Um Sistema de Monitoramento para Detecção de Objetos em Tempo Real empregando Câmera em Movimento

Gustavo Carvalho, José F. L. de Oliveira, Eduardo A. B. da Silva, Sergio L. Netto, Gustavo Freitas, Gabriel C. Motta-Ribeiro, and Ramon R. Costa

***Resumo*— Neste trabalho, empregamos uma câmera acoplada a um robô em movimento sobre um trilho retilíneo para detectar, em tempo real, objetos abandonados em um ambiente desordenado. Para tanto, comparamos as imagens capturadas com um vídeo de referência registrando geometricamente ambos os sinais. Implementamos um alinhamento temporal entre os vídeos baseado no conceito de máxima verossimilhança, de modo a permitir uma comparação criteriosa dos dois sinais. Incorporamos uma estratégia de votação temporal entre quadros consecutivos ao sistema, resultando num processo de detecção robusto e em tempo real de objetos abandonados.**

***Palavras-Chave*— Câmera em movimento, detecção de objeto, tempo real, ambiente desordenado, visão computacional**

***Abstract*— In this paper, we use a camera attached to a robot platform along a straight track to detect, in real time, abandoned objects in a cluttered environment. To do so, we compare the captured images to a reference video employing geometric registration between the two signals, with a temporal voting strategy along consecutive frames to increase the detection robustness. A model-based maximum likelihood temporal video alignment is proposed in order to allow proper comparison between the videos. This way, as intended, we were able to successfully find abandoned objects in real time.**

***Keywords*— Moving camera, object detection, real time, cluttered environment, computer vision**

I. INTRODUÇÃO

Técnicas de Visão Computacional podem ajudar a extrair mais e melhores informações de ambientes e processos, permitindo uma economia significativa de recursos. Além disso, sua aplicação em ambientes perigosos também pode minimizar possíveis riscos inerentes ao trabalho a ser executado ou aumentar a eficiência humana, particularmente quando se trata de tarefas repetitivas ou tediosas. Neste contexto, a detecção automática de objetos abandonados poderia ser de grande ajuda durante procedimentos de vigilância e de inspeção remotas. Ao usar câmeras estáticas para este fim, por exemplo, a simples subtração do fundo da imagem e de comportamento, utilizando abordagens estatísticas, permite detectar ou até mesmo rastrear objetos nos vídeos adquiridos [1]–[6]. No entanto, muitas vezes, empregam-se câmeras em movimento a fim de se aumentar a cobertura da vigilância a ser realizada sem se incorrer em grande aumento de custo [7]. Neste caso, técnicas mais elaboradas são necessárias para se compensar o movimento da câmera (que também pode incluir uma quantidade significativa de vibração), bem como para se registrar o vídeo sendo adquirido em relação a um dado sinal de referência [8]–[13]. Além disso, os ambientes a serem monitorados muitas vezes tendem a ser muito carregados (desordenados), tornando mais difícil a obtenção de informações relevantes, contribuindo, desta forma, para a diminuição da robustez da detecção sendo realizada.

Neste trabalho, descrevemos um sistema de vigilância completo usando uma câmera em movimento sobre uma plataforma robótica realizando movimento translacional de ida e volta. O sistema é capaz de detectar objetos anômalos em um ambiente industrial desordenado. Todo o processamento é feito em tempo real, o que requer soluções muito específicas e exigentes de processamento de sinais, e torna o sistema adequado para uma ampla gama de situações práticas.

As principais contribuições deste trabalho são quatro. Na primeira, após a detecção de pontos salientes nos vídeos de referência e alvo, realizamos o registro dos quadros através da imposição de um modelo de câmera translacional durante a aplicação do algoritmo RANSAC [13], [14] para o processo iterativo de remoção de *outliers* seguido do cálculo de homografias. A detecção de alterações entre os vídeos original e alvo é realizada utilizando a conhecida correlação cruzada normalizada (NCC, do inglês *normalized cross-correlation*) [13]. Como segunda contribuição principal, propomos reduzir a complexidade computacional e as detecções incorretas restringindo o cálculo da NCC às áreas onde há uma grande variação de luminância entre os vídeos de referência e alvo. Uma outra contribuição é a introdução de uma estratégia de votação temporal ao longo de quadros consecutivos para melhorar a robustez da detecção. Outra questão relevante em tais sistemas de vigilância é o alinhamento temporal dos vídeos de referência e alvo. Soluções para este problema geralmente incluem a utilização de sinais de gatilho externos para se determinar a posição da câmera. Como exemplo, pode-se citar sinal de GPS [13], a odometria do robô [15], [16], ou um laser [15], [16]. Neste trabalho, propomos evitar o uso desses sinais de gatilho externos realizando estimação de posição baseada em modelos empregando máxima verossimilhança com base em dados de movimento obtidos a partir dos vídeos de referência e alvo.

