz[2] and Don Kimber[3]

***Abstract*— Two related challenges with current teleoperated robotic systems are lack of peripheral vision and awareness, and difficulty or tedium of navigating through remote spaces. We address these challenges by providing an interface with a *focus plus context* (F+C) view of the robot location, and where the user can navigate simply by looking where they want to go, and clicking or drawing a path on the view to indicate the desired trajectory or destination. The F+C view provides an undistorted, perspectively correct central region surrounded by a wide field of view peripheral portion, and avoids the need for separate views. The navigation method is direct and intuitive in comparison to keyboard or joystick based navigation, which require the user to be in a control loop as the robot moves. Both the F+C views and the direct click navigation were evaluated in a preliminary user study.**

I. Introduction

A remotely controllable robot base with video transmission and reception capability theoretically allows a remote person to interact with an environment as a local person would, for example 'walking' through the hallways to see who is in their offices and available to talk. However, in practice, the experience of driving a robot by viewing video from an on-board camera is frustrating at best, and dangerous at worst. In particular, there are two distinct types of problem:

- It is very difficult for a user to steer a robot using video if there is high or inconsistent latency in the video feed.
- The "tunnel vision" that arises from using a standard perspective camera leads to a lack of awareness of the environment.

In this paper, we describe our efforts to address both of these problems, firstly with "reach through the screen navigation", in Section II, then with custom camera projections in Section III. We then describe a user study that evaluates both these technologies in Section IV.

II. Navigation

Simultaneous Localization and Mapping (SLAM) [1] is a widely accepted technique for determining the position of a robot. However, there is a significant investment involved in building the maps, for example it took three days for us to generate a map of our 2000 m^2 office and laboratory space using a Hokuyo laser scanner on a mobile robot. With localization, path planning and obstacle avoidance, which are available in ROS [2], autonomous navigation within an indoor environment is feasible. For other work, we have implemented a user-interface where a user can click on a map to instruct the robot where to move. However, in our current work, we have been operating robots without autonomous navigation, so that the remote user can experience a sense of being in the robot's location, and to understand how the process may be improved. A goal of the system is that it be deployable in new environments without any setup or training time, either for the system or its operators.

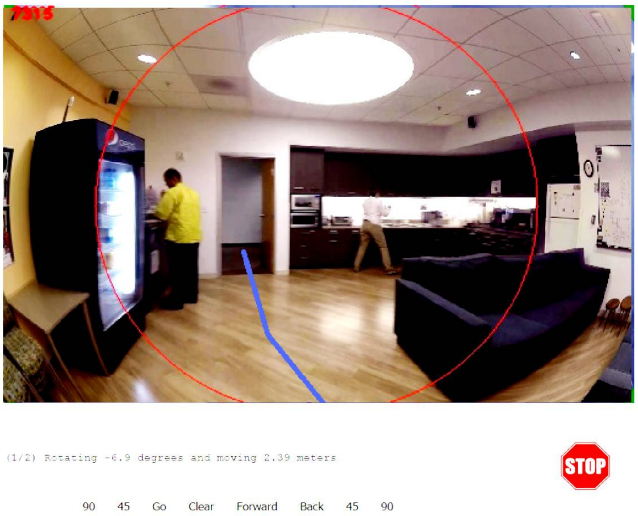


Fig. 1. Robot teleoperation interface with Focus+Context view in Through-the-Screen input mode with user-specified path.

Humans are good at analyzing a scene and determining which parts of the scenes are floor, walls, obstacles, etc. and planning a path through the environment in the scene. This can be done quickly, and with low cognitive effort. However, driving a robot via key or joystick input is challenging, particularly if there is latency in the video. Latency or drop outs in the video confuse the user and cause over-correction. This is most noticeable when driving down long, straight hallways.

Foote *et al* [3] proposed the metaphor of "reaching through the screen" to interact with the remote environment during a video conference. We applied that metaphor to the task of navigating a telepresence robot and developed the "reach through the screen" interface shown in Figure 1. In this interface, the user clicks points on the floor to specify a path for the robot to follow in terms of *waypoints*. The user specified path is shown in Figure 1 as a series of blue line segments. The user is therefore *directing* the robot, but the low-level control loop of following the path is handled by the computer on the robot using wheel odometry data. TIn our

[1]Jim Vaughan is with FX Palo Alto Laboratory, Palo Alto, CA, USA jimv@fxpal.com
[2]Sven Kratz is with FX Palo Alto Laboratory, Palo Alto, CA, USA kratz@fxpal.com
[3]Don Kimber is with FX Palo Alto Laboratory, Palo Alto, CA, USA kimber@fxpal.com