Este artigo está organizado da seguinte forma: a Seção II

Gustavo Carvalho, José F. L. de Oliveira, Eduardo A. B. da Silva, Sergio L. Netto, Gustavo Freitas, Gabriel C. Motta-Ribeiro, and Ramon R. Costa DEL/POLI, PEE/COPPE, UFRJ, POBox 68504, Rio de Janeiro, RJ, Brazil, E-mails: {gustavo.carvalho, jleite, eduardo, sergioln }@smt.ufrj.br {gfreitas, gabrielcasulari, ramon }@coep.ufrj.br.

descreve a configuração geral do sistema, incluindo o *hardware* (câmera e sua plataforma de movimentação) e as características do *software*. A Seção III detalha todas as soluções de sistemas específicas desenvolvidas neste trabalho, enfatizando as quatro contribuições descritas acima. Na Seção IV são discutidos os resultados experimentais obtidos pelo sistema proposto. A Seção V conclui o artigo destacando suas principais contribuições.

II. DESCRIÇÃO GERAL DO SISTEMA

A configuração geral do sistema de vigilância consiste em uma câmera de alta definição (HD) montada numa plataforma sobre um robô "roomba" realizando um movimento translacional de ida e volta em uma calha metálica. A câmera está conectada a um *notebook* por um cabo ethernet, que tem um trilho deslizante específico ao longo da calha. A configuração que utilizamos é mostrada na Figura 1. O robô se movendo leva cerca de 45 segundos para cobrir todos os 6 m de comprimento da calha. Empregamos uma câmera de rede Axis P1346, com resolução *Full-HD*, obtendo 30 quadros/segundo. Ela pesa 0,63 kg, e consome no máximo 9,6 W.

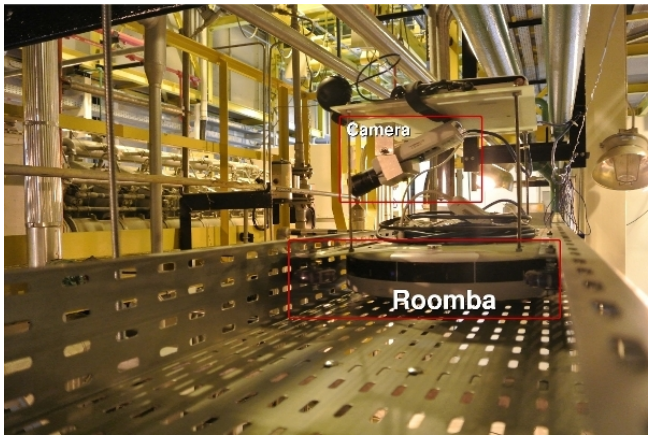

Fig. 1. Configuração real do sistema com um vislumbre do ambiente de interesse ao fundo.

Usamos a seguinte configuração para a operação do sistema em tempo real: num estágio inicial, o vídeo obtido numa determinada passagem do robô é validado por algum operador (vídeo de referência), e os vídeos de todas as passagens subsequentes passam então a ser comparados com esse vídeo de referência. Se necessário, o operador do sistema pode alterar o vídeo de referência usando um procedimento de atualização simples, e depois disso, o sistema de monitoramento retorna à operação normal. Para uma detecção de objetos adequada, dada uma sequência de vídeo, o sistema desenvolvido inclui as seguintes etapas de processamento, melhor detalhadas nas seções seguintes:

- Identificação de pontos de interesse (PoI, do inglês *points of interest*), que geralmente são pontos salientes, para permitir um processamento de vídeo simplificado em tempo real;
- Sincronismo preciso com o vídeo de referência, tanto para o alinhamento inicial quanto para compensar por pequenas variações na velocidade do robô;
- Registro geométrico com a sequência de vídeo de referência, para reduzir efeitos de vibração produzidos pelo movimento do robô ao longo de seu trajeto;
- Comparação de quadros robusta e numericamente eficiente (por conta da restrição de operação em tempo real), de modo a se identificarem discrepâncias significativas;
- Estratégia de votação ao longo de quadros consecutivos, para se identificar discrepâncias temporais consistentes, que devem estar associadas ao objeto de interesse.

III. IMPLEMENTAÇÃO DO SISTEMA

Esta seção descreve em detalhes os estágios de processamento de sinais implementados para uma devida operação em tempo real do sistema de monitoramento apresentado na seção anterior.

A. Alinhamento Inicial dos Vídeos

Em [13], o alinhamento inicial entre os vídeos de referência e alvo é feito usando um GPS, enquanto que em [15] e em [16] a posição do robô é obtida utilizando-se um laser ou a sua própria odometria. Neste trabalho, no entanto, optamos por não empregar sinais externos. Propomos a utilização de uma abordagem de máxima verossimilhança com base nos dados de movimento dos vídeos e num modelo de movimento do robô. O modelo de movimento do robô que utilizamos baseia-se no pressuposto de que o robô move-se ao longo de uma reta com velocidade constante, invertendo o sentido de seu movimento ao atingir as extremidades da calha por onde transita. A posição inicial do robô é desconhecida. Para calcular este modelo, usamos os PoIs dos quadros do vídeo de referência, extraídos como descrito na Subseção III-B. A partir desses pontos, calculamos as homografias entre quadros adjacentes, utilizando o algoritmo RANSAC a fim de remover os *outliers*, também como descrito na Subsecção III-B. A partir destas homografias, podemos caracteriar o movimento de translação da câmera. Ao integrar a componente horizontal (ao longo do percurso) do movimento da câmera, podemos obter o seu deslocamento horizontal em função do número do quadro até uma constante, ao qual podemo-nos referir como $d_r(n)$, onde n é o número do quadro. A Figura 2 mostra um gráfico do deslocamento obtido em função do número de quadros adquiridos de dados reais, onde o r subscrito é utilizado por se tratar do vídeo de referência. Os máximos e mínimos da curva representam os dois extremos do percurso. Deve-se notar que diferentes posições iniciais do robô produziriam curvas com a mesma forma, diferindo apenas em seu valor médio.

A curva obtida é ruidosa devido à vibração da câmera, mas podemos obter um modelo de movimento sem ruído, realizando um ajuste de mínimos quadrados de um modelo linear por partes composto por duas linhas retas de coeficientes angulares opostos. O modelo obtido é sobreposto sobre a curva da Figura 2. Suponha-se que o deslocamento do padrão de movimento é dado pela função $d_m(x)$, onde x é o número do quadro. Nota-se que o modelo obtido é uma função contínua de x. Sem perda de generalidade, podemos supor que uma mudança de sentido no modelo ocorre para $x = 0$, conforme ilustrado na Figura 2.

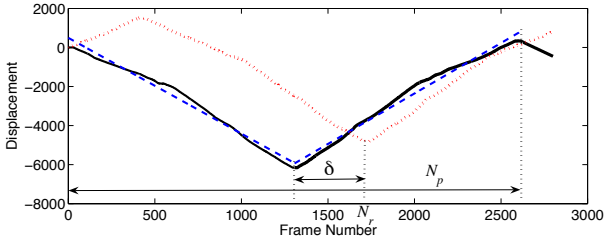

Fig. 2. Exemplo de dados de posição: linha sólida - dados do vídeo de referência; linha tracejada - modelo linear por partes; linha pontilhada - dados do vídeo alvo.