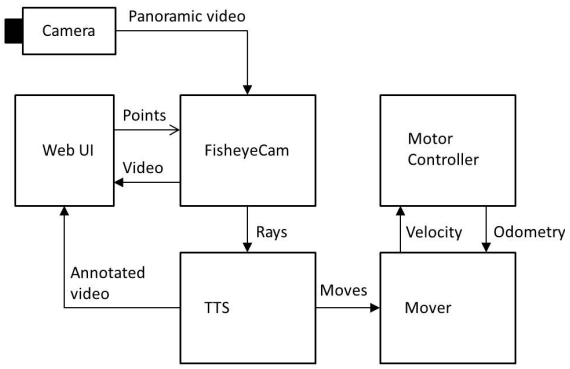


Fig. 2. System overview

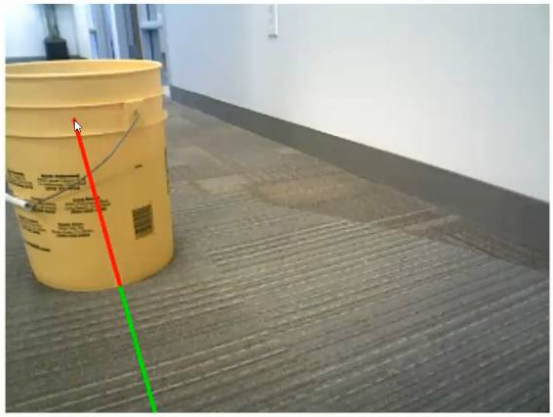


Fig. 3. Augmented video path: the red portion of the path shows the presence of an obstacle.

prototype system, the motion is decomposed into a series of rotation and linear segments. For each waypoint, the robot first rotates to be facing the desired direction and then drives straight towards it. The intention was that this would allow large distances to be covered with little attention required from the user.

A. Prior Work Related to Navigation

Others have developed robot teleoperation user interfaces with overlayed information. For example. Baker *et al* [4] proposed overlaying system information on the camera navigation view. In contrast to our work, the information they overlay is related to the robots systems and not related to navigation planning tasks. Maxwell *et al* [5] describe a system which allows users to point to a specific point in the camera navigation view with the mouse. Clicking on that point then makes the mobile robot autonomously navigate to that point. A similar approach is described by Coltin *et al* [6]. Our approach differs from these two approaches, in that we allow the user to specify a destination or a path, thereby leveraging the user's ability to easily understand a scene and navigate a path through it. Jung *et al* [7] describe a camera-based interface, where robot paths can be sketched. This work, however does not visualize any robot sensor information, nor does it provide any interactive feedback about the users planned path. Chronis *et al* [8] and "Sketch and Run" by Sakamoto et al. [9], propose robot control via sketch inputs. The main difference to our proposal is that their system uses a tablet PC with a top-down view of the robots environment. Sketch input is done within this abstract top-down representation of the robot and not on a live camera view.

B. Implementation Details

Our prototype system was built using the Robot Operating System (ROS) [2] framework. In a ROS system, the processing is performed by a number of *nodes*, which communicate through *topics*, which are named message channels. One of the nodes that is available with ROS, *rosbridge_websocket*, provides a websocket interface to ROS. We used this to build a web-based user interface in JavaScript. It was a design goal that the robot would be able to execute its tasks even if it lost connection to the network. To this end, the JavaScript interface merely sent commands to the robot and displayed their response.

An overview of the system is shown in Figure 2. In the web interface, the user clicks a number of points in the image to specify the robots path. For each pixel (x, y) clicked in the image, a ray can be determined. The intersection of this ray and the plane $z = 0$ formed by the floor determines the position on the floor where the robot should go. In the case of a perspective camera, the conversion from image points to rays is performed by multiplying by the camera's projection matrix. With the camera projections described in Section III, the conversion is more involved. For this reason, we integrated the image remapping system (*FisheyeCam*) with ROS, and had it subscribe to a topic containing a set of image topics and publish a topic of rays in response. These rays are received by the *TTS* node, which performs the transform from the camera's co-ordinate system to the robot's coordinate system and then computes the intersection points with the floor plane. Rays which are parallel to the floor intersect at infinity, and rays which point upwards intersect behind the robot. These interesect points are discarded. The valid intersects are used as a list of waypoints have been determined, an the TTS node generates a list of rotation and linear movement commands from them. These are used to drive the robot using a closed-loop system with the robot's odometry by the *mover* node.

Our initial prototype used an Asus Xtion Pro Red, Green, Blue and Depth (RGBD) camera. From the depth image, we were able to estimate the parameters of the most dominant plane using RANSAC, and assume this is the floor, and then uses them to determine the pose of the camera. This allowed the system to be self calibrating.

Having depth information available also allows for some rudimentary scene analysis. Once the parameters of the floor plane has been estimated, it is trivial to determine the parts of the scene that are not the floor. This information can be used to guide the user in several ways, for example by highlighting an infeasible path, or by actually preventing the