Com o modelo de movimento-padrão obtido, podemos adquirir os parâmetros de movimento do vídeo alvo, do mesmo modo como foi feito para o vídeo de referência. Para isto, formamos um vetor $d_t(n)$, onde n é o número do quadro, contendo a posição no vídeo alvo contra o número do quadro, da mesma forma como ilustrado na Figura 2. Uma vez que não sabemos a posição inicial da câmera, esta função pode ter um nível médio arbitrário. A obtenção do alinhamento dos vídeos de referência e alvo é equivalente a encontrar o deslocamento δ de tal modo que a covariância cruzada entre o modelo $d_m(x-\delta)$ e $d_t(n)$ é máximo. Em outras palavras,

$$\hat{\delta} = \underset{\delta}{\operatorname{argmax}} \left\{ \sum_{i=N_1}^{N_2} (d_m(i-\delta) - \mu_m)(d_t(i) - \mu_t) \right\}, \quad (1)$$

onde μ_m e μ_t são os valores médios das funções $d_m(x)$ e $d_t(n)$, respectivamente.

À primeira vista, seria preciso escolher um intervalo de soma tal que $[N_2 - N_1]$ seja aproximadamente igual ao número de quadros N_p de um movimento de ida e volta completo da câmera. Na Figura 2, $N_p \approx 2600$. No entanto, isto imporia restrições para o funcionamento em tempo real, pois seria preciso processar um movimento completo de ida e volta do vídeo alvo antes de se sincronizar os vídeos de referência e alvo. Para acelerar este processo, restringimos o intervalo de soma $N_2 - N_1$ para Δ (que nos experimentos foi definido como 200 quadros), com o cuidado de garantir que ele contenha ao menos uma mudança de sentido do movimento da câmera. Quando se inicia o processo num quadro arbitrário do vídeo alvo, isto é feito calculando-se o deslocamento da câmera como a soma dos deslocamentos obtidos de um quadro para outro (equivalente à velocidade instantânea da câmera). Como a velocidade da câmera é muito variável devido à vibração, só podemos supor que a câmera está se movendo para a direita (esquerda), quando, nos quadros anteriores, o número de quadros que parecem mover-se para a direita (esquerda) excede o número de quadros que parecem se mover no sentido oposto (para a esquerda) por um limiar T (que nos experimentos foi definido como $T = 7$). Usando essa informação, calculamos o número do quadro N_r, onde ocorre uma mudança de sentido do movimento. Se a mudança for da direita para a esquerda vamos definir o intervalo de pesquisa de δ, na equação 1, igual a $[N_r - \frac{\Delta}{2}, N_r + \frac{\Delta}{2}]$. Se a mudança de direção é da esquerda para a direita definimos este intervalo de pesquisa como $[N_r - \frac{N_p}{2} - \frac{\Delta}{2}, N_r - \frac{N_p}{2} + \frac{\Delta}{2}]$.

B. Detecção dos PoIs e Correspondências

Considerando-se que as duas sequências de vídeo (a recém-adquirida e a de referência) foram devidamente alinhadas no tempo, empregamos o algoritmo SURF (do inglês *speeded-up robust feature*), que é um método para detectar e descrever características locais em imagens, para identificar os PoIs em dois quadros correspondentes de ambos os vídeos [17]. Num passo subsequente, determinamos uma correspondência ponto-a-ponto entre os dois conjuntos de PoIs previamente identificados utilizando o algoritmo RANSAC (do inglês *random sample consensus*), que é um método iterativo para estimar parâmetros de um modelo matemático de um conjunto de dados observáveis que contém *outliers* [13], [14]. Com base nestas correspondências de pontos, aplicamos uma homografia [13], [14] no quadro de referência para permitir uma comparação adequada com o quadro correspondente do vídeo recém-adquirido.

Por ser um método iterativo, o RANSAC pode ser bastante complexo. Para mitigar este problema, decidimos implementar uma remoção inicial de *outliers* empregando a restrição de movimento translacional do nosso modelo: as imagens de referência e alvo são colocadas lado a lado, e desenhamos linhas ligando os pontos correspondentes gerados pelo SURF. Se qualquer linha formar um ângulo com a horizontal fora do intervalo $[-1°, 1°]$, descartamos o par de pontos utilizado para a sua formação. Esta solução é capaz de eliminar um grande número de pares de pontos não consistentes com o fato de a câmera mover-se ao longo de uma calha horizontal. A aplicação do algoritmo RANSAC após esta remoção de *outliers* produz resultados consistentes e robustos.