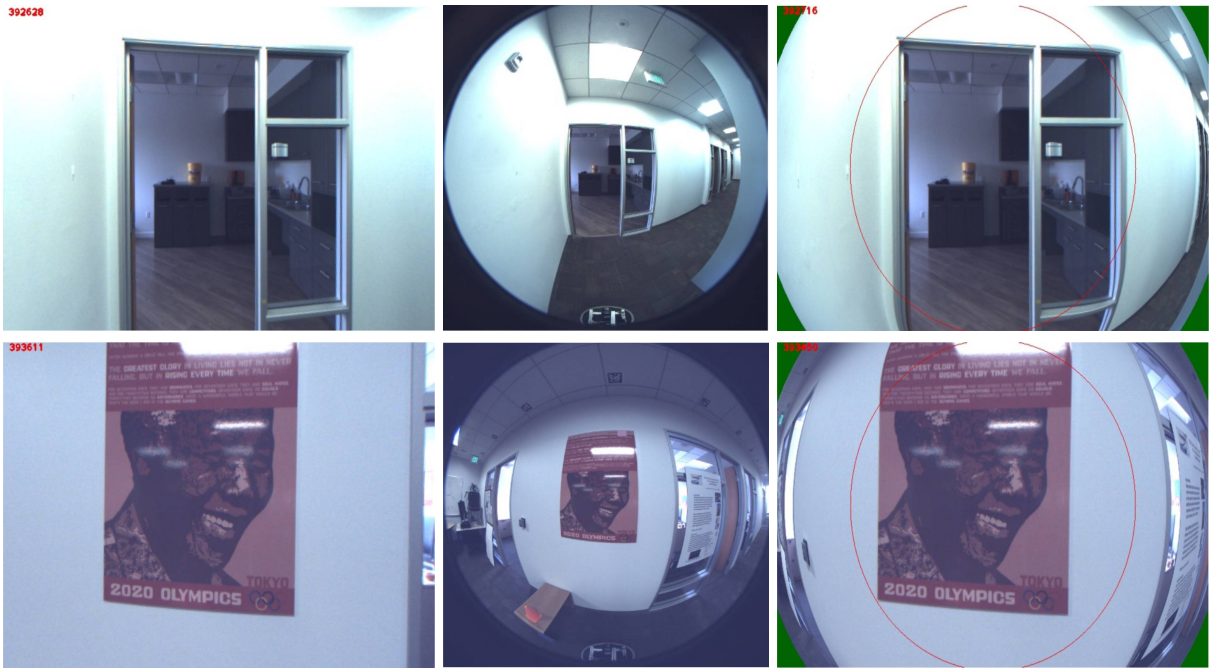


Fig. 4. Views: top row: a navigational view, bottom row: a close up view; left: perspective. center: fisheye. right: F+C

user from choosing such a path, either by "rubber banding" such that the user's mouse movements have less effect in certain areas, or by refusing to move. In our prototype, we have implemented two methods: a overlay that highlights all non-floor elements of the scene, and a colored path that shows obstacles, where path segments that traverse any obstacles are marked red . An example of this is shown in Figure 3.

In order to address all the problems with video-based teleoperation, we combined the reach-through the screen interface with a camera with a fisheye lens, so that it could be combined with the Focus and Context view described in Section III. Without depth image, calibration requires careful measurement of the camera's extrinsic parameters and testing of the correspondence between image points and waypoints in the robot's co-ordinate system. However, the extra information provided by the panoramic and focus and context views is useful in navigation. For example, in Figure 4, the hallway to the right is apparent in the last two images in the top row. During the user study described in Section IV all the images were captured from the same camera with panoramic lens, and the perspective camera images were produced from a virtual camera, so that they would all have similar contrast and response to lighting.

III. Focus and Context Views

Human vision provides us with the sense of a single comprehensive view of our surroundings. For a fixed position of each eye, we have a horizontal field of view of about 70 degrees [10], of which only a small central foveal region has "high resolution". The fovea sees only about the central two degrees of the visual field, with a "resolution" of about 31.46 arc seconds. It takes up about 1 percent of the retina, but over 50 percent of the visual cortex in the brain. Combining head and eye motion, we have a field of view of about 190 degrees [10]. Our brains integrate this so well that we are normally not aware of how low is our resolution outside of the small foveal region, nor of our need for saccades to produce an overall visual sense of the space, nor even of our head motions.

By contrast, when watching a view from a remote camera, we have consistent resolution within the view, and see nothing outside of the view. This gives us a sense of tunnel vision that leaves us feeling removed from the remote space. Even if the camera is steerable or is on a robot or remote vehicle that we can move around and turn, the effort to move the camera and make sense of the overall scene moves from the "perceptual" to "cognitive" level of mental processing. For highly immersive experiences, a user can be provided with a head-mounted-display having field of view (FOV) close to that of the eye, and a head tracker so that head motion provides essentially the same view as if they were at the remote location. Alternatively, the user may sit in a "CAVE" with video shown on the walls or other surrounding surface. For many applications however, these are not practical or are otherwise undesirable. Providing such views requires high bandwidth, low latency networking, more elaborate hardware, or requires the user to be wearing an unnatural device and essentially be disconnected from their own local environment. A user may prefer, for example, to be watching the view on a large display, in a web page, or on a tablet.

To address the problem of tunnel vision, and provide effortless peripheral vision in a simple one window display, we propose the use of Focus plus Context (F+C) views, motivated by the human foveal visual system. F+C views are

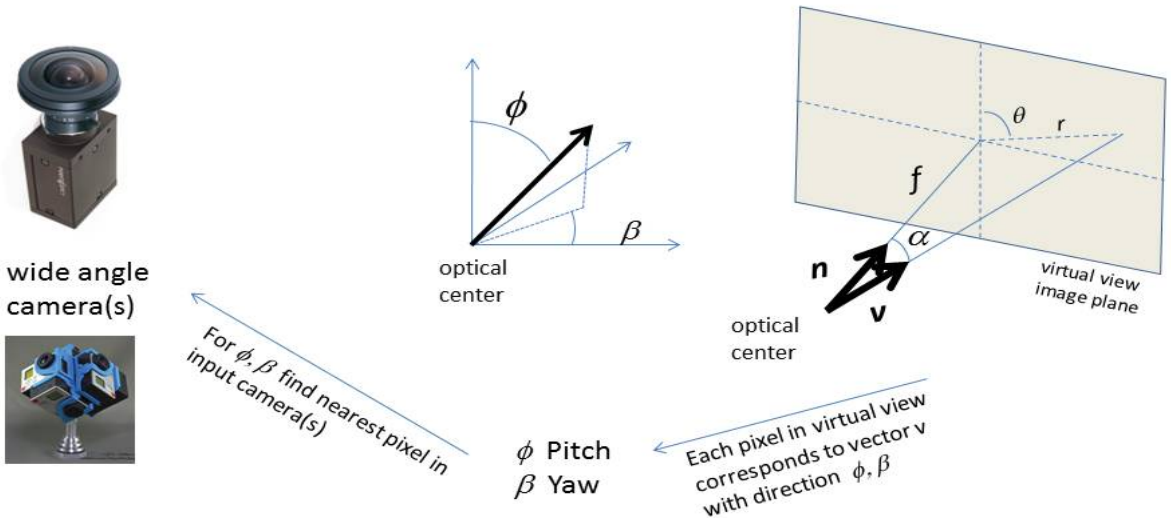


Fig. 5. Geometry of Focus+Context view generation.

closeup, detailed and undistorted in the middle, and show a large field of view in the periphery, but with a continuous transition that does not introduce false edges or corners. The class of F+C views we investigate can be thought of as having a fixed field of view perspective camera inside the focus region, with a rapidly growing field of view outside that.

A. Prior Work Related to Focus plus Context

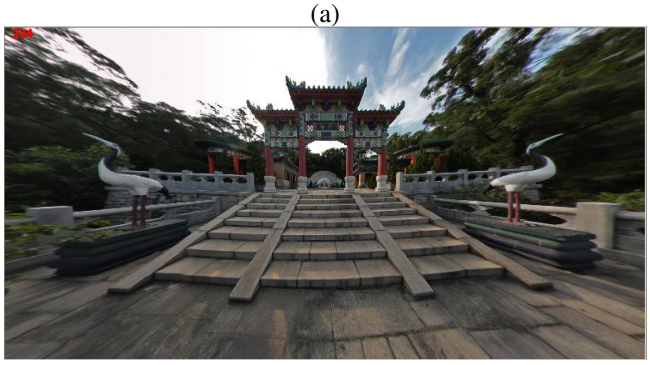
Focus plus context methods have been widely applied for information visualization purposes [11], [12], and has been used to create specialized viewers for images such as maps. For example Zhou *et al* [13] describe a system based on approximate conformal mappings a view of a map with one or more focus regions with preserving local shape in those regions. Trapp [14] describes a number of projection methods for viewing virtual scenes that could be considered F+C views. The video gaming community has recently also begun to explore special projections for first person shooter games, that provide gamers with 'enhanced peripheral vision' [15]. However we have found limited literature on the use of Focus+Context views for video telepresence or teleoperation. Of course Wide-angle lenses, including "fisheye lenses" are widely used but are different from our proposed focus plus context views. Wide angle views which preserve perspective (i.e. corresponding to a pinhole camera model) are limited to a maximum FOV of 180 degrees, and become very "area distorted" as the FOV gets larger than about 160 degrees, whereas fisheye views involve some distortion over the entire image. Johnson *et al* report on how field of view affects telepresence collaboration, and experimented with using a main view combined with a context view. They indicate that a focus plus context view is helpful in collaborative telerobotics. However they do not describe the construction of their F+C view in detail, and it is apparent from their figures that it is not implemented as a continuous projection via image processing.

B. Radially Symmetric Image Remapping

The basic task is to produce a view image showing a central undistorted (perspectively correct) region surrounded by a warped larger peripheral image, joined seamlessly (i.e. without discontinuities or artificial corners.) Each pixel of the view image corresponds to some direction in space, and must be provided by the pixel from input camera(s) corresponding to that direction. (Or more generally by an interpolated value of the input pixels closest to the desired direction.)

Many methods of warping are possible, but we first focus on radially symmetric warping functions. Consider a virtual camera pointed in some direction, so that the principle ray corresponds to a vector $\mathbf{n}$ from the optical center through the central pixel. A given pixel at view image coordinates v_x, v_y corresponds to a ray in some direction given by a normal vector $\mathbf{v}$. In the view image, we may convert v_x, v_y to polar coordinates θ, r. Note that the angle α between $\mathbf{n}$ and $\mathbf{v}$ is given by $\alpha = \tan^{-1}(r/f)$ where f is the focal length of an idealized virtual pinhole camera. A natural class of distortions are radially symmetric, described by $\theta \rightarrow \theta, r \rightarrow d(r)$, for some continuous monotonically increasing function $d(\cdot)$. If d is non-monotonic some pixels (view directions) could be replicated in the image. For $d(r) = kr$ for some fixed constant k, the effect is the same as "magnifying" the image uniformly by $1/k$ (equivalent to using a longer focal length to get a narrower field of view.) We can pick $d(r) = kr$ for $r \leq r_0$ to generate a view matching a perspective camera inside the circle with radius r_0. Outside of r_0 we can let $d(r)$ grow faster than linearly, to push more peripheral view into the generated view. As long as $d(r)$ is chosen to be continuous and differentiable we will not introduce any discontinuities into the image, and will avoid introducing corners into the images of lines. For our experiments we have let $d(r)$ grow quadratically as $d(r) = r_0 + k_2(r - r_0)^2$ outside of r_0, but $d(r)$ could be any monotonic function.