C. Comparação de Imagens

Uma vez que registramos os dois vídeos, poderíamos empregar a simples subtração entre quadros a fim de detectar os objetos abandonados. No entanto, esta simples subtração muitas vezes não fornece informações significativas em ambientes desordenados, devido à quantidade excessiva de detalhes a serem considerados, como pode ser observado na Figura 3. Neste caso, determinamos a NCC [13] entre as duas imagens, seguida por uma simples detecção de limiar, o que produz uma imagem binária indicando áreas do quadro do vídeo alvo que são candidatas a conterem objetos abandonados.

Calculamos a imagem NCC é da seguinte forma: primeiramente, centralizamos uma janela (matriz) de tamanho N×N (N ímpar) num dado pixel da primeira imagem, enquanto centralizamos outra janela N×N na mesma posição na segunda imagem. Os valores dos pixels da região das imagens que se sobrepõe com as janelas são copiados para elas. Com as janelas preenchidas, estas são normalizadas e, em seguida, calculamos o produto escalar entre elas. O resultado obtido será o valor do pixel, na imagem NCC, da mesma posição do pixel em que as janelas foram centralizadas nas imagens originais. Produzimos a imagem NCC como um todo repetindo este processo para cada pixel das imagens originais.

Como os resultados da NCC são independentes da amplitude das características detectadas, pode ocorrer um grande número de falsos positivos. Neste artigo, propomos lidar com

(a) (b) (c)

Fig. 3. Exemplo de simples subtração de imagens: (a) quadro de referência; (b) quadro alvo; (c) subtração entre os quadros.

esta questão calculando a NCC apenas nas regiões dos quadros em que o valor absoluto da diferença entre os dois quadros registrados for maior do que um limiar. Isto tem a vantagem adicional de reduzir a complexidade computacional associada ao cálculo da NCC.

D. Detecção de Objetos

A fim de reduzir ainda mais os falsos positivos (quando objetos inexistentes são detectados) e falsos negativos (quando objetos abandonados não são detectados pelo sistema) empregamos uma filtragem temporal, como descrito em [13], nas imagens NCC binárias. Fazemos isto alinhando os quadros NCC binarizados através de homografias obtidas utilizando um procedimento semelhante ao descrito na Subseção III-B.

Para aumentar ainda mais a robustez da detecção, incorporamos um processo de votação ao sistema para a escolha final das áreas que serão identificadas como contendo um objeto abandonado. Para isso, calculamos a interseção de um grupo de imagens binárias NCC resultantes do processo de filtragem temporal. Esta interseção identifica pixels como candidatos a pertencerem a um objeto abandonado. Em seguida, aplicamos um procedimento de votação em que se considera que ocorreu uma detecção apenas se o número de vezes que um pixel é candidato a pertencer ao suposto objeto abandonado é maior do que um determinado limiar, definido empiricamente. Utilizamos aqui o mesmo processo de cálculo de homografia empregado na etapa de filtragem temporal, a fim de alinharmos corretamente as imagens a serem comparadas.

IV. RESULTADOS EXPERIMENTAIS

Para desenvolver e testar nossos algoritmos, empregamos a configuração descrita na Seção II, instalada em um galpão contendo geradores de eletricidade e muitos tubos conforme indicado na Figura 4. Os objetos abandonados utilizados foram uma mochila, um guarda-chuva e um casaco, como mostrado na Figura 5 (linha superior). Para a detecção dos pontos de interesse, subamostramos os quadros por 4 em cada direção. Para o cálculo da imagem NCC, subamostramos os quadros por 32 em cada direção. O tamanho da janela NCC variou com o tamanho do objeto a ser detectado, 13×13 para a mochila e o casaco, e 3×3 para o guarda-chuva. Empregamos estes valores porque a janela NCC deve ter dimensões semelhantes às do objeto a ser detectado. Os quadros com os quais calculamos a NCC devem ter as suas dimensões limitadas a fim de manter a complexidade computacional dentro de limites razoáveis, permitindo a operação em tempo real. Isso não representa qualquer problema para a detecção, uma vez que podemos aplicar uma abordagem multi-escala, detectando primeiramente objetos grandes, usando um valor de subamostragem mais elevado para o cálculo da NCC, empregando valores um pouco menores para a subamostragem ao detectarmos objetos um pouco menores, e assim sucessivamente, até que tenhamos detectado todos os objetos, com um aumento na complexidade computacional que é apenas linear com o número de escalas utilizadas.