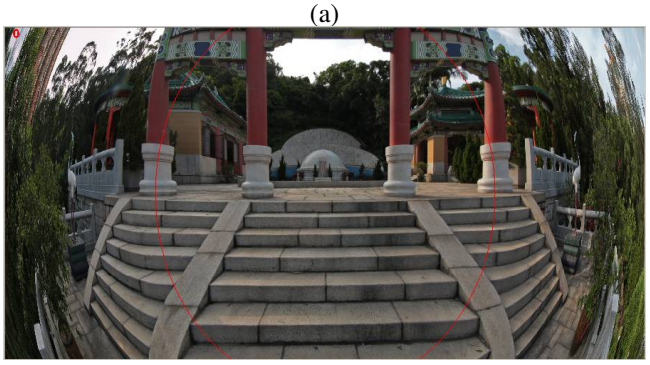
(a)



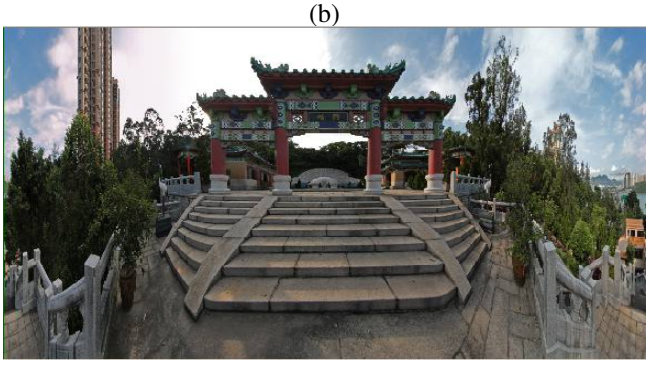
(b)

Fig. 6. Panoramic and perspective views of a scene. (a) shows a complete spherical panorama, showing the view in any direction from the camera, where the x coordinate is proportional to yaw (from 0-360) and the vertical coordinate is proportional to pitch. (b) shows a perspective view with horizontal field of view (HFOV) of about 160 degrees, and displays a large degree of "area stretching" away from the center.

(a)



(b)



(c)

Fig. 7. Focus+Context views with wide periphery. (a) The region inside the thin red circle is an undistorted perspective view with HFOV of 80 degrees. The image has a maximum HFOV of 180 degrees, and is green beyond that. (b) is the same as (a) inside the circle, but has an overall HFOV of 260 degrees and diagonal FOV of 334 degrees, meaning the view shows objects 'behind' the camera. (c) Also contains a perspective view with HFOV of 80 degrees, but is surrounded by a full peripheral HFOV of 360 degrees.

Note that there is an apparent limitation of this method, in that no matter how fast $d(r)$ grows, it cannot generate a field of view greater than 180 degrees. That is because $\alpha = \tan^{-1}(r/f)$ asymptotically approaches $\pi/2$ (that is, 90 degrees from the principal direction $\mathbf{n}$) as $d(r)$ goes to infinity. This can be seen in the top image of Figure 7, where green pixels are provided for directions greater than 180 degrees from the forward direction of the view. However, this limitation may be easily overcome by re-parameterizing view coordinates as θ, α rather than θ, r. That is, we map v_x, v_y to θ, α rather than to θ, r where $\alpha = \tan^{-1}(r/f)$, and then produce the distortion by mapping $\alpha \rightarrow d_\alpha(\alpha)$. Here $d_\alpha(\alpha)$ is $d(r)$ re-parameterized from r to α. So for $\alpha < \alpha_0 = \tan^{-1}(r_0/f)$ if we take $d_\alpha(\alpha) = \tan^{-1}(kf\tan(\alpha))/f$ we will again have a perspectively correct image with magnification $1/k$ inside r_0, but outside r_0, we can let α grow all the way to 180 for a total field of view of 360 degrees.

C. Cylindrically Based Focus + Context Remapping

The radial warping method provides a projection that matches a perspective view inside a circle, and compresses the periphery outside the circle. One consequence of this is that straight lines in the world appear straight inside the circle but become warped outside. This can be unsettling particularly for vertical lines, which are ubiquitous in real world scenes, and help to orient the view. Cylindrical panoramas preserve straightness for vertical lines, which suggests another type of Focus + Context construction. As with the radial construction, we need to determine a value for each pixel of the synthesized view. We start by mapping each image coordinate v_x, v_y to the direction it's corresponding ray would point for a perspective view. This direction can be characterized by yaw β and pitch ϕ values. Then, we choose a yaw threshold β_0 and for values greater than β_0 we let yaw value grow faster than linearly. That is, we map $\beta \rightarrow d_\beta(\beta)$ where $d_\beta(\cdot)$ is linear for $\beta \leq \beta_0$ and grows monotonically faster than linearly for $\beta > \beta_0$. As with the radial mapping methods, this could be done in many ways, but we have chosen d_β as quadratic in the compressed region. The result of this kind of F+C image is shown in Figure 7 (c). Notice that the buildings to the upper left appear vertical, whereas

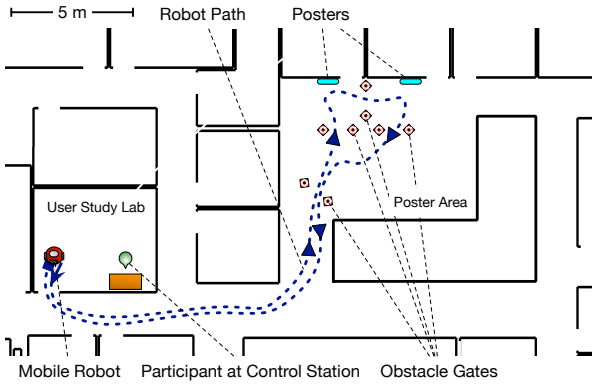


Fig. 8. Depiction of the spatial layout of the experiment.. All participants were instructed to follow the exact same route as indicted by the arrows.

in (b) they curve along a circle. The panoramic image from which these views were derived is shown in Figure 6, along with a perspective view for comparison.

D. Interface for using F+C views

We implemented a simple interface for using Focus+Context views. Dragging on the image moves the focus region allowing a user may pan the view. Double clicking on any point, whether inside the focus region or outside, pans the view so that the clicked point becomes the new center of the focus. We have experimented with this interface for viewing high resolution spherical panoramic images, as well as for spherical and hemispherical video.

The spherical video is recorded with a rig containing 6 GoPro Hero cameras, but currently requires post-processing and can not be streamed live. The remaining sections describe our implementation using live streaming from a Point Grey USB camera with a 185 degree Fujinon fisheye lens.

IV. Preliminary User Study

We conducted a preliminary user study to evaluate both the focus+context viewing and through-the-screen robot control in comparison to baseline techniques. As baselines, we used a perspective camera view with which to compare focus+context and keyboard control (using the arrow keys) as baseline for comparison with through-the-screen.

The goal of the study was to gain usage insights of our system by observing participants while using all combinations of the control and view techniques. In addition we wanted to elicit from the participants a ranking for each of the control and view techniques, and gather free-form comments for each input condition.

A. Participants

We recruited 8 participants from an industrial research lab. 4 participants were female and one participant was left-handed. The majority of participants was between 31 and 35 years old. On average, participants had very little to no experience in controlling mobile robots (M=1.75 on a 5 point Likert scale 1 = *no experience* 5 = *a lot of experience*).

B. Task Description and Apparatus

We needed a task that would enable us to evaluate both our view techniques and our navigation techniques at once. Therefore, we chose a visit to a conference poster session scenario for the task. We thought this would provide realistic usage conditions as it would show the usability of both the view techniques, e.g., when using the view to read posters, and robot control techniques, e.g., when navigating between posters or around obstacles.

Participants were asked to control a mobile robot remotely from within a user study room in our laboratory. Their task was to use the provided user interface to navigate the robot from the initial starting position in the user study room to a mock poster session area. At the poster session area, the participants were asked to maneuver the mobile robot into a position to be able to read two posters placed at fixed locations in the space. After that, participants were to return the mobile robot to the starting location in the user study room.

To realize our planned task scenario, we set up a 8×6 m sized area of our laboratory as a simulated conference poster session. Figure 8 shows the spatial layout of the poster session area and its location relative to the user study room. The poster session area was around 10 m along a straight corridor from the user study room. We hung posters (in a randomized order) in two in the space, and added obstacles to simulate obstacles or attendees at the poster session. Obstacles were configured in a standardized way as directional *gates* the robot would need to pass through in a specific order (see Figure 8) to complete the task. The width of the gates was set to 36 inches (91.44 cm), a standard door width required by most commercial building codes in the USA. Poster and gate locations were the same for all participants.

We used a (differential-drive) Pioneer 3-DX mobile robot base [16] for the experiment. To implement all different view conditions, the robot was fitted with a forward-facing, 185° FOV fisheye camera, mounted on a truss at a height of around 1.46 m. For the perspective view condition, a perspective view with a horizontal FOV of 101° was rendered from the fisheye image.

Participants controlled the robot through a web interface (Figure 1) shows the it with F+C and TTS input) that implemented all combinations view and control conditions. In the conditions using through-the-screen (TTS) control, the interface showed buttons that allowed the user to move the robot forward and back 0.5 m, and turn the robot 45° or 90° to the left or right, respectively. Users were given the option to use these buttons when positioning the robot close to objects (e.g., the posters to be viewed), where field of view constraints would make small corrections of the robot's position via TTS difficult.

C. Experimental Design and Measures

Participants were asked to perform the task with all combinations of input and viewing. Input techniques were *keyboard* and *through-the-screen* (TTS) and viewing techniques were

perspective, *fisheye* and *focus+context* (FC). FC used the radially symmetric remapping technique. Figure 4 shows the actual robot views used in the study.

This resulted in a total of 6 conditions to be evaluated: keyboard+perspective (**C1**), keyboard+fisheye (**C2**), keyboard+F+C (**C3**), TTS+perspective (**C4**), TTS+fisheye (**C5**), TTS+FC (**C6**).

To compensate for any learning effects, the order of conditions was randomized for each participant using a Latin Square design. Prior to starting the experiment, participants were given a trial run in order for the experimenter to familiarize them with all input and viewing techniques.