Fig. 4. Imagem de vídeo de referência de ambiente carregado de elementos.

De modo que o algoritmo proposto funcione corretamente, o objeto a ser detectado deve ser visível por pelo menos 55 quadros, devido às etapas de filtragem temporal e de votação. Portanto, o poder computacional do *hardware* utilizado determina a velocidade máxima do movimento da câmera. Em nossos experimentos, utilizamos um processador Intel Core i7 2630QM, com um *clock* de 2 GHz, e com 8 GB de RAM. Isto permitiu operação em tempo real com subamostragem temporal de 8, o que foi suficiente para que o algoritmo operasse com o robô se movendo com a velocidade descrita na Seção II. Para velocidades de processamento menores é necessária uma subamostragem maior e, portanto, o robô tem de ser mais lento. A Figura 5 mostra os resultados de detecção.

V. CONCLUSÕES

Neste trabalho, propusemos um sistema para detecção em tempo real de objetos abandonados em um ambiente desordenado. Baseamos a nossa proposta em um grupo de técnicas comumente empregadas em problemas semelhantes, como SURF para detecção de pontos de interesse, juntamente com RANSAC para remoção de *outliers* e cálculo

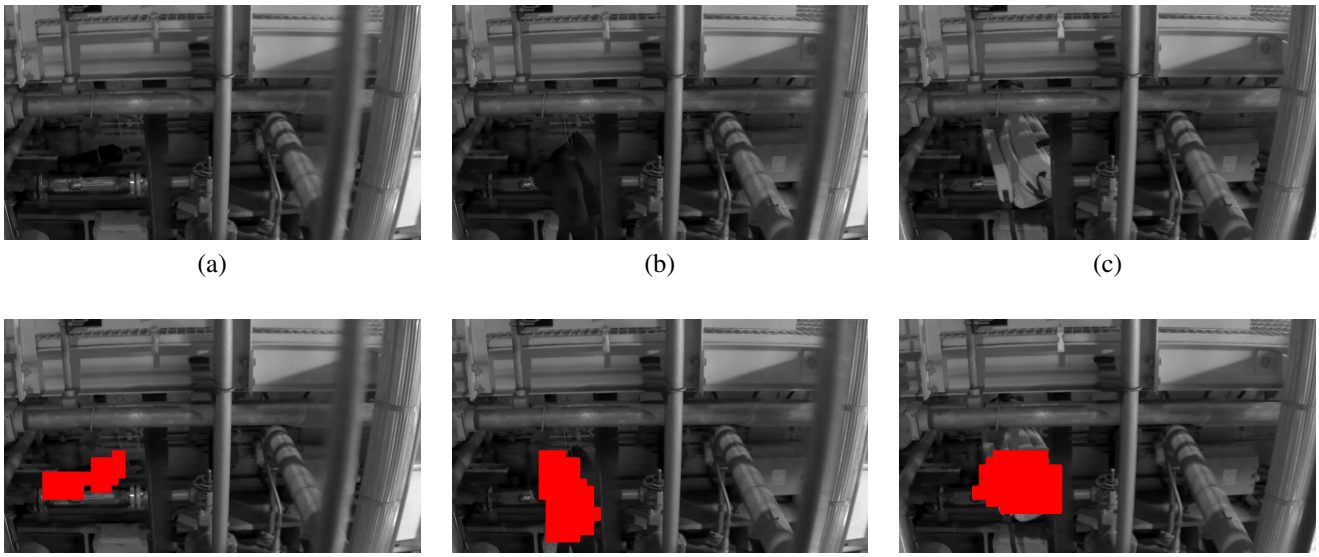

Fig. 5. Exemplos de objetos abandonados: (a) guarda-chuva; (b) casaco; (c) mochila. Resultados da deteção dos objetos abandonados: (d) guarda-chuva; (e) casaco; (f) mochila.

de homografias para registro de quadros. Também usamos a conhecida correlação cruzada normalizada para a detecção de mudanças entre quadros. As contribuições originais deste artigo foram: (i) a remoção inicial de *outliers*; (ii) a utilização da diferença absoluta entre quadros em conjunto com NCC para a geração de um quadro binarizado indicando candidatos a objetos abandonados; (iii) o alinhamento inicial de vídeos utilizando uma abordagem de máxima verossimilhança que torna desnecessário o uso de qualquer sinal de gatilho externo; (iv) um processo de votação entre quadros para aumentar a robustez da detecção.