At the end of each trial, participants were asked to provide free-form feedback about the current teleoperation user interface. At the end of each study session, participants were asked to provide preference rankings for the input methods and viewing methods, respectively.

D. Results: Preference Rankings

The following table summarizes the preference rank counts given to each input method and viewing method:

Technique	Ranked 1st	Ranked 2nd	Ranked 3rd
Keyboard	2	6	–
Through-the-Screen ★	6	2	–
Perspective ★	4	1	3
Fisheye	0	3	5
Focus in Context ★	4	4	0

TABLE I

THE AGGREGATED TECHNIQUE PREFERENCE RANKINGS PROVIDED BY THE PARTICIPANTS. THE STARS INDICATE THE FIRST PREFERENCE OBTAINED WHEN RESOLVING THE VOTE USING THE SCHULZ METHOD (TIES ARE ALLOWED).

In order to determine the preferred input and viewing techniques, respectively, we aggregated the rankings according to the Schulze method [17] using the online tool Condorcet Vote [18]. As shown in Table I, Through-the-Screen was ranked as the most preferred input technique. Perspective and Focus in Context are tied as most preferred viewing techniques. However, we can observe at this point that Focus in Context was never ranked 3rd by any participant.

E. Results: Free-Form Comments

In the following, we summarize the free-form comments we gathered from participants, grouped by input technique and view technique.

1) Input Techniques: As Table I shows, participants generally favored TTS for robot control. In support of this, Participant (P7) stated: *"I like the concept of reaching through the screen. It still feels the most precise."*. However, we obtained some comments that suggest some improvements for TTS. One of the two most common issues were an imprecision for long trajectories—*"Long routes tend to have more errors, so you have to replan your navigation."* (P3). A reason for the imprecision on long trajectories is that the robot relied solely on wheel odometry for this study. This issue could, e.g., be fixed by using local obstacle avoidance based on LIDAR sensors to correct for odometry errors.

The second issue we found for TTS was GUI-related. The TTS interface in this study was set up in a way that only supported a single routing to be executed at a given time. However, some users attempted to append additional path segments while the robot was still executing a path, which led to inconsistent behavior of the robot: *"Sometimes when navigating using the lines, I would add another line path and it would mess up my original path—need to have robot finish the path before adding new lines"*. It is clear that future TTS interfaces will need to support appending segments to original paths or rearranging paths on-the-fly.

2) View Techniques: The free-form comments we received about view techniques mostly reflect the ranking results shown in Table I. Fisheye was generally disliked due to distortions making it difficult to navigate and avoid obstacles (P5, P6, P7) and to view the poster contents (P7, P8).

Perspective and F+C seemed to be liked equally well. There was praise specifically for F+C—*"'This was better than staight fisheye—I could judge the straightline approaches better."* (P5), *"I think I like he F&C interface best"* (P1). Participants did feel more mixed about perspective, and was rated both best—*"This was my favorite of the reach through the screen options due to the view."'* (P7)—and worst—*"'The straight view is my least favorite view"'* (P4)—by participants.

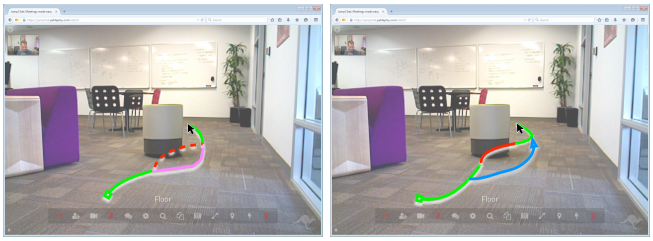
One criticism of F+C was that items in the periphery (distorted view region) were difficult to identify (P6, P7). This issue could be overcome by varying the size of the undistorted view region and adjusting the amount of distortion in the peripheral region. These results indicate that settings of these parameters certainly need to be improved in future versions of F+C. The main criticism of perspective was that users could not see close obstacles as there was no possibility to view the floor near the robot's wheels (P3,P5,P6,P8).

F. Discussion

Our preliminary user study was conducted mainly to observe participants using our proposed input and viewing techniques and baseline techniques. Our results do show a qualitative preference for TTS over keyboard interaction and indicate that users equally enjoyed F+C and perspective view techniques.

User comments indicate that TTS requires additional refinements in the user interface to increase usability. The most important requested UI feature is the ability to add aditional path segments once the robot is in motion.

For the view techniques, participant comments indicate that F+C is a great improvement over Fisheye. It also deals with certain drawbacks of the perspective view, most notably that near-environment is not visible. Furthermore, the participants' comments highlighted the importance that must be given to the design criteria of the F+C view regions: future work may need to analyze (from a user perspective)



(a) Rubber banding around impossible path (b) Indication of path alternative

Fig. 9. (a) Rubber banding of the user's input path to avoid an obstacle. (b) Indication of path alternatives for paths intersecting with obstacles.

the trade-offs in choosing the right size for the undistorted and distorted regions and the degree of distortion (resulting from large field of view) present in the distorted region.

V. Conclusions and Future Work

In this paper, we contribute two novel techniques that can improve the usability of teleoperation interfaces for mobile robots.

First, *Through-the-screen* (TTS), is an input technique that permits teleoperation of a robot by defining a path directly in the robot's camera view. This has two main advantages: firstly, it removes the user from a closed control loop while operating the robot and thus permits the user to address other tasks once a path has been defined for the robot. Secondly, TTS can be advantageous in situations where there is a large amount of lag (e.g., from a transcontinental Internet link) in the connection to the mobile robot. Our preliminary user study indicated that TTS was preferred by participants compared with a keyboard-based control technique. Although disallowed in our experiment, combining TTS and keyboard control may be the most flexible variant, allowing both planning paths ahead of time and instant control of the robot.

Although the qualitative results from our study positively favor TTS, future studies will need to explore the quantitative effects of TTS. To highlight some of the advantages of TTS, a future study may need to incorporate a distractor task, as we believe that TTS can reduce the user's workload while controlling the robot. Given our hypothesis that TTS performs well in high-latency conditions, it would also be interesting to evaluate TTS on a robot connection with simulated network latency.

The second technique is a novel focus-and-context (F+C) view technique, that makes use of image data from wide-angle (fisheye) cameras by preserving an undistorted center region (in contrast to standard fisheye views) while simultaneously providing a view of the periphery in a distorted outer region. F+C addresses the problems associated with a lack of peripheral vision and spatial awareness present in teleoperation interfaces using a standard perspective camera view.

Our preliminary user study showed that F+C was preferred to a standard fisheye view. However, user comments also highlighted the necessity of adjusting certain parameters of the view generation in order to improve the users' perceived quality and utility of F+C.

We currently have a number of ideas for improving TTS. For instance, the naive path planning approach of decomposing waypoints into rotation and linear movement could be improved upon. It would be preferable to have the mobile robot execute "fluid" motions while navigating the user's paths, e.g., using Bézier curves to connect path segments.

On the UI side, we are also thinking of making path planning more interactive. For instance, instead of just marking a path segment that traverses a recognized obstacle red (as shown in Figure 3), the path drawing process could provide additional feedback by enforcing obstacle avoidance through rubber banding, or suggesting improved paths as alternatives (see Figure 9).

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[2] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[3] J. Foote, Q. Liu, D. Kimber, P. Chiu, and F. Zhao, "Reach-through-the-screen: A new metaphor for remote collaboration," in *Advances in Multimedia Information Processing-PCM 2004*. Springer, 2005, pp. 73–80.

[4] M. Baker, R. Casey, B. Keyes, H. A. Yanco, *et al.*, "Improved interfaces for human-robot interaction in urban search and rescue." in *SMC (3)*. Citeseer, 2004, pp. 2960–2965.

[5] B. A. Maxwell, N. Ward, and F. Heckel, "A human-robot interface for urban search and rescue." *AAAI Mobile Robot Competition*, vol. 3, p. 01, 2003.

[6] B. Coltin, J. Biswas, D. Pomerleau, and M. Veloso, "Effective semi-autonomous telepresence," in *RoboCup 2011: Robot Soccer World Cup XV*. Springer, 2012, pp. 365–376.

[7] S. M. Jung, D. Choi, K. Kwon, and J. W. Jeon, "A sketch-based interface for remote robot control on an online video screen," in *Consumer Electronics (ICCE), 2013 IEEE International Conference on*. IEEE, 2013, pp. 428–429.

[8] G. Chronis and M. Skubic, "Sketch-based navigation for mobile robots," in *Fuzzy Systems, 2003. FUZZ'03. The 12th IEEE International Conference on*, vol. 1. IEEE, 2003, pp. 284–289.

[9] D. Sakamoto, K. Honda, M. Inami, and T. Igarashi, "Sketch and run: a stroke-based interface for home robots," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009, pp. 197–200.

[10] U.S. Department of Defense, "Design Criteria For Military Systems, Equipment, And Facilities, MIL-STD-1472F," *Department of Defense Design Criteria Standard, Human Engineering*, 1998.

[11] A. Cockburn, A. Karlson, and B. B. Bederson, "A review of overview+ detail, zooming, and focus+ context interfaces," *ACM Computing Surveys (CSUR)*, vol. 41, no. 1, p. 2, 2008.

[12] P. Baudisch, N. Good, V. Bellotti, and P. Schraedley, "Keeping things in context: a comparative evaluation of focus plus context screens, overviews, and zooming," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2002, pp. 259–266.

[13] X. Zhao, W. Zeng, X. D. Gu, A. E. Kaufman, W. Xu, and K. Mueller, "Conformal magnifier: A focus+ context technique with local shape preservation," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 11, pp. 1928–1941, 2012.

[14] M. Trapp, "Interactive rendering techniques for focus+ context visualization of 3d geovirtual environments," Ph.D. dissertation, University of Potsdam, 2013.

[15] S. E. Williams, "Blinky: proof of concept to put peripheral vision into games," https://github.com/shaunlebron/blinky, see also https://www.youtube.com/watch?v=f9v_XN7Wxh8, 2015.

[16] Mobile Robots, "Pioneer P3-DX," http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx, 2015.

[17] M. Schulze, "A new monotonic and clone-independent single-winner election method," *Voting matters*, vol. 17, no. 1, pp. 9–19, 2003.

[18] Condorcet.Vote, "Condorcet Vote," Retrieved on 9/16/2015 from http://www.condorcet.vote/, 2015.