Testamos o método proposto num ambiente de vigilância desordenado real, consistindo de um galpão com geradores de energia e muitos tubos. Conseguimos detectar com sucesso objetos abandonados em tempo real. Tal sistema tem um bom potencial de aplicação em tarefas de vigilância e inspeção em plataformas de petróleo, onde sua operação econômica pode se beneficiar em muito de tarefas automatizadas.

REFERÊNCIAS

[1] Alessio Dore, Mauricio Soto, and Carlo S. Regazzoni, "Bayesian tracking for video analytics," *IEEE Signal Processing Magazine*, vol. 27, no. 5, pp. 46–55, September 2010.

[2] Venkatesh Saligrama, Janusz Konrad, and Pierre-Marc Jodoin, "Video anomaly identification," *IEEE Signal Processing Magazine*, vol. 27, no. 5, pp. 18–33, September 2010.

[3] Badri Narayan Subudhi, Pradipta Kumar Nanda, and Ashish Ghosh, "A change information based fast algorithm for video object detection and tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 7, pp. 993–1004, July 2011.

[4] YingLi Tian, Rogerio Feris, Haowei Liu, Arun Humpapur, and Ming-Ting Sun, "Robust detection of abandoned and removed objects in complex surveillance videos," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 41, no. 5, pp. 565–576, 2011.

[5] Pierre-Marc Jodoin, Venkatesh Saligrama, and Janusz Konrad, "Behavior subtraction," *IEEE Transactions on Image Processing*, vol. 21, no. 9, pp. 4244–4255, September 2012.

[6] Li Cheng, Minglun Gong, Dale Schuurmans, and Terry Caelli, "Real-time discriminative background subtraction," *IEEE Transactions on Image Processing*, vol. 20, no. 5, pp. 1401–1414, May 2011.

[7] Yoichi Tomioka, Atsushi Takara, and Hitoshi Kitazawa, "Generation of an optimum patrol course for mobile surveillance camera," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 2, pp. 216–224, February 2012.

[8] Yue-Meng Chen and Ivan V. Bajie, "A joint approach to global motion estimation and motion segmentation from a coarsely sampled motion vector field," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 9, pp. 1316–1328, September 2011.

[9] Jae Kyu Suhr, Ho Gi Jung, Gen Li, Seung-In Noh, and Jaihie Kim, "Background compensation for pan-tilt-zoom cameras using 1-D feature matching and outlier rejection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 3, pp. 371–377, March 2011.

[10] Kang Xue, Gbolabo Ogunmakin, Yue Liu, Patricio A. Vela, and Yongtian Wang, "PTZ camera-based adaptive panoramic and multi-layered background model," in *18th IEEE International Conference on Image Processing*, Brussels, Belgium, September 2011.

[11] Jiman Kim, Guensu Ye, and Daijin Kim, "Moving object detection under free-moving camera," in *17th IEEE International Conference on Image Processing*, Hong Kong, September 2010.

[12] Yi Xie, Liang Lin, and Yunde Jia, "Tracking objects with adaptive feature patches for PTZ camera visual surveillance," in *20th International Conference on Pattern Recognition*, Istanbul, Turkey, August 2010, pp. 1739–1742.

[13] Hui Kong, Jean-Yves Audibert, and Jean Ponce, "Detecting abandoned objects with a moving camera," *IEEE Transactions on Image Processing*, vol. 19, no. 8, pp. 2201–2210, August 2010.

[14] Richard Hartley and Andrew Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge, U.K, 2nd edition, 2003.

[15] Abhijit Kundu, C. V. Jawahar, and K Madhava Krishna, "Realtime moving object detection from a freely moving monocular camera," in *IEEE International Conference on Robotics and Biomimetics*, Tianjin, China, December 2010, pp. 1635–1640.

[16] Guilherme N. DeSouza and Avinash C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 237–267, February 2002.

[17] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